from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import RandomForestClassifier

#from xgboost import XGBRegressor as XGBR

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor

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

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

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import OneHotEncoder

from sklearn.decomposition import PCA

from sklearn.feature\_selection import RFE

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error as MSE

from sklearn.metrics import mean\_absolute\_error as MAE

from sklearn.metrics import log\_loss

import matplotlib

import matplotlib.pyplot as plt

import matplotlib.cm as cm

#import shap

import numpy as np

import pandas as pd

import os

from pandas import DataFrame as df

from sklearn.preprocessing import LabelEncoder

import re

from sklearn.metrics import confusion\_matrix

#from sklearn.metrics import balanced\_accuracy\_score

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

**# load Data**

data8 = pd.read\_csv(r''./Pb\_0822\_5.csv,engine='python')

df = pd.DataFrame(data8)

plt.figure(figsize=(6, 4))

plt.boxplot(df['Pb\_NPs'])

plt.title('Boxplot of Pb\_NPs')

plt.show()

Q1 = df['Pb\_NPs'].quantile(0.25)

Q3 = df['Pb\_NPs'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df = df[(df['Pb\_NPs'] >= lower\_bound) & (df['Pb\_NPs'] <= upper\_bound)]

df = df[(df['Pb\_NPs'] > 4.78)]

df.iloc[:, :1] = df.iloc[:, :1].astype(str)

df.fillna(df.mean())

data0\_8 = df.fillna(df.mean())

y = data0\_8['Pb\_NPs']

print(y)

columns\_to\_drop = ['No.', 'Ag\_NPs', 'Cr\_NPs', 'Cu\_NPs', 'Fe\_NPs', 'Hg\_NPs', 'Mn\_NPs', 'Pb\_NPs', 'Ti\_NPs']

x = data0\_8.drop(columns=columns\_to\_drop, axis=1)

Xtrain,Xtest,Ytrain,Ytest = train\_test\_split(x,y,test\_size=0.2,random\_state=130)

rfc = RandomForestRegressor(random\_state=60)

CV\_score = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

regressor = rfc.fit(Xtrain, Ytrain)

CV\_predictions = cross\_val\_predict(rfc, Xtrain,Ytrain,cv=5)

rmse = np.sqrt(mean\_squared\_error(Ytrain,CV\_predictions))

score\_test = regressor.score(Xtest,Ytest)

print("5cv:",CV\_score,"TEST:",score\_test)

print("rmse\_5CV",rmse)

for i in range(500):

Xtrain,Xtest,Ytrain,Ytest = train\_test\_split(x,y,test\_size=0.2,random\_state=i)

rfc = RandomForestRegressor(random\_state=60)

CV\_score = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

regressor = rfc.fit(Xtrain, Ytrain)

score\_test = regressor.score(Xtest,Ytest)

if CV\_score>0.4 and score\_test>0.4:

print("5cv:",CV\_score,"TEST:",score\_test,"random\_state:",i)

Xtrain,Xtest,Ytrain,Ytest = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

rfc = RandomForestRegressor(random\_state=60)

CV\_score = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

regressor = rfc.fit(Xtrain, Ytrain)

CV\_predictions = cross\_val\_predict(rfc, Xtrain,Ytrain,cv=5)

rmse = np.sqrt(mean\_squared\_error(Ytrain,CV\_predictions))

score\_test = regressor.score(Xtest,Ytest)

print("5cv:",CV\_score,"TEST:",score\_test)

print("rmse\_5CV",rmse)

**#Parameter optimization and five-fold cross validation and parameter optimization**

**#random\_state**

score\_5cv\_all = []

for i in range(0, 200, 1):

rfc =RandomForestRegressor(random\_state=i)

score\_5cv =cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

score\_5cv\_all.append(score\_5cv)

pass

score\_max\_5cv = max(score\_5cv\_all)

random\_state\_5cv = range(0, 200)[score\_5cv\_all.index(max(score\_5cv\_all))]

print("Best\_5cv score：{}".format(score\_max\_5cv),

"random\_5cv:{}".format(random\_state\_5cv))

