# 基于 XGBoost 预测模型和帝国竞争算法的矿石加工质量问题


## 摘要

本文研究矿石加工以及产品质量相关问题，建立 XGBoost 回归模型、广义回归神经网络模型、XGBoost 分类模型对各项指标进行预测。最终建立温度调控规划模型分析指定合格率条件下的系统设定温度，并从宏观与微观的角度对结果进行分析。

**针对问题一**，本文建立 XGBoost 回归模型对产品质量指标进行预测，并利用置信区间确定可能性最大产品指标。首先，本文利用**小波降噪**对数据进行处理，通过 **DBSCAN 算法**对温度变化情况进行聚类，并针对温度数据的不同情况进行缺失值填补。由此建立 **XGBoost 回归模型**和**置信区间**对产品质量指标与可能性分布进行预测。其中，通过**十折交叉验证**划分数据集，利用**贝叶斯调参**寻找模型的最优参数。最终，本文利用**残差正态性检验**和**置信区间检验**对模型的预测效果进行检验。得到第一种温度组合下的产品质量指标 ABCD 为：**80.56**、**23.52**、**11.47**、**18.38**；第二种温度组合下的产品质量指标 ABCD 为：**79.10**、**24.30**、**11.21**、**17.19**.

**针对问题二**，本文建立**广义回归神经网络 (GRNN) 模型**对系统设定温度进行双输出预测。首先，预测数据集即为问题一中处理得到的原矿参数、系统设定温度以及产品质量指标数据。之后建立 **GRNN 模型**进行预测，在模型求解过程中，利用**交叉验证法**确定最优参数，从而达到最佳预测效果。求解得到**双输出预测结果**，即系统 I 温度与系统 II 温度。最终，本文对预测结果进行**残差正态性检验**。得到第一类产品质量的系统 I、II 温度为 **858.31**、**719.92**，第二类质量的系统 I、II 温度为 **890.09**、**726.28**.

**针对问题三**，本文建立 XGBoost 分类模型对产品合格率进行预测。首先，本文对原矿参数和过程数据的缺失值进行填补。之后利用 **SMOTE** 对数据进行**过采样**，使数据量表现均衡。接着本文基于 **ADF 单位根检验**与**格兰杰因果检验**，选择 **XGBoost 分类模型**对产品合格率进行预测。最终，利用 **Friedman 检验**对 XGBoost 分类模型的效果进行评估。由此得到第一种温度组合下的产品合格率为：**38.36**%；第二种温度组合下的产品合格率为：**15.11**%。

**针对问题四**，本文建立温度调控规划模型求解指定合格率下的系统设定温度，并进行宏微观分析。首先，本文以最小温度变化量为目标建立**温度调控规划模型**，结合问题三的 XGBoost 分类模型以及题干条件对规划模型进行约束。接着利用**帝国竞争算法**对规划模型进行求解。之后，本文从**宏观**角度对温度调控规划模型进行**敏感性分析**，从**微观**角度对模型进行**准确性分析**。求解得到 4 月 10 日 80% 的合格率可以达到，系统 I、II 设定温度分别为 **850.12**、**513.47**，而 4 月 11 日 99% 的合格率无法达到。

**本文的优势在于：** 1. 利用 XGBoost 的置信区间表示预测值的可能性大小，使结果更加全面可靠；2. 从宏微观角度对规划模型的调控结果进行探讨，使分析结果更加完善。



**关键字**： 小波降噪　XGBoost　GRNN　帝国竞争算法　宏微观分析

# 目录

# 一、 问题重述

## 1.1 问题背景

随着全球碳达峰、碳中和“双碳”目标共识的形成，全球能源系统从燃料密集型逐渐向材料密集型转变，这将导致新能源产业对矿产资源的需求不断增加[8]。由此，矿产资源将在全球能源转型过程中发挥至关重要的作用。

矿石加工质量的提高，可以节约不可再生的矿物资源以及加工所需能源，从而助力“双碳”目标的实现。矿石加工是一个复杂的过程，在加工过程中，电压、水压、温度作为影响矿石加工的重要因素，直接影响着矿石产品的质量。矿石加工过程需经过系统 I、系统 II 两个环节，其中生产技术人员通过调温指令进行温度调节，从而改变产品质量。

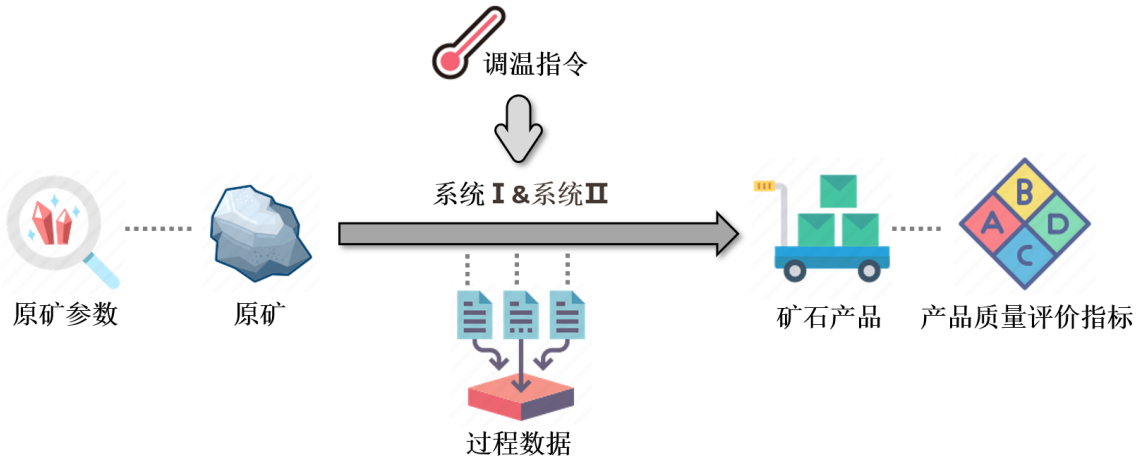


**图 1　矿石加工背景图**

## 1.2 问题提出

随着全球碳达峰、碳中和“双碳”目标共识的形成，矿产资源将在全球能源转型过程中发挥至关重要的作用。矿石加工质量的提高，可以节约不可再生的矿物资源以及加工所需能源，从而助力“双碳”目标的实现。针对矿石加工质量的相关信息，我们对以下几个问题进行研究：

