The codes for the machine learning techniques used in this work

1. Prediction model

```
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)
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()
```

2. 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(°C)","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]]
             )
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()
```

3. 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()
```

4. 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 (°C)", "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
)
```

5. 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 (°C)','P (bar)','t (min)']
coll=['DE (%)']
X = data[cols]
y=data[coll]
from sklearn.preprocessing import StandardScaler
transformer = StandardScaler().fit(X) 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")
```