Supplementary material (S1). Python Code

```
from sklearn import metrics
import pandas as pd
import numpy as np
import xlrd
import xlsxwriter
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split as split
from sklearn.model_selection import LeaveOneOut
from sys import stdout
import re
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

workbook = xlrd.open_workbook(r"path file.xlsx","rb")
sheets = workbook.sheet_names()
sheet_num=1
sh = workbook.sheet_by_name(sheets[(sheet_num)-1])
n_cols, n_rows, m_cols, m_rows =0, 0, 0, 0
M_rows=sh.nrows-m_rows
M_cols=sh.ncols-m_cols
required_data = []
for rownum in range(n_rows,M_rows):
   row_valaues = sh.row_values(rownum)
   required_data.append(row_valaues[n_cols:M_cols])
Required_data=np.asarray(required_data)
X=Required_data[1:63,8:43].astype(float)
X= np.array (X)
Y=Required_data[1:63,7:8].astype(float).ravel()
print(X.shape)
bands = Required_data[0:1, 8:43]
bands= np.array (bands)
X = MinMaxScaler().fit(X).transform(X)
X_train, X_test, Y_train, Y_test = split(X, Y, test_size=(0.2), random_state=0)

Y_test_true = [] ; Y_test_pedict= [];max_depth1_2 =[]
min_samples_leaf1_2 =[] ; max_leaf_nodes1_2 =[]
RMSE_best = []; N_best = []; best_index = []; bands_best = []; mse_cv_final = []
r2_val_CV = []; Rc2_train = []; Rmse_train1 =[]; R2_test= []; RMSE_test= []
coffecifient1 = []; coefficient_features = []; Yprdict_test = []; Ytrue_test = []
Y_test_true_cv = [];Y_test_pedict_tcv = [] ; Rc2_test =[] ; Rmse_test1 = []

index = min(X_train.shape)
k = min(X_train.shape)
for j in range (0,34):
   n=min(X_train.shape)
   max_depth1_1 =[]
```

```
min_samples_leaf1_1 =[]
max_leaf_nodes1_1 =[]
mse_cv = []
Y_test_val_cv = []
Y_test_val_pred_cv = []
r2_val_final =[]
rpd_cv1 = []
for max_depth1 in  (1, 3, 5, 7, 9):
   for min_samples_leaf1 in  (2, 4, 6, 8, 10):
      for max_leaf_nodes1 in (None, 10, 20, 30, 40, 50):
         print ('max_depth1','min_samples_leaf,', 'max_leaf_nodes1')
         print (max_depth1,min_samples_leaf1,max_leaf_nodes1)
         loo = LeaveOneOut()
         loo.get_n_splits(X_train)
         r2_train=[]
         rmse_val=[]
         rmse_train=[]
         Y_test_val_pred1 = []
         Y_train_val_pred1 = []
         Y_train_val1 = []
         Y_test_val1 = []
         r2_val = []
         for train_index, test_index in loo.split(X_train):
            X_train_val, X_test_val = X_train[train_index], X_train[test_index]
            Y_train_val, Y_test_val = Y_train[train_index], Y_train[test_index
            model = DecisionTreeRegressor(max_depth =max_depth1,
              min_samples_leaf = min_samples_leaf1, max_leaf_nodes=max_leaf_nodes1
              ,random_state=0)
            model_val=model.fit(X_train_val, Y_train_val)
            Y_test_val_pred=model_val.predict(X_test_val)
            Y_train_val_pred=model_val.predict(X_train_val)
            Y_test_val_pred1.append (Y_test_val_pred[0])
            Y_test_val1.append (Y_test_val)
            r2_train.append(metrics.r2_score(Y_train_val, Y_train_val_pred))
            rmse_val.append(np.sqrt(metrics.mean_squared_error(Y_test_val, Y_test_val_pred)))

         Y_test_val = np.array (Y_test_val1)
         Y_test_val_pred = np.array (Y_test_val_pred1)
         r2_val.append(np.mean (metrics.r2_score(Y_test_val, Y_test_val_pred))) ## accuracy of test-cv
         print ('accuracy of test-CV=', r2_val)