问题一：结合生产加工数据建立数学模型，利用系统温度对产品质量进行预测。在相同系统温度所得产品质量差别较大的情况下，对可能性最大的产品指标进行预测。

问题二：结合问题一，利用原矿参数和产品目标质量数据，建立模型对系统设定温度进行估计。对于同一组产品质量对应的多种调温方法，给出可能性最大的系统设定温度。

问题三：结合过程数据以及生产加工数据，利用指定系统设定温度建立数学模型，对矿石产品合格率进行预测，并建立数学模型对给出的合格率的准确性进行评价。

问题四：分析在指定合格率的条件下设定系统温度的方法，并完成以下任务：（1）适当的敏感性分析；（2）对结果准确性的分析；（3）判断能否达到合格率要求，若可以达到，给出系统设定温度。

## 二、 模型假设

（1）数据真实可靠；
（2）矿石必须在两小时非调温时间段内进行加工；
（3）系统温度与调温指令设定的温度相同；
（4）产品隔日重新开始加工。

## 三、 符号说明

| 符号 | 含义 | 说明 |
|---|---|---|
| $x_{1,i},\cdots,x_{6,i}$ | 第 $i$ 组数据的原矿参数与系统温度 | XGBoost 回归模型对应自变量指标 |
| $y_{1,i},\cdots,y_{4,i}$ | 第 $i$ 组数据的四项产品质量指标 | XGBoost 回归模型预测产品质量指标 |
| $X_1,\cdots,X_8$ | 四项产品质量指标与原矿参数 | 广义回归神经网络输入层 |
| $\sigma$ | 广义回归神经网络唯一超参数 | |
| $t_{i,1},\cdots,t_{i,10}$ | 第 $i$ 组数据自变量指标 | XGBoost 分类模型对应自变量指标 |
| $u_i$ | XGBoost 分类模型对应产品合格率 | |

注：未列出的以及重复的符号均以首次出现处为准

## 四、 问题一的模型建立与求解

### 4.1 问题一的描述与分析

问题一要求结合生产加工数据建立数学模型，利用系统温度对产品质量进行预测。首先，本文利用**小波降噪**对数据进行处理，之后利用 **DBSCAN 算法**对温度变化情况进行聚类，从而对是否处于调温过程进行分析。接着针对温度数据的不同情况进行缺失值填补。本文建立 **XGBoost 回归模型**对产品质量指标进行预测。之后，本文利用 XGBoost 的**置信区间**得到产品指标的可能性分布。其中，通过**十折交叉验证**划分数据集，利用**贝叶斯调参**寻找模型的最优参数。最终，本文利用残差正态性检验和置信区间检验对模型的预测效果进行检验。

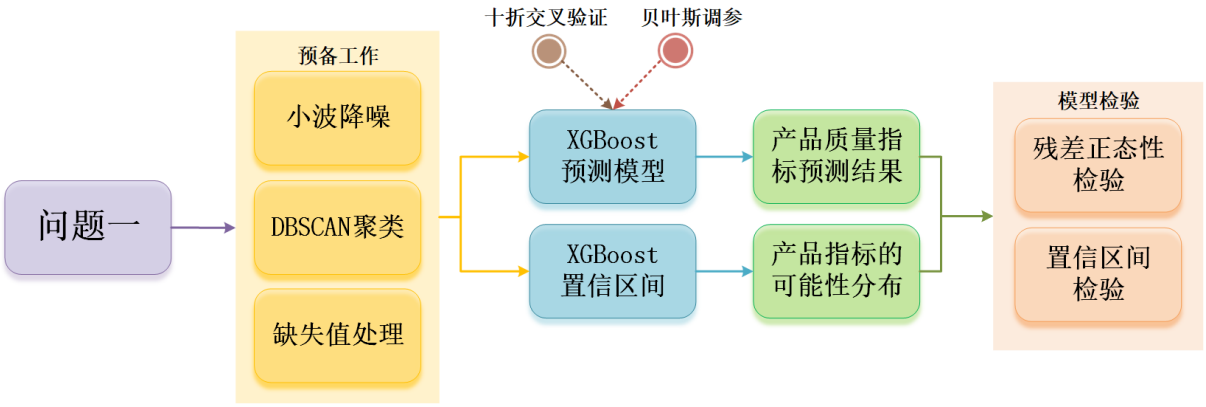


**图 2　问题一思路图**

### 4.2 预备工作

本文基于附件 1 中数据的波动性特征，对温度数据进行小波降噪处理。之后，为了更好地区分温度数据是否处于调温过程，本文利用 DBSCAN 算法对相邻数据的温度变化进行聚类。最终结合聚类结果，针对调温过程与非调温过程的缺失值分别进行填补。

#### 4.2.1 数据的小波降噪

由于数据存在一定的波动变化，本文利用小波降噪对数据进行降噪处理，得到高质量的数据源。首先，对原始温度数据进行小波分解，得到各高频数据以及低频数据。之后，对高频数据进行阈值处理，用处理后的各分量进行小波重构，从而得到去噪后的数据。其中，共有 3 种阈值处理方法，数据的小波降噪需从 3 种方法中选取效果最好的方法进行去噪。系统 I 和系统 II 的不同阈值处理方法对应效果如下表:

**表 1　不同阈值处理方法效果对比表**

| | 系统 I SNR | 系统 I RMSE | 系统 II SNR | 系统 II RMSE |
|---|---|---|---|---|
| 硬阈值 | 66.8247 | 0.54697 | 64.5079 | 0.48341 |
| 软阈值 | 62.2941 | 0.9215 | 59.6461 | 0.84606 |
| 固定阈值 | 65.0705 | 0.66938 | 65.3491 | 0.43879 |

上表展示了不同阈值处理方法对应的信噪比（SNR）以及均方根误差（RMSE），信噪比越大，均方根误差越小，对应的去噪效果越好。由此可得，系统 I 的最佳阈值处理方法为硬阈值，系统 II 的最佳阈值处理方法为固定阈值。

为了方便展示，本文选取系统 I 的前 103 条温度数据的小波降噪效果展示如下图3:

