import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from skopt import BayesSearchCV  
from sklearn.model\_selection import StratifiedKFold  
from keras.models import Sequential  
from keras.layers import LSTM, Dense  
from keras.wrappers.scikit\_learn import KerasRegressor  
from sklearn.metrics import mean\_squared\_error  
from sklearn.preprocessing import StandardScaler  
  
# 加载数据  
bfgfasheng\_data = pd.read\_csv(r'E:\新建文件夹\代码\lstm\data\BFG\BFGFSL.csv')  
fengya\_data = pd.read\_csv(r'E:\新建文件夹\代码\lstm\data\BFG\FengYa.csv')  
fl\_data = pd.read\_csv(r'E:\新建文件夹\代码\lstm\data\BFG\FL.csv')  
fy\_data = pd.read\_csv(r'E:\新建文件夹\代码\lstm\data\BFG\FY.csv')  
xiaohao\_data = pd.read\_csv(r'E:\新建文件夹\代码\lstm\data\BFG\热风炉消耗BFG.csv')  
# 提取特征列  
bfgfasheng = bfgfasheng\_data["value"][1:5000]  
fengya = fengya\_data["value"][1:5000]  
fl = fl\_data["value"][1:5000]  
fy = fy\_data["value"][1:5000]  
xiaohao = xiaohao\_data["value"][1:5000]  
time = xiaohao\_data["datetime"]  
# 合并数据  
merged\_data = pd.concat([bfgfasheng, fengya, fl, fy], axis=1)  
  
# 提取特征和目标变量  
X = merged\_data.values  
y = xiaohao.values  
# 定义时间戳列的索引  
timestamp\_column\_index = 0 # 请根据你的数据调整这个索引  
  
# 转换时间戳列为 Unix 时间戳  
X[:, timestamp\_column\_index] = pd.to\_datetime(X[:, timestamp\_column\_index]).astype('int64') // 10\*\*9  
  
scaler = StandardScaler()  
X = scaler.fit\_transform(X)  
y = scaler.fit\_transform(y)  
  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 其余代码保持不变  
  
# 其余代码保持不变  
  
# 定义LSTM模型  
def create\_lstm\_model(units, dropout\_rate, input\_shape):  
 model = Sequential()  
 model.add(LSTM(units=units, input\_shape=input\_shape))  
 model.add(Dense(1))  
 model.compile(loss='mean\_squared\_error', optimizer='adam')  
 return model  
  
# 定义超参数搜索空间  
param\_space = {'units': (50, 200),  
 'dropout\_rate': (0.0, 0.5)}  
  
# 初始化贝叶斯优化器  
lstm\_model = KerasRegressor(build\_fn=create\_lstm\_model, epochs=10, batch\_size=32, verbose=0)  
cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)  
bayes\_search = BayesSearchCV(lstm\_model, param\_space, n\_iter=10, cv=cv, n\_jobs=-1)  
  
# 运行优化过程  
bayes\_search.fit(X\_train, y\_train)  
  
# 输出最优参数  
print("Best parameters found: ", bayes\_search.best\_params\_)  
  
# 应用最优参数并重新训练模型  
best\_lstm\_model = create\_lstm\_model(units=bayes\_search.best\_params\_['units'],  
 dropout\_rate=bayes\_search.best\_params\_['dropout\_rate'],  
 input\_shape=(X\_train.shape[1], X\_train.shape[2]))  
best\_lstm\_model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_data=(X\_test, y\_test))  
  
# 在测试集上评估模型  
y\_pred = best\_lstm\_model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print("Mean Squared Error on Test Set: ", mse)