import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import mean\_squared\_error, r2\_score

from hyperopt import fmin, tpe, hp, Trials

import matplotlib.pyplot as plt

# 设置图片清晰度

plt.rcParams['figure.dpi'] = 300

# 设置 matplotlib 支持中文

plt.rcParams['font.sans-serif'] = ['SimHei'] # 使用黑体字体

# 解决负号显示问题

plt.rcParams['axes.unicode\_minus'] = False

# 定义超参数搜索空间

parameter\_space\_rf = {

'n\_estimators': hp.choice('n\_estimators', range(10, 201)), # 树的数量

'max\_depth': hp.choice('max\_depth', range(1, 51)), # 树的最大深度

'min\_samples\_split': hp.quniform('min\_samples\_split', 2, 20, 1), # 确保是整数

}

# 目标函数

def objective(params):

params['min\_samples\_split'] = int(params['min\_samples\_split']) # 确保整数

model = RandomForestRegressor(

n\_estimators=params['n\_estimators'],

max\_depth=params['max\_depth'],

min\_samples\_split=params['min\_samples\_split'],

random\_state=42

)

scores = cross\_val\_score(model, X\_train, y\_train, cv=5, scoring='neg\_mean\_squared\_error')

return -np.mean(scores) # 目标是最小化均方误差

# 读取数据

import pandas as pd

# 读取数据

data = pd.read\_excel('训练集(1).xlsx', header=1)

columns = data.columns[1:]

X\_train = data.loc[:, columns]

y\_train = data.iloc[:, 0]

# 读取数据

data = pd.read\_excel('测试集(1).xlsx', header=1)

columns = data.columns[1:]

X\_test = data.loc[:, columns]

y\_test = data.iloc[:, 0]

# 运行贝叶斯优化

trials = Trials()

best\_params = fmin(

fn=objective,

space=parameter\_space\_rf,

algo=tpe.suggest,

max\_evals=100,

trials=trials

)

# 解析最优参数

n\_estimators\_best = best\_params['n\_estimators']

max\_depth\_best = best\_params['max\_depth']

min\_samples\_split\_best = int(best\_params['min\_samples\_split']) # 转换为整数

print('最优的 n\_estimators:', n\_estimators\_best)

print('最优的 max\_depth:', max\_depth\_best)

print('最优的 min\_samples\_split:', min\_samples\_split\_best)

# 训练最终模型

best\_model = RandomForestRegressor(

n\_estimators=n\_estimators\_best,

max\_depth=max\_depth\_best,

min\_samples\_split=min\_samples\_split\_best,

random\_state=42

)

best\_model.fit(X\_train, y\_train)

y\_pred = best\_model.predict(X\_test)

# 计算评估指标

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'优化后的随机森林回归均方误差 (MSE): {mse:.4f}')

print(f'优化后的随机森林回归 R²: {r2:.4f}')

# 绘制优化过程

losses = [t['result']['loss'] for t in trials.trials]

min\_loss = min(losses)

min\_loss\_index = losses.index(min\_loss)

plt.plot(range(len(losses)), losses, marker='o', linestyle='-', color='b', label='Loss (MSE)')

plt.scatter(min\_loss\_index, min\_loss, marker='\*', color='r', s=200, label='Optimal Loss',zorder=2)

plt.grid(True, linestyle="--", alpha=0.6)

plt.xlabel('Iteration')

plt.ylabel('Loss (MSE)')

plt.title('Bayesian Optimization of Random Forest Regression')

plt.legend()

plt.show()

X\_test=pd.read\_excel('X\_test.xlsx')

y\_test=pd.read\_excel('y\_test.xlsx')

X\_train=pd.read\_excel('X\_train.xlsx')

y\_train=pd.read\_excel('y\_train.xlsx')

# 导入joblib库

from joblib import dump, load

# 保存模型

dump(best\_model, 'best\_random\_forest\_model.joblib')

from joblib import dump, load

best\_model = load('best\_random\_forest\_model.joblib')

import numpy as np

from sklearn.metrics import (

mean\_squared\_error,

mean\_absolute\_error,

r2\_score,

explained\_variance\_score,

median\_absolute\_error,

max\_error

)

def regression\_metrics(y\_true, y\_pred):

"""

计算回归模型的常用评价指标。

参数:

y\_true: 真实值 (numpy 数组或列表)

y\_pred: 预测值 (numpy 数组或列表)

返回:

metrics\_dict: 包含所有评价指标的字典

"""

# 计算各项指标

mse = mean\_squared\_error(y\_true, y\_pred)

rmse = np.sqrt(mse)

mae = mean\_absolute\_error(y\_true, y\_pred)

r2 = r2\_score(y\_true, y\_pred)

explained\_variance = explained\_variance\_score(y\_true, y\_pred)

medae = median\_absolute\_error(y\_true, y\_pred)

max\_err = max\_error(y\_true, y\_pred)

# 计算 MAPE

# 为避免除零错误，将真实值为零的位置替换为一个极小值（例如 1e-8）

y\_true = np.asarray(y\_true)

y\_true = np.where(y\_true == 0, 1e-8, y\_true)

mape = np.mean(np.abs((y\_true - y\_pred) / y\_true)) \* 100

metrics\_dict = {

'MSE': mse,

'RMSE': rmse,

'MAE': mae,

'R2': r2,

'Explained Variance': explained\_variance,

'Median Absolute Error': medae,

'Max Error': max\_err,

'MAPE': mape

}

return metrics\_dict

import numpy as np

from sklearn.model\_selection import KFold

from sklearn.metrics import (

mean\_squared\_error,

mean\_absolute\_error,

r2\_score,

explained\_variance\_score,

median\_absolute\_error,

max\_error

)

def cv(best\_model, X, y):

"""