         print('rmse of test_CV='+str(np.mean(rmse_val)))
         max_depth1_1.append (max_depth1)
         min_samples_leaf1_1.append (min_samples_leaf1)
         max_leaf_nodes1_1.append (max_leaf_nodes1)
         mse_cv.append(np.mean(rmse_val))
         r2_val_final.append (r2_val)
         Y_test_val_cv.append((Y_test_val))
         Y_test_val_pred_cv.append((Y_test_val_pred))

# After selecting all the parameters, we need to train the model with the best parameters
msemin = np.argmin(mse_cv)
mse_less = min (mse_cv)
```

```
max_depth = max_depth1_1 [msemin]
min_samples_leaf = min_samples_leaf1_1 [msemin]
max_leaf_nodes = max_leaf_nodes1_1 [msemin]
Y_test_val_cv = Y_test_val_cv [msemin]
Y_test_val_pred_cv = Y_test_val_pred_cv [msemin]
r2_val = r2_val_final[msemin]
model = DecisionTreeRegressor(max_depth =max_depth,
  min_samples_leaf = min_samples_leaf, max_leaf_nodes=max_leaf_nodes
  ,random_state=0)

BPNN_model =model.fit(X_train, Y_train)
## print data
Y_calibrate=BPNN_model.predict(X_train)
Y_pred = BPNN_model.predict(X_test)
Rc2=metrics.r2_score(Y_train, Y_calibrate)
Rt2=metrics.r2_score(Y_test, Y_pred)
msec=metrics.mean_squared_error(Y_train, Y_calibrate)
msec_test =metrics.mean_squared_error(Y_test, Y_pred)

RMSEC=np.sqrt(msec)
RMSEC_test=np.sqrt(msec_test)
print('Rc2_train= '+str(Rc2))
print('RMSEc_train= '+str(RMSEC))
print('Rc2_test= '+str(Rt2))
print('RMSEc_test= '+str(RMSEC_test))
coffecifient = BPNN_model.feature_importances_
sorted_ind = np.argsort(np.abs(BPNN_model.feature_importances_))
X_train = X_train[:,sorted_ind]
X_test = X_test[:,sorted_ind]
bands = bands [:,sorted_ind]
coefficient_features.append(coffecifient)
print ("number of indices = ", min (X_train.shape) )
print ('The end-------------------------------------------')
bands_best.append (bands)
RMSE_best.append (mse_less)
best_index.append (X_train)
r2_val_CV.append (r2_val)
mse_cv_final.append (mse_less)
RPD_final.append (rpd_cv1)
Rc2_train.append (Rc2)# R2_train
Rmse_train1.append (RMSEC)
Rc2_test.append (Rt2)# R2_train
Rmse_test1.append (RMSEC_test)
Y_test_true.append (Y_test)
Y_test_pedict.append (Y_pred)
max_depth1_2.append (max_depth)
min_samples_leaf1_2.append (min_samples_leaf)
max_leaf_nodes1_2.append (max_leaf_nodes)
#===
print ('The second loop')
X_train = X_train[:, 1:]
X_test = X_test[:, 1:]
```

```
    bands = bands [:, 1:]

# The least RMSE
A_less_mse = np.argmin (RMSE_best)
# the best parameter
max_depth = max_depth1_2 [A_less_mse]
min_samples_leaf = min_samples_leaf1_2 [A_less_mse]
max_leaf_nodes =  max_leaf_nodes1_2 [A_less_mse]
#==========
A_best_index = best_index [A_less_mse]
A_bands = bands_best [A_less_mse]
r2_val_CV1 =  r2_val_CV[A_less_mse]  # R2_CV
mse_cv_final1 =  mse_cv_final[A_less_mse]  # RMSE_CV
Rc2_train1 =  Rc2_train[A_less_mse]   # R2_train
RMSE_train11 =   Rmse_train1 [A_less_mse]
Rc2_test111 =   Rc2_test [A_less_mse]
Rmse_test111 =   Rmse_test1 [A_less_mse]
print ('----------------------')
#print ("The best N_component =",A_N_best_final)
print ('max_depth=', max_depth)
print ('min_samples_leaf=', min_samples_leaf)
print ('max_leaf_nodes=', max_leaf_nodes)
print ("The best band =",A_bands)
print ('r2_val_CV=', r2_val_CV1)
print ('mse_cv=', mse_cv_final1)
print ('RPD_cv=', RPD_final1)
print ('Rc2_train=', Rc2_train1)
print ('RMSE_train=', RMSE_train11)
print ('Rc2_test=', Rc2_test111)
print ('RMSE_test=', Rmse_test111)
```