1. 回归模型

from sklearn.ensemble import RandomForestRegressor

import numpy as np

import pandas as pd

data=pd.read\_excel(r' ', index\_col=0)

data=data.astype(float) #格式转换

array = data.values

X = array[:,0:8]

Y = array[:,8:9]

from sklearn.preprocessing import StandardScaler

transformer = StandardScaler().fit(X)

newX = transformer.transform(X)

X=newX

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

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X, Y, test\_size=0.3)

rfc = RandomForestRegressor(n\_estimators= 40 ,max\_features= 2 ,random\_state=0)

rfc = rfc.fit(Xtrain, Ytrain)

predictions=rfc.predict(Xtest)

errors=abs(predictions-Ytest)

import seaborn as sns

x=Ytest

y=predictions

score\_r = rfc.score(Xtest,Ytest)

print( "Random Forest:{}".format(score\_r) )

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import r2\_score

mse\_test1 = mean\_squared\_error(Ytest,predictions)

mae\_test1 = mean\_absolute\_error(Ytest,predictions)

rmse\_test1 = mse\_test1 \*\* 0.5

r2\_score1 = r2\_score(Ytest,predictions)

print(' The result of calling the function directly is as follows：')

print(' Mean-squared error:{}, Mean absolute error:{},\n Root-mean-square error:{}, Coefficient of determination:{}'.format(mse\_test1,mae\_test1,rmse\_test1,r2\_score1))

import matplotlib.pyplot as plt

sns.set(style='darkgrid', color\_codes=True)

sns.regplot(x=Ytest, y=predictions)

plt.rcParams['font.sans-serif']=['Times New Roman']

plt.tick\_params(labelsize=20)

plt.xlabel('Actual DP')

plt.ylabel('Predicted DP')

y=x

plt.plot(x,y,color='red',linewidth = 3)

plt.show()

1. PCCs

from sklearn.inspection import plot\_partial\_dependence

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor

import seaborn as sns

import numpy as np

import pandas as pd

import matplotlib as mpl

import scipy

from scipy.stats import pearsonr

vegetables = ["DP","DE"]

farmers = ["DC(%)","M(g/mL)","Ratio","C(ug/mL)","pH","T(℃)","t(min)"]

data = np.array([[-0.16,0.22,0.48,0.28,0.14,-0.11,0.35],

[-0.21,-0.42,-0.29,0.46,0.41,-0.16,0.25]]

)

#data.corr(method = "pearson")

plt.figure(figsize=(4,3), dpi= 400)

plt.rcParams['font.sans-serif']=['Times New Roman']

plt.xticks(np.arange(len(farmers)), labels=farmers, fontsize=10,

rotation=45, rotation\_mode="anchor", ha="right")

plt.yticks(np.arange(len(vegetables)), labels=vegetables, fontsize=10)

plt.title("PEG",fontsize=10)

for j in range(len(farmers)):

for i in range(len(vegetables)):

text = plt.text(j, i, data[i,j], ha="center", va="center", color="w")

plt.imshow(data)

plt.colorbar()

plt.tight\_layout()

plt.show()

1. Box plot

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib.font\_manager as font\_manager

from matplotlib.font\_manager import FontProperties

from sklearn.linear\_model import LinearRegression

data=pd.read\_excel(r' ', index\_col=0)

plt.rcParams['font.sans-serif']=['Times New Roman']

data.plot.box( subplots=True,layout=(2,5), sharex=False,sharey=False,fontsize=28)

plt.tight\_layout()

plt.grid(linestyle="--", alpha=0.3)

plt.show()

1. ICE

from sklearn.datasets import make\_hastie\_10\_2

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.inspection import plot\_partial\_dependence

from pdpbox import pdp, info\_plots

import matplotlib.font\_manager

from sklearn.ensemble import RandomForestRegressor

from sklearn.inspection import PartialDependenceDisplay

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import shap

shap.initjs()

from sklearn.datasets import make\_hastie\_10\_2

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

from pdpbox import pdp, get\_dataset, info\_plots

from time import time

from sklearn.preprocessing import StandardScaler

data=pd.read\_excel(r' ', index\_col=0)

tic = time()

cols\_to\_use=[ "DC (%)","M (g/mL)", "Ratio", "C (ug/mL)", "pH" ,"T (℃)", "t (min)", "P (bar)"]

coll=["DE (%)"]

X = data[cols\_to\_use]

y=data[coll]

model = RandomForestRegressor(random\_state=42).fit(X, y)

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

transformer = StandardScaler().fit(X)

newX = transformer.transform(X)

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X, y, test\_size=0.3)

plt.rcParams['font.sans-serif']=['Times New Roman']

gbm = GradientBoostingRegressor()

gbm.fit(Xtrain, Ytrain)

shap.plots.partial\_dependence(

"t (min)", model.predict, X, ice=True,

model\_expected\_value=True, feature\_expected\_value=True

)

1. Shap

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import shap

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPRegressor

from sklearn.pipeline import make\_pipeline

from sklearn.datasets import load\_diabetes

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

import xgboost as xgb

import matplotlib.pyplot as plt; plt.style.use('seaborn')

shap.initjs()

data=pd.read\_excel(r' ', index\_col=0)

cols = ['C (ug/mL)','DC (%)','M (g/mL)','pH','T (℃)','P (bar)','t (min)']

coll=['DE (%)']

X = data[cols]

y=data[coll]

from sklearn.preprocessing import StandardScaler

transformer = StandardScaler().fit(X) # 实例化 # fit，本质是生成均值和方差

newX = transformer.transform(X)

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

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(X, y, test\_size=0.3)

plt.rcParams['font.sans-serif']=['Times New Roman']

plt.tick\_params(labelsize=15)

plt.xlabel('shap',size=15)

model = xgb.XGBRegressor(max\_depth=4, learning\_rate=0.05, n\_estimators=150)

model.fit(data[cols],data['DE (%)'].values)

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(data[cols])

print(shap\_values.shape)

y\_base = explainer.expected\_value

print(y\_base)

data['pred'] = model.predict(data[cols])

print(data['pred'].mean())

shap\_interaction\_values = explainer.shap\_interaction\_values(X)

shap.summary\_plot(shap\_interaction\_values,X,max\_display=20,plot\_type="compact\_dot")