**#n\_estimators**

score\_5cv\_all = []

for i in range(1, 400, 1):

rfc = RandomForestRegressor(n\_estimators=i

, random\_state=random\_state\_5cv)

score\_5cv = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

score\_5cv\_all.append(score\_5cv)

pass

score\_max\_5cv = max(score\_5cv\_all)

n\_est\_5cv = range(1,400)[score\_5cv\_all.index(score\_max\_5cv)]

print("Best\_5cv score：{}".format(score\_max\_5cv),

"n\_est\_5cv:{}".format(n\_est\_5cv))

**#max\_depth**

score\_5cv\_all = []

for i in range(1, 300, 1):

rfc = RandomForestRegressor(n\_estimators=n\_est\_5cv

,random\_state=random\_state\_5cv

,max\_depth=i)

score\_5cv = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

score\_5cv\_all.append(score\_5cv)

pass

score\_max\_5cv = max(score\_5cv\_all)

max\_depth\_5cv = range(1,300)[score\_5cv\_all.index(score\_max\_5cv)]

print("Best\_5cv score：{}".format(score\_max\_5cv),

"max\_depth\_5cv:{}".format(max\_depth\_5cv))

**#max\_features**

score\_5cv\_all = []

for i in range(1,x.shape[1]+1):

rfc = RandomForestRegressor(n\_estimators=n\_est\_5cv

,random\_state=random\_state\_5cv

,max\_depth=max\_depth\_5cv

,max\_features=i)

score\_5cv = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

score\_5cv\_all.append(score\_5cv)

pass

score\_max\_5cv = max(score\_5cv\_all)

max\_features\_5cv = range(1, x.shape[1]+1)[score\_5cv\_all.index(score\_max\_5cv)]

print("Best\_5cv score：{}".format(score\_max\_5cv),

"max\_features\_5cv:{}".format(max\_features\_5cv))

rfc = RandomForestRegressor(random\_state=6

,n\_estimators=5

,max\_depth=9

,max\_features=4)

RandomForest = rfc.fit(Xtrain, Ytrain)

test\_predictions = RandomForest.predict(Xtest)

score\_test = RandomForest.score(Xtest,Ytest)

score\_train = RandomForest.score(Xtrain,Ytrain)

CV\_score = cross\_val\_score(rfc, Xtrain, Ytrain, cv=5).mean()

rmse = np.sqrt(mean\_squared\_error(Ytest,test\_predictions))

mae = MAE(Ytest, test\_predictions)

print("train:",score\_train)

print("test:",score\_test)

print("score\_5cv",CV\_score)

print("rmse\_test",rmse)

print("MAE",mae)

expvspred\_test = {'Exp':Ytest,'Pred':test\_predictions}

predictions = RandomForest.predict(x)

pre\_Pb = {'Exp':y,'Pred':predictions}

pd.DataFrame(pre\_Pb).to\_excel('./Pb\_pre\_0822\_5\_test\_largest.xlsx')

rfc.score(Xtest,Ytest)

**#Feature importance**

features\_import = pd.DataFrame(Xtrain.columns, columns=['feature'])

features\_import['importance'] = rfc.feature\_importances\_

features\_import.to\_excel('./Pb\_imp\_0822\_5\_test\_largest.xlsx', index=False)

features\_import

**#Prediction**

dp = pd.read\_csv(r''./prediction\_Pb\_0822\_5.csv,engine='python')

dp.iloc[:, :1] = dp.iloc[:, :1].astype(str)

dp.fillna(dp.mean())

dp0\_2 = dp.fillna(dp.mean())

columns\_to\_drop = [ 'leachate\_Cr','leachate\_Mn']

dp0\_3=dp0\_2.drop(columns=columns\_to\_drop, axis=1)

print(dp0\_3)

predictions=RandomForest.predict(dp0\_3)

pd.DataFrame(predictions).to\_excel('./prediction\_Pb\_0822\_5\_global-95.xlsx')