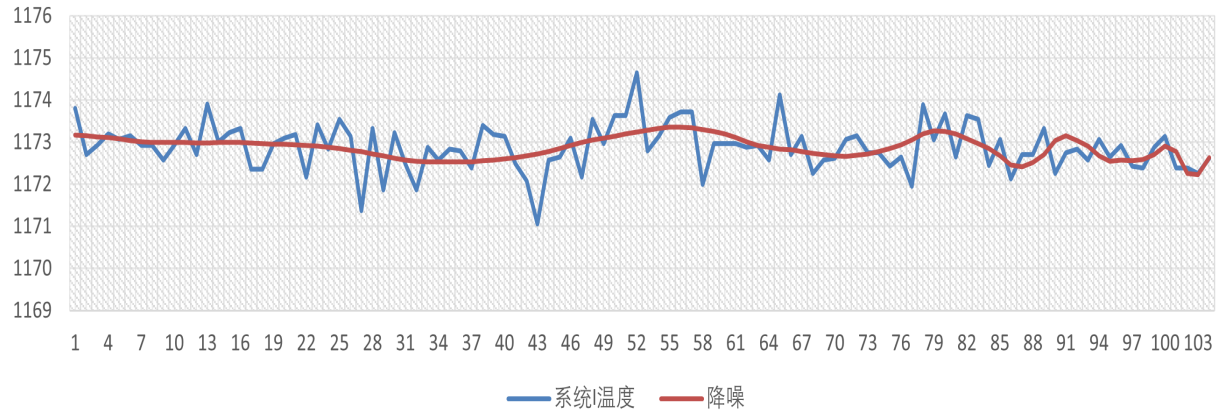


**图 3　温度数据小波降噪效果图**

由图可知，经过小波降噪处理后的温度数据表现平稳，降噪效果较好。小波降噪所得的数据可以更好地用于后续的预测工作。

### 4.2.2 温度变化的 DBSCAN 聚类

对数据进行降噪处理后，对 14229 组数据的相邻温度数据作差并取绝对值，得到 14228 项温度变化值。DBSCAN 聚类[6] 流程如下图4：

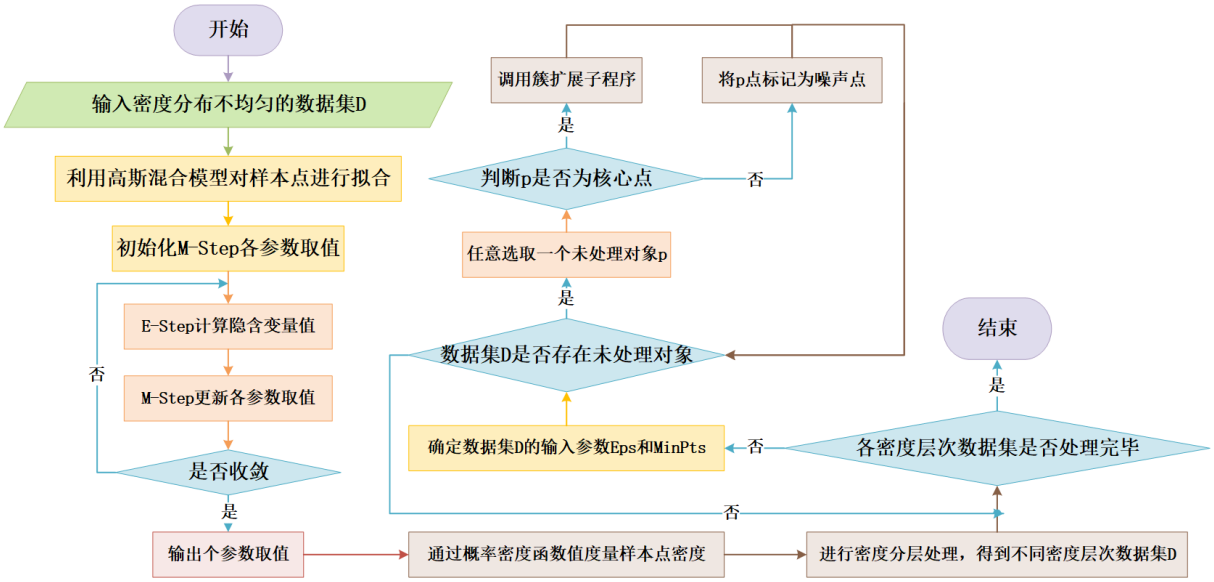


**图 4　DBSCAN 聚类算法流程图**

本文对系统 I 和系统 II 对应温度变化值进行二维 DBSCAN 聚类如下图5。图中红色星号表示处于非调温过程的温度变化，共有 14172 项，此时温度的波动较小。图中黑色圈则表示处于调温过程的温度变化，此时温度波动较大，进行了温度调节。

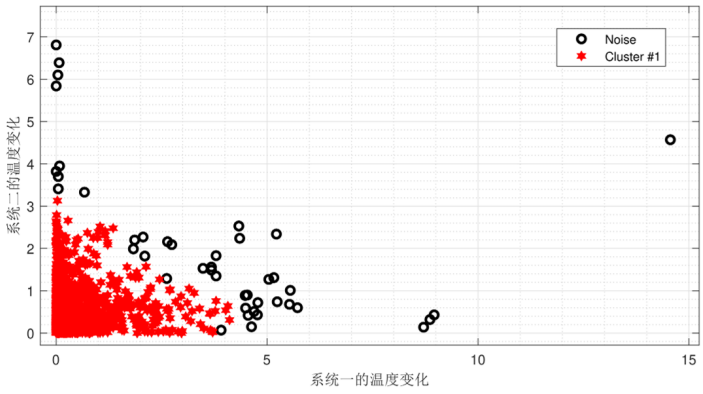


**图 5　温度变化聚类效果图**

### 4.2.3 缺失值处理

对附件 1 中的温度数据进行分析发现，部分数据存在缺失情况，本文对不同类型的缺失值分别进行处理：

▷ 调温过程数据：结合4.2.2中的聚类结果，若缺失值属于调温过程，则利用该调温过程的温度数据对缺失值进行插值填补。

▷ 非调温过程数据：结合4.2.2中的聚类结果，若缺失值属于非调温过程，则对该平稳加工过程的温度数据取均值，对缺失值进行填补。

▷ 其他型数据：此外，数据中还存在一处无法判断的情况。附件 1 中 2022 年 1 月 20 日 6:25-9:03 时间段的数据存在缺失，且温度存在较大变化，无法判断该段数据中是否存在两小时的非调温过程。因此，本文对该数据进行删除处理。

## 4.3 模型建立

### 4.3.1 基于 XGBoost 回归模型预测产品质量

本文通过 Python 软件利用 XGBoost 算法对模型进行训练和预测。因为隔日产品需要重新加工，在确定预测数据集时，首先以天为单位结合4.2所得结果确定非调温过程的温度数据，对每段产品加工过程对应温度求均值得到系统设定温度，并由此得到对应时间的原矿参数以及产品质量指标数据。数据集的具体确定方法如下图6：

