import pandas as pd

from datetime import datetime

import numpy as np

import statistics as st

import matplotlib.pyplot as plt

import seaborn as sns

import pickle

import numpy as np

import warnings

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

import xgboost as xgb

import re

from sklearn.model\_selection import GridSearchCV

import warnings

from sklearn.metrics import accuracy\_score

warnings.filterwarnings("ignore")

data = pd.read\_table("cleaned\_data.csv", sep=",", encoding="utf-8")

print(data.columns)

# build\_xgb(data):

# get\_mean\_rank

select = ["ITEM", "ORDERDR", 'BED\_chair',

"DEPT", "POS", "SEX", 'PLACE',

"MODEL\_NAME", "IO"]

for col in select:

value = data['total'].groupby(data[col]).mean().rank() - 1

exec("%s = value.to\_dict()" % col) # 動態變數

# 新增欄位 內容為排名 -> mean\_rank

for col in select:

data[col + "\_n"] = data[col].apply(lambda x: eval("%s[x]" % str(col)))

# 控制建模放入的變數

select = ["ITEM\_n", "ORDERDR\_n","DEPT\_n",

"total","SEX\_n", "POS\_n","AGE",

"PLACE\_n", "IO\_n", 'PNO']

data = data[select]

check = data.columns

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

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

#

y\_train = y\_train.drop(['PNO'], axis=1)

#

X\_train = X\_train.as\_matrix()

X\_test = X\_test.as\_matrix()

#

MR\_params = {'n\_estimators': [35,36,37,38,39,40]}

MR\_params = {'max\_depth': [3, 4, 5, 6, 7, 8, 9, 10], 'min\_child\_weight': [1, 2, 3, 4, 5, 6]}

MR\_params = {'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]}

MR\_params = {'subsample': [0.6, 0.7, 0.8, 0.9], 'colsample\_bytree': [0.6, 0.7, 0.8, 0.9]}

MR\_params = {'reg\_alpha': [0.05, 0.1, 1, 2, 3], 'reg\_lambda': [0.05, 0.1, 1, 2, 3]}

MR\_params = {'learning\_rate': [0.01, 0.05, 0.1, 0.2]}

#調動上面修改下面 一行一行調 每調動一次請把最佳結果填入下面參數

# mr\_params\_list = [{'n\_estimators': [35,36,37,38,39,40]},

# {'max\_depth': [3, 4, 5, 6, 7, 8, 9, 10], 'min\_child\_weight': [1, 2, 3, 4, 5, 6]},

# {'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]},

# {'subsample': [0.6, 0.7, 0.8, 0.9], 'colsample\_bytree': [0.6, 0.7, 0.8, 0.9]},

# {'reg\_alpha': [0.05, 0.1, 1, 2, 3], 'reg\_lambda': [0.05, 0.1, 1, 2, 3]},

# {'learning\_rate': [0.01, 0.05, 0.1, 0.2]}]

# mr\_params\_dict = {

# 'n\_estimators': [35,36,37,38,39,40],

# 'max\_depth': [3, 4, 5, 6, 7, 8],

# 'min\_child\_weight': [ 2, 3, 4, 5, 6],

# 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5],

# 'subsample': [0.6, 0.7, 0.8, 0.9],

# 'colsample\_bytree': [0.6, 0.7, 0.8, 0.9],

# 'reg\_alpha': [0.05, 0.1, 1, 2, 3],

# 'reg\_lambda': [0.05, 0.1, 1, 2, 3],

# 'learning\_rate': [0.01, 0.05, 0.1, 0.2]

# }

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 39, 'max\_depth': 4, 'min\_child\_weight': 5,

'seed': 0,'subsample': 0.6, 'colsample\_bytree': 0.8, 'gamma': 0.1,

'reg\_alpha': 3, 'reg\_lambda': 0.1}

gbm = xgb.XGBRegressor(\*\*other\_params)

optimized\_GBM = GridSearchCV(estimator=gbm, param\_grid=MR\_params,

scoring='r2', cv=5, verbose=1, n\_jobs=-1)

optimized\_GBM.fit(X\_train, y\_train)

evalute\_result = optimized\_GBM.cv\_results\_

print('引數的最佳取值：{0}'.format(optimized\_GBM.best\_params\_))

# print('每輪迭代執行結果:{0}'.format(evalute\_result))

# print('最佳模型得分:{0}'.format(optimized\_GBM.best\_score\_))

gbm = xgb.XGBRegressor(learning\_rate=0.1, n\_estimators=38, max\_depth=4,

min\_child\_weight=5, seed=0, subsample=0.6, colsample\_bytree=0.6,

gamma=0.1, reg\_alpha=2, reg\_lambda=2)

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

predictions = gbm.predict(X\_test)

def MAPE(true, pred):

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

global y\_test

y\_test['pred'] = pred.astype(int)

y\_test['diff']= diff.astype(int)

print(y\_test)

return np.mean(diff / true)

print('xgb模型的MAPE :', MAPE(y\_test['total'].astype(int), predictions))

def MAPE(true, pred):

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

global y\_test

y\_test['pred'] = pred.astype(int)

y\_test['diff']= diff.astype(int)

print(y\_test)

y\_test.to\_csv('Result.csv')

y\_test.to\_csv('Result.csv')

# print("np.array(true): ", np.array(true).astype(int))

np.savetxt("true3.csv", true, delimiter=",")

# print(type(true.astype(int)))

# print("np.array(pred): ", np.array(pred))

np.savetxt("pred3.csv", np.array(pred), delimiter=",")

# print("diff: ", diff)

np.savetxt("diff3.csv", np.array(diff), delimiter=",")

# print("y\_test: ", true)

return np.mean(diff / true)

# 模型估計的mape

print('xgb模型的MAPE :', MAPE(y\_test['total'].astype(int), predictions))

#視覺化

import seaborn as sns; sns.set()

import matplotlib.pyplot as plt

df = y\_test.copy()

# print(df)

ax = sns.barplot(x="PNO", y="pred",color='pink', data=df[0:50])

ax = sns.barplot(x="PNO", y="total",facecolor=(1, 1, 1, 0),edgecolor=".2", data=df[0:50])

ax.figure.savefig("output.png")

pickle.dump(gbm, open("pima\_20190915\_xgb.pickle", "wb"))