Electronic Supplementary Material (ESI) for RSC Advances.

Supplementary information 1: The source code of Boost GRNN

Environment: Python 3.7

IDE: Jupyter Notebook

Data set: GHSR1α agonist data set

```
import warnings

warnings.filterwarnings('ignore')

%config InlineBackend.figure_formats = {'png', 'retina'}

%matplotlib inline

import numpy as np

import pandas as pd

from sklearn import preprocessing

import numpy as np

import pandas as pd

from sklearn.feature_selection import SelectfromModel

from sklearn.linear_model import Lasso

from sklearn import ensemble

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

from sklearn.metrics import r2_score
```

```
import numpy as np

from sklearn import svm

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.feature_selection import VarianceThreshold
```

```
raw_data = pd.read_csv('ghsr_50_y.csv',index_col = 0)

y = raw_data['Y'].values.ravel()

X = raw_data.drop(["Y"], axis = 1).values

featnames=list(raw_data.drop(["Y"], axis = 1).columns)

min_max_scaler = preprocessing.MinMaxScaler()

X_minmax = min_max_scaler.fit_transform(X)

var = VarianceThreshold(threshold=0.06)

X_var = var.fit_transform(X_minmax)

X_scaled = preprocessing.scale(X_var)

lasso = Lasso(alpha=0.034, max_iter=1000)

lasso.fit(X_scaled, y)

model = SelectfromModel(lasso, prefit=True)

X_new = model.transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=153)

model1=ensemble.RandomForestRegressor( random_state=25,n_estimators=10)

model1.fit(X_train, y_train)
```

```
predict1 = model1.predict(X_train)

predict2 = model1.predict(X_test)
```

```
X_scaled.shape

random_pred = model1.predict(X_new)
```

```
yc = pd.read_csv(r'F:\TzqProgram \GHSR_DOCKING_SELECTED30.csv')

Z = yc.iloc[:, 61:]

print("Z:", Z.shape)

Z_minMax = min_max_scaler.fit_transform(Z)

print("Z_minMax:", Z_minMax.shape)

Z_var = var.transform(Z_minMax)

print("Z_var:", Z_var.shape)

Z_scaled =preprocessing.scale(Z_var)

Z_new = model.transform(Z_scaled)

Z_new.shape

Z_scaled.shape
```

```
random_pred_new = model1.predict(Z_new)
```

```
import pandas as pd

import numpy as np

import time

import matplotlib.pyplot as plt
```

```
from mpl_toolkits.mplot3d import Axes3D

from sklearn import preprocessing

from sklearn.metrics import mean_squared_error as MSE

from sklearn.model_selection import train_test_split, GridSearchCV

from pyGRNN import GRNN

from pyGRNN import feature_selection as FS

GRNN_DATA = pd.read_csv(r"F:\TzqProgram\ghsr_50_y.csv", index_col = 0)

from sklearn.feature_selection import VarianceThreshold



y = GRNN_DATA['Y'].values.ravel()

X = GRNN_DATA.drop(["Y"], axis = 1).values

V = X.copy

title = GRNN_DATA.drop(["Y"], axis = 1).columns

var = VarianceThreshold(threshold=3500)

X_var = var.fit_transform(X)

min_max_scaler = preprocessing.MinMaxScaler()

X_minmax = min_max_scaler.fit_transform(X_var)

X_train, X_test, y_train, y_test = train_test_split(X_minmax, y, test_size=0.25, random_state = 142)

featnames=list(GRNN_DATA.drop(["Y"], axis = 1).columns)

IsotropicSelector = FS.Isotropic_selector()
```

```
X_train_BestSet = X_train

X_test_BestSet = X_test

IGRNN = GRNN()

params_IGRNN = {'kernel':["RBF"],

                'sigma' : list(np.arange(0.1, 4, 0.01)),

                'calibration' : ['None']

                 }

grid_IGRNN = GridSearchCV(estimator=IGRNN,

                          param_grid=params_IGRNN,

                          scoring='neg_mean_squared_error',

                          cv=5,

                          verbose=1,

                          n_jobs = -1

                          )

grid_IGRNN.fit(X_train_BestSet, y_train.ravel())

best_model = grid_IGRNN.best_estimator_

y_pred = best_model.predict(X_test_BestSet)

mse_IGRNN = MSE(y_test, y_pred)

ypred_all = best_model.predict(X_minmax)

grnn_pred = ypred_all
```

```
X_minmax.shape

random_pred_l = []

for pr in random_pred:

    random_pred_l.append(pr)

grnn_pred_l = []

for gp in grnn_pred:

    grnn_pred_l.append(gp)

BG_DATA = pd.DataFrame({ 'grnn_pred':grnn_pred_l,'random_pred':random_pred_l,
'Y':y})

BG_DATA

writer = pd.ExcelWriter('boost_input_train.xlsx')

BG_DATA.to_excel(writer,float_format='%.5f')

writer.save()

GRNN_DATA_YC                                                                    =
pd.read_csv(r"C:\Users\tzq\Desktop\gshr\GHSR_DOCKING_SELECTED30_BG.csv
", index_col = 0)

Z= GRNN_DATA_YC.drop(["YY"], axis = 1).values

Z_var = var.transform(Z)

Z_minMax = min_max_scaler.fit_transform(Z_var)

Z_scaled =preprocessing.scale(Z_minMax)

grnn_pred_new = best_model.predict(Z_scaled)
```

```
rpnl = []

for rpn in random_pred_new:

        rpnl.append(rpn)

gpnl = []

for gpn in grnn_pred_new:

        gpnl.append(gpn)

BG_DATA_new = pd.DataFrame({'random_pred_new':rpnl, 'grnn_pred_new':gpnl})

BG_DATA_new

writer = pd.ExcelWriter('boost_input_predict.xlsx')

BG_DATA_new.to_excel(writer,float_format='%.5f')

writer.save()
```

```
import numpy as np

import pandas as pd

from sklearn import preprocessing

import numpy as np

from sklearn.feature_selection import SelectfromModel

from sklearn.linear_model import Lasso

from sklearn import ensemble

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error
```

```
from sklearn.metrics import r2_score

from sklearn.ensemble import GradientBoostingRegressor



BG = pd.read_csv(r"C:\Users\tzq\Desktop\gshr\boost_input_train.csv", index_col = 0)

y = BG['Y'].values.ravel()

X = BG.drop(["Y"], axis = 1).values

featnames=list(BG.drop(["Y"], axis = 1).columns)

min_max_scaler = preprocessing.MinMaxScaler()

X_minmax = min_max_scaler.fit_transform(X)

lasso = Lasso(alpha=0.001, max_iter=800)

lasso.fit(X_minmax, y)

model = SelectfromModel(lasso, prefit=True)

X_new = model.transform(X_minmax)



from sklearn.ensemble import GradientBoostingRegressor

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=99)

model_BG=ensemble.GradientBoostingRegressor(random_state=81,n_estimators=99, learning_rate=0.199)

model_BG.fit(X_train, y_train)

predict1 = model_BG.predict(X_train)
```

```
true1 = y_train

predict2 = model_BG.predict(X_test)

true2 = y_test

print(r2_score(true1, predict1))

print(r2_score(true2, predict2))
```

```
BGG = pd.read_csv(r"C:\Users\tzq\Desktop\gshr\boost_input_predict.csv", index_col = 0)

Z = BGG.values

featnames=list(BGG.columns)

Z_scaled =preprocessing.scale(Z)

Z_new = model.transform(Z_scaled)

Z_new.shape
```

```
BG_predict = model_BG.predict(Z_new)

print("BG_predict", BG_predict)
```