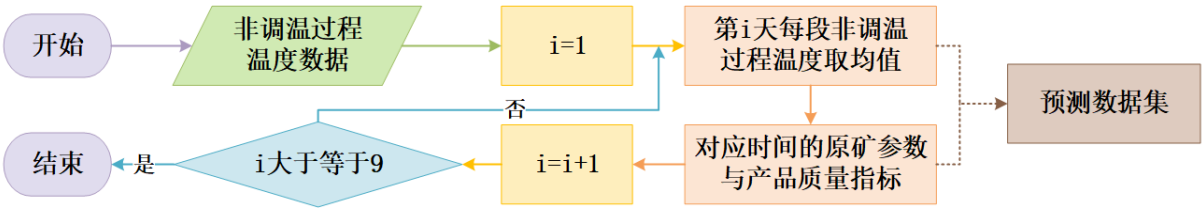


**图 6　预测数据集选择流程图**

设四项原矿参数与两个系统设定温度数据分别为 $x_{1,i},\cdots,x_{6,i}$，四项产品质量指标数据分别为 $y_{1,i},\cdots,y_{4,i}$。则本文给定数据集为 $D=(x_{1,i},\cdots,x_{6,i},y_{1,i}\cdots,y_{4,i})$，XGBoost 进行 additive training，学习 $K$ 棵树，采用以下函数对样本进行预测[2]，对四项产品质量指标分别进行下述过程：

$$\tilde{y_{t,i}}=\sum_{k=1}^{K}f_k(x_{j,i}),\quad f_k\in\Gamma;t=1,2,\cdots,4;j=1,2,\cdots,6. \tag{1}$$

其中 $f(x)$ 是回归树，$\Gamma$ 是假设空间：

$$\Gamma=\{f(x)=w_{q(x)}\}(q:R^m\to T,w\in R^T). \tag{2}$$

其中，$q(x)$ 表示样本 $x$ 分到了某个叶子节点上，$w$ 是叶子节点的分数，所以 $w_{q(x)}$ 表示回归树对产品质量指标样本的预测值。

### 4.3.2 基于 XGBoost 的置信区间确定可能性最大产品质量指标

此外，本文利用 XGBoost 的置信区间确定相同温度下产品质量指标的可能性分布。由于 XGBoost 具有自定义目标函数的特点，因此本文确定分位数回归目标，从而将置信区间与 XGBoost 预测相结合。本文选择 90% 置信区间，将 $\alpha=0.95$ 用作上限，$\alpha=0.05$ 用作下限分别对区间的上限和下限进行训练。由此，可以实现为 XGBoost 预测过程预测可靠的置信区间，从而得到可能性最大的产品质量指标。

## 4.4 模型求解

- **10-fold 交叉验证**

在求解上述 XGBoost 回归模型的过程中，本文利用 10-fold 交叉验证对数据集进行处理，10-fold 交叉验证示意图如下图7：

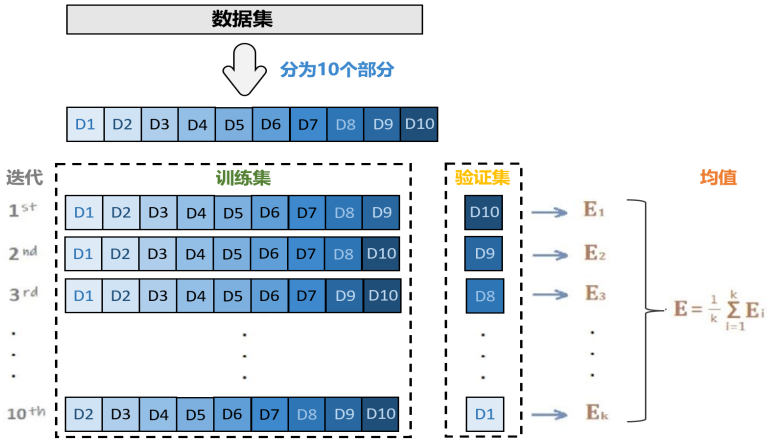


**图 7　10-fold 交叉验证示意图**

由上图可知，10-fold 交叉验证将数据集划分为 10 等份，每次取其中一组数据作为测试集，其余数据作为训练集进行求解[11]。

- **贝叶斯调参**

本文以 10-fold 交叉验证集方差平均值最小为目标，利用贝叶斯调参对 XGBoost 回归模型进行最优参数寻找。贝叶斯调参流程的伪代码如下[7]：

**Algorithm 1** Sequential Model-Based Optimization

**Input:** $f, X, S, M$

D $\leftarrow INITSAMPLES(f, X)$

**for do**$i \leftarrow |D|$**to** $T$

  $p(y|x, D) \leftarrow FITMODEL(M, D)$; $x_i \leftarrow arg\max_{x \in X} S(x, p(y|x, D))$

  $y_i \leftarrow f(x_i)$ ▷ Expensive step; $D \leftarrow D \cup (x_i, y_i)$

**end for**

### 4.5 求解结果与分析

本文利用 XGBoost 回归模型对产品质量指标 A、B、C、D 进行预测的拟合系数 $R^2$ 如下表：

**表 2　XGBoost 预测拟合系数表**

| | 指标 A | 指标 B | 指标 C | 指标 D |
|---|---|---|---|---|
| $R^2$ | 0.50235 | 0.325529 | 0.711616 | 0.531541 |

由上表可知，产品质量指标 A、C、D 的拟合效果较好，可解释性较高，其中模型对于指标 C 的预测能力最强。而预测产品质量指标 B 对应的 $R^2$ 并不理想，本文分析这是由于除温度外其他因素对指标 B 的影响程度较大，从而导致仅利用温度进行的预测不准确。

基于 XGBoost 回归模型对表 1 中对应温度下的产品质量指标进行补全。其中，对于相同温度对应不同产品质量指标的情况，基于置信区间结果选择可能性最大的产品质量指标。若温度相同时，该组产品质量指标处于 90% 的置信区间内，则说明该情况出现的可能性较大。经过比较得到每个温度组合下可能性最大的产品质量指标。由此可得两种系统温度组合情况下各项产品质量指标及其对应置信区间展示如下表：

**表 3　两组温度下产品质量指标及置信区间表**

| | 第 1 组 | 置信区间 | 第 2 组 | 置信区间 |
|---|---|---|---|---|
| 指标 A | 80.56056 | [79.999,80.766] | 79.10446 | [78.823,79.500] |
| 指标 B | 23.52265 | [22.711,23.970] | 24.30242 | [23.905,24.413] |
| 指标 C | 11.46769 | [11.263,11.841] | 11.21381 | [10.657,11.909] |
| 指标 D | 18.38207 | [17.344,19.373] | 17.099 | [16.800,17.926] |

由上表可知，系统 I 设定温度不同系统 II 设定温度相同时，各项产品质量指标的预测结果表现出差异性。且两种情况下的各项产品质量指标数据均处于 90% 置信区间内，印证了本文预测结果的可靠性。观察上表各项指标置信区间结果发现，90% 置信区间长度仍较小，说明产品质量指标数据的分布较为集中。该结果也说明了处于置信区间内的产品质量指标数据能较大程度地反映该温度组合下的该项指标数据，出现可能性较大。

### 4.6 模型检验

- **残差正态性检验**

为了检验 XGBoost 回归模型预测的准确性，本文对残差进行正态性检验如下图8：

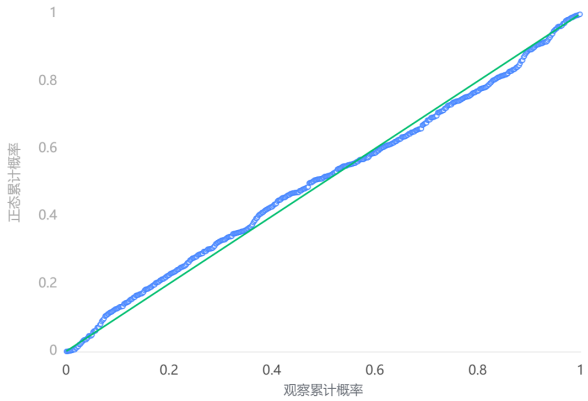


**图 8　XGBoost 回归模型残差检验图**

观察上图可知，XGBoost 回归模型预测所得残差通过了正态性检验。由此可知 XGBoost 回归模型残差中不存在可以用到预测中的遗留信息，预测的残差为白噪声。该结果说明本文 XGBoost 回归模型预测结果的准确性与可靠性。

- **置信区间检验**

本文对预测所得不同温度组合对应的各项产品质量指标 ABCD 进行置信区间检验，判断各指标数据是否处于 90% 置信区间内，从而对结果准确性进行检验。以 1 月 23 日第一组系统温度对应产品质量指标的置信区间检验结果，将置信区间的概率分布函数视为正态函数，展示如下图9：

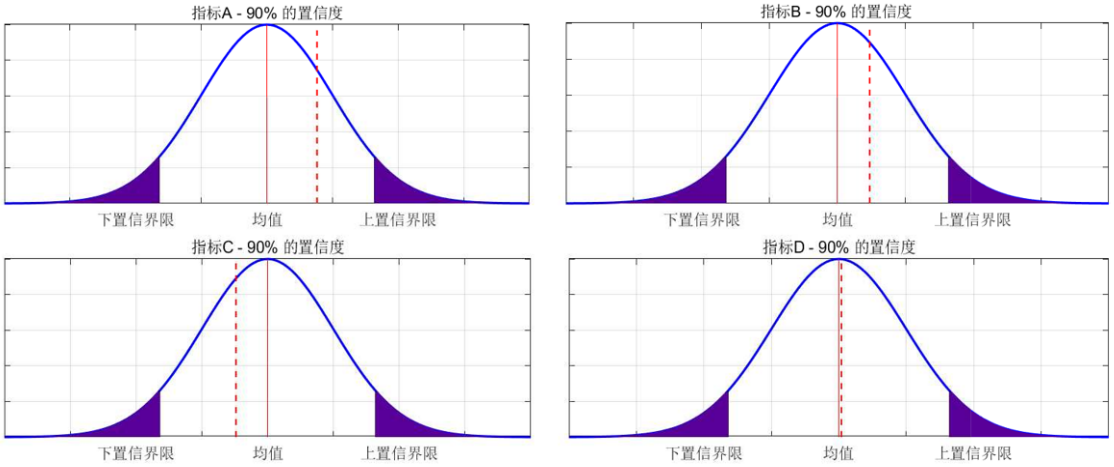


**图 9　产品指标 ABCD 置信区间检验结果图**

由上图可知，四项产品质量指标 A、B、C、D 均处于 90% 的置信区间，且相应的概率值较大，该结果说明预测所得四项产品质量指标为该组系统温度下可能性较大的结果组合，验证了预测结果的准确性与可靠性。

# 五、 问题二的模型建立与求解

## 5.1 问题二的描述与分析

问题二要求利用原矿参数和产品目标质量指标，对系统设定温度进行估计。问题二与问题一的不同之处在于该问题需要多输出结果，因此本文建立**广义回归神经网络**对系统设定温度进行预测。预测数据集即为问题一中处理得到的原矿参数、系统设定温度以及产品质量指标数据。在模型求解过程中，利用**交叉验证法**确定最优参数，从而达到最佳预测效果。最终，本文对预测结果进行残差正态性检验评估模型的预测效果。

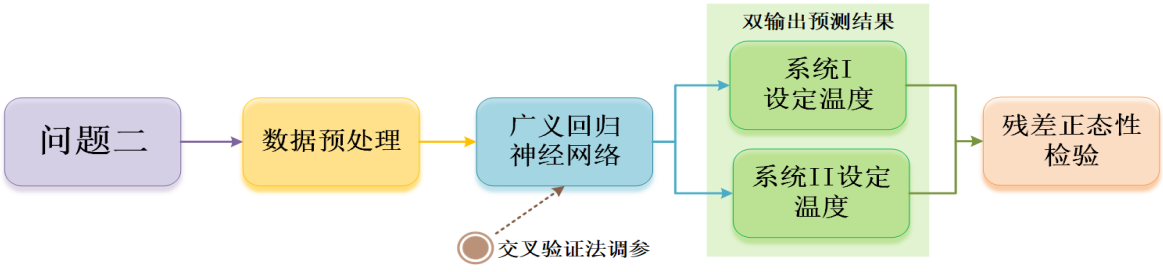


**图 10　问题二思路图**

### 5.2 用于系统温度预测的广义回归神经网络模型

由于对系统设定温度的预测为多输出结果，即输出系统 I 和系统 II 的对应温度，因此本文选择广义回归神经网络对系统设定温度进行预测[10]。广义回归神经网络 (Generalized Regression Neural Network,GRNN) 是径向基网络的变形形式。GRNN 的基础时非线性回归，后验条件为样本数据，并进行 Parzen 非参数估计，根据最大概率原则计算网络输出[1]。GRNN 的网络结构与径向基神经网络类似，是一个四层网络结构，如下图11所示：

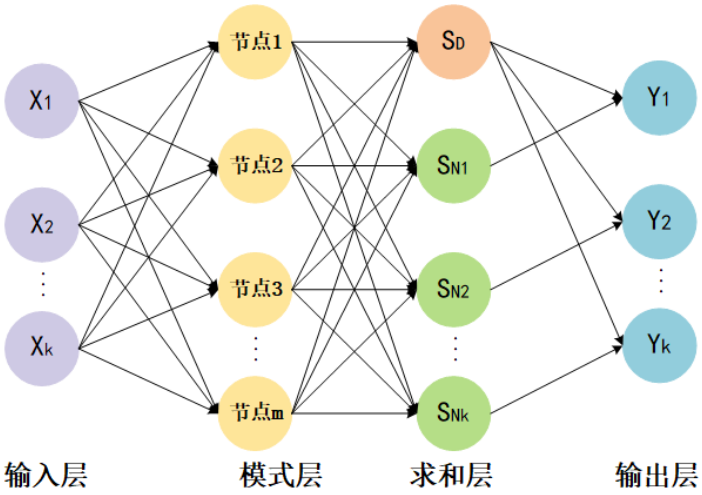


**图 11　GRNN 网络结构图**

**输入层：**记产品质量指标 A、B、C、D 分别为 $X_1$、$X_2$、$X_3$、$X_4$，四项原矿参数为 $X_5$、$X_6$、$X_7$、$X_8$。各指标数据的选择与处理方式与问题一相同。

**模式层：**模式层神经元数目即为学习样本数量 5，将权重定为 1，模式层的神经传递函数如下：

$$p_i = \exp[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}] \quad i = 1, 2, \cdots, 8. \tag{3}$$

其中，$X$ 为网络输入变量，$X_i$ 为第 $i$ 个神经元对应的学习样本，$\sigma$ 是模型唯一的超参数[1]。

**求和层：**对两种类型的神经元求和，一种为模式层输出的总和，即图中的 $S_D$ 部分，满足下式：

$$S_D = \sum_{i=1}^{n} \exp[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}]. \tag{4}$$

另一种为所有模式层输出的加权总和，传递函数为：

$$S_{Nj} = \sum_{i=1}^{n} y_{ij} \exp[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}], \quad j = 1, 2, \cdots, k. \tag{5}$$

其中，$y_{ij}$ 为神经元间的连接权重，即第 $i$ 个输出样本 $Y_i$ 的第 $j$ 个元素。

**输出层：**计算系统设定温度：

$$y_j = \frac{S_{Nj}}{S_D}, \quad j = 1, 2, \cdots, k. \tag{6}$$

### 5.3 基于交叉验证法调参的模型求解

上述 GRNN 模型中，平滑参数 $\sigma$ 是模型唯一的超参数。若 $\sigma$ 较高会导致数据过于平滑而较低的 $\sigma$ 则会导致过拟合。由此，本文利用交叉验证法对 GRNN 模型确定最优参数，以达到最好的预测效果。交叉验证法将原始数据集分为 $K$ 份，每次取其中一份为测试集，剩余 $K-1$ 份为训练集，从而得到 $K$ 个模型。

本文利用 $k$ 个模型精度的平均值 $\bar{q}$ 对模拟效果进行衡量：

$$\bar{q} = \frac{\sum_{i=1}^{K} \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}}{K}. \tag{7}$$

交叉验证法对 GRNN 模型调参流程如下图12：

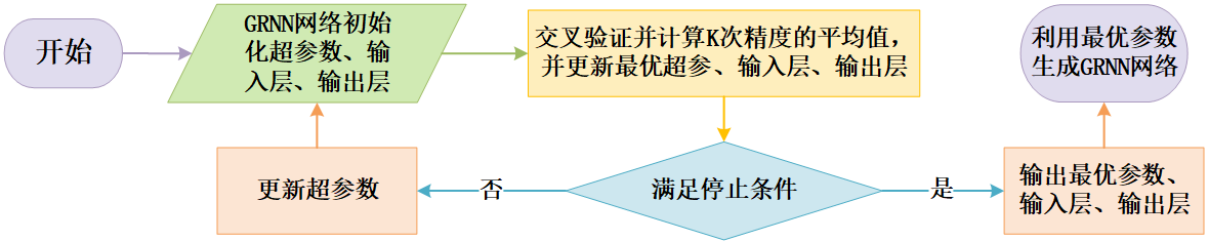


**图 12　交叉验证发超参数调优流程图**

本文将原矿参数以及产品质量指标数据用作训练和验证。利用 10-fold 交叉验证确定最优超参 $\sigma$，本文采用循环训练的方式，[0.1,0.5] 被认为是最优超参 $\sigma$ 的初始搜索区间，下表展示了 10 种交叉验证结果中最优超参 $\sigma$ 下的最大 $R^2$。

**表 4　10 种交叉验证结果中最优超参 $\sigma$ 下的最大 $R^2$**

| 次数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma$ | 0.4 | 0.5 | 0.5 | 0.1 | 0.1 | 0.4 | 0.4 | 0.4 | 0.2 | 0.1 |
| $R^2$ | 0.42 | 0.49 | 0.33 | 0.95 | 0.89 | 0.14 | 0.80 | 0.65 | 0.55 | 0.82 |

由上表可以看出，超参数 $\sigma$ 越大，接近过程越平滑但拟合系数下降；超参数 $\sigma$ 越小，逼近能力越强。第 4 次交叉验证结果最好，因此将第 4 次交叉验证的输入、输出作为最优输入、输出，此时 $\sigma$ 为 0.1，可以视为在此参数下模型预测所得温度的出现可能性最大。

### 5.4 求解结果与分析

本文利用广义回归神经网络对系统 I 和系统 II 的温度进行预测得到题干中表 2 不同产品质量指标对应的温度如下表：

**表 5　问题 2 结果表**

| 时间 | 指标 A | 指标 B | 指标 C | 指标 D | 系统 I 设定温度 | 系统 II 设定温度 |
|---|---|---|---|---|---|---|
| 2022-1-24 | 79.17 | 22.72 | 10.51 | 17.05 | 858.3099 | 719.9165 |
| 2022-1-24 | 80.1 | 23.34 | 11.03 | 13.29 | 890.0858 | 726.2835 |

由上表可知，产品质量指标不同时，系统 I 和系统 II 对应的温度也会发生变化，四项指标的变化同时作用，对系统设定温度产生影响。

### 5.5 残差正态性检验

为了检验广义回归神经网络模型预测的准确性，本文对交叉验证集上的残差进行正态性检验，结果如下图13：

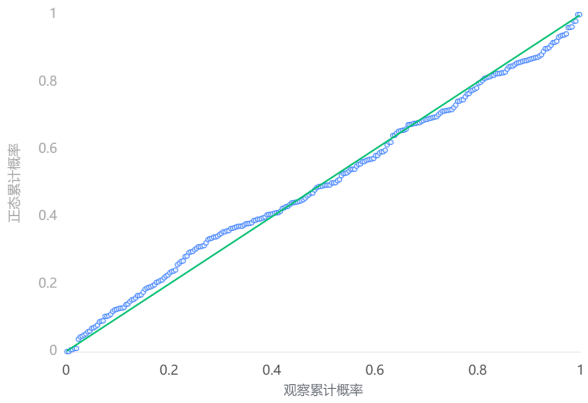


**图 13　广义回归神经网络模型残差检验图**

观察上图可知，广义回归神经网络模型预测所得残差通过了正态性检验。由此可知广义回归神经网络模型残差中不存在可以用到预测中的遗留信息，预测的残差为白噪声。该结果说明了本文广义回归神经网络模型预测结果的准确性与可靠性。

# 六、 问题三的模型建立与求解

## 6.1 问题三的描述与分析

问题三要求结合过程数据以及生产加工数据，利用指定的系统设定温度建立数学模型，对矿石产品合格率进行预测。首先，本文对原矿参数和过程数据的缺失值进行填补。之后，由于表现为合格的产品数量与表现为不合格的产品数量存在差异，本文利用 **SMOTE** 对数据进行**过采样**，使数据量表现均衡。由于合格率是由产品质量指标所决定的，因此本问题的预测对象与问题一类似。为了进步提高预测的准确性，本文基于 ADF 单位根检验与格兰杰因果检验选择 **XGBoost 分类模型**对产品合格率进行预测。最终，本文利用 **Friedman 检验**对 XGBoost 分类模型的效果进行评估。

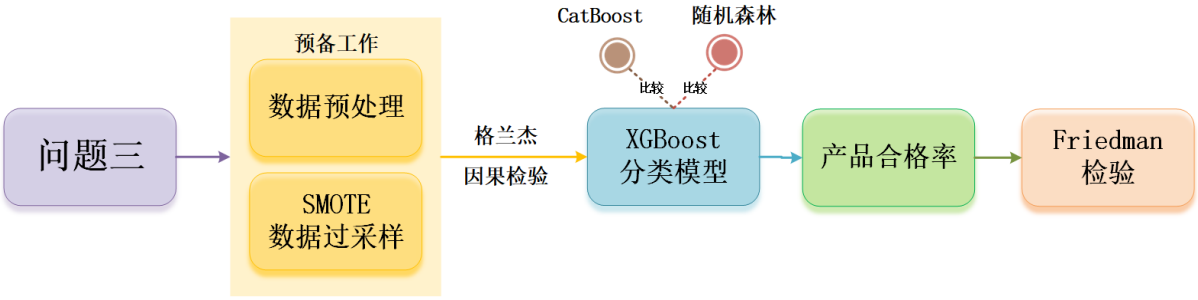


**图 14　问题三思路图**

## 6.2 预备工作

### 6.2.1 数据预处理

对于原矿参数中的缺失值，本文将每日零点对应的该项原矿参数对当日的原矿参数进行补全。对于过程数据中的缺失值，本文利用插值法对其进行补全。此外，对于温度数据，本文与4.2进行相同处理，对附件 2 中温度数据进行小波降噪、DBSCAN 聚类以及缺失值填补。DBSCAN 聚类结果如下图15，其中红色星号为非调温过程的温度变化数据，共有 102932 项。

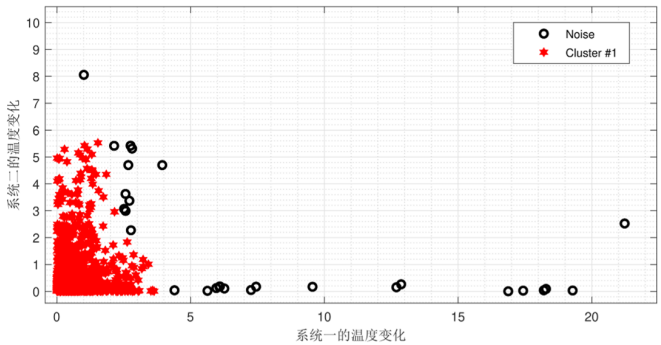


**图 15　温度变化聚类效果图**

### 6.2.2 SMOTE 数据过采样

分析数据发现，仅有 115 个产品表现为合格，而 307 个产品表现为不合格，数据表现不均衡。不平衡样本会导致训练模型侧重样本数目较多的类别，而“轻视”样本数目较少类别，这样模型在测试数据上的泛化能力就会受到影响。因此，为解决不平衡样本问题，我们需对数量少的数据进行过采样操作。

本文采用可处理分类特征过采样方法的 SMOTE 合成少数类过采样技术对数据进行过采样，这是在随机采样的基础上改进的一种过采样算法[5]，其实现过程为：

首先，从少数类样本中选取一个样本 $x_i$。其次，按采样倍率 $N$，从 $x_i$ 的 $K$ 近邻中随机选择 $N$ 个样本 $x_{zi}$。最后，依次在 $x_{zi}$ 和 $x_i$ 之间随机合成新样本，合成公式如下：

$$x_n = x_i + \beta \times (x_{zi} - x_i). \tag{8}$$

数据过采样前后的数据分布如下图16：

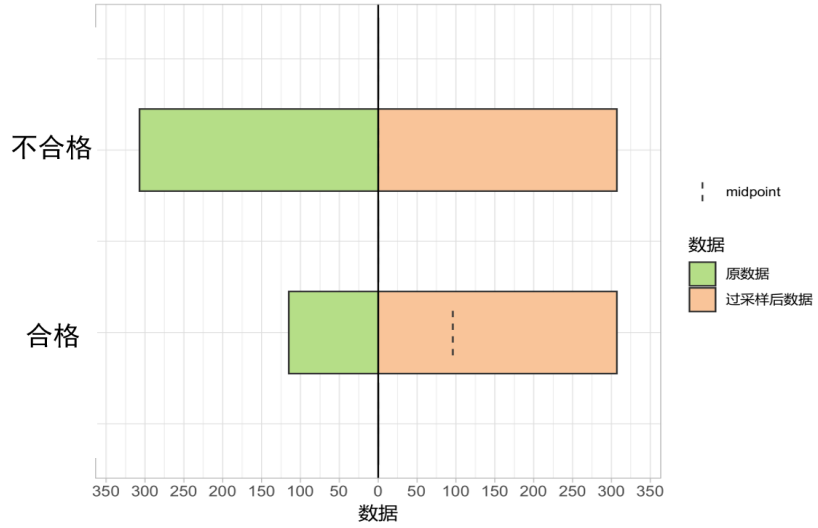


**图 16　过采样前后数据分布图**

由图可以看出，数据过采样后，合格与不合格的数据量差别减少，数据类别达到平衡，从而避免模型在测试数据上的泛化能力受到影响。

### 6.3 基于 ADF 单位根与格兰杰因果检验的模型选择

- **ADF 单位根检验**

首先，本文利用 ADF 单位根检验对系统 I、II 设定温度以及产品合格情况序列的平稳性进行分析。检验结果如下：

**表 6　ADF 单位根检验结果表**

| 变量 | t 统计量 | p 值 | 临界值 1% | 临界值 5% | 临界值 10% |
|---|---|---|---|---|---|
| 系统 I 温度 | -4.198 | 0.001*** | -3.443 | -2.867 | -2.57 |
| 系统 II 温度 | -4.924 | 0.000*** | -3.443 | -2.867 | -2.57 |
| 是否合格 | -6.64 | 0.000*** | -3.443 | -2.867 | -2.57 |

由上表可知，数据检验后 $t$ 小于各检验的临界值，且 $p < 0.05$，系统 I、II 设定温度以及产品合格情况序列均为平稳序列。

- **格兰杰因果检验**

本文基于上述系统 I、II 设定温度以及产品合格情况序列，对系统设定温度与产品合格情况进行格兰杰因果检验，验证系统设定温度对产品合格情况的影响。检验结果如下表：

**表 7　格兰杰因果检验结果表**

| 配对样本 | | F 统计量 | p 值 |
|---|---|---|---|
| 系统 I 温度 | 是否合格 | 5.916 | 0.015** |
| 系统 II 温度 | 是否合格 | 3.14 | 0.077* |

观察上表可知，系统 I 设定温度和系统 II 设定温度对产品合格情况均表现显著。依据该检验结果并结合问题一中的分析，本文选择 XGBoost 分类模型进行后续预测。

### 6.4 基于 XGBoost 分类模型预测系统设定温度

由于合格率是由产品质量指标所决定的，因此问题三的预测对象与问题一中类似。为了进步提高预测的准确性，本文选择 XGBoost 分类模型对产品合格率进行预测。XGBoost 分类模型的预测原理与式 (1) 和式 (2) 类似。

设两个系统设定温度数据、四项原矿参数以及四项过程数据分别为 $t_{i,1}, \cdots, t_{i,10}$。产品合格率为 $u_i$。XGBoost 分类模型对合格率的预测满足下式[5]：

$$\tilde{u_i} = \sum_{k=1}^{K} g_k(t_{i,j}), \quad g_k \in \Gamma; j = 1, 2, \cdots, 10. \tag{9}$$

其中 $g(x)$ 是回归树，$\Gamma$ 是假设空间：

$$\Gamma = \{g(x) = o_{q(x)}\}(q : R^m \to T, o \in R^T). \tag{10}$$

其中，$q(x)$ 表示样本 $x$ 分到了某个叶子节点上，$o$ 是叶子节点的分数。$o_{q(x)}$ 表示回归树对产品合格情况的预测值，基于产品合格情况求出当日产品合格率。

### 6.5 模型求解

为了验证 XGBoost 分类模型预测效果的优越性，本文将 XGBoost 与随机森林以及 CatBoost 的预测效果进行比较，比较结果如下图17。

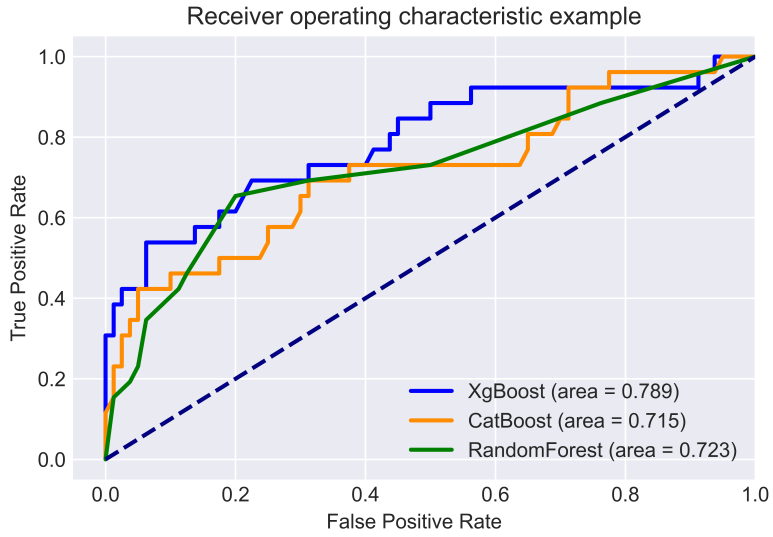


**图 17　三种分类方法对应 ROC 曲线图**

由上图可以看出 XGBoost 算法对应 ROC 曲线下方 AUC 面积最大，说明基于 XGBoost 算法的预测模型性能良好，且优于随机森林与 CatBoost 的方法[9]。

本文对上述 XGBoost 分类模型进行了训练和预测，并利用随机森林与 CatBoost 与其进行对比，三种分类算法在交叉验证集上的效果如下表所示：

**表 8　三种算法预测效果对比表**

| | 准确率 | 召回率 | 精准度 | F1 分数 |
|---|---|---|---|---|
| XGBoost | 0.83 | 0.83 | 0.82 | 0.82 |
| CatBoost | 0.80 | 0.80 | 0.79 | 0.79 |
| 随机森林 | 0.76 | 0.76 | 0.75 | 0.75 |

由上表数据可知，XGBoost 的准确率、召回率、精确率以及 F1 分数均高于随机森林与 CatBoost。因此，XGBoost 的预测效果优于随机森林与 CatBoost，验证了本文模型选择与建立的合理性、优良性。

### 6.6 求解结果与分析

利用 XGBoost 分类模型预测得到 4 月 8 日与 4 月 9 日两种温度组合下的平均合格率结果如下表：

**表 9　问题 3 结果表**

| 时间 | 系统 I 设定温度 | 系统 II 设定温度 | 合格率 |
|---|---|---|---|
| 2022-4-8 | 341.4 | 665.04 | 0.383573 |
| 2022-4-9 | 1010.32 | 874.47 | 0.151063 |

观察上表合格率结果发现，两种系统温度组合下的合格率均较低。本文分析这是由于矿石加工过程为 2 小时整，若要保证矿石成功加工，2 小时内的温度须保持平稳。因此，部分矿石未成功进行加工，从而导致了较低的合格率。

### 6.7 基于 Friedman 检验的模型效果评估

本文利用 Friedman 检验对 XGBoost 分类模型的预测效果进行评估，基于十个交叉验证集进行检验，并与 CatBoost 与随机森林分类模型进行比较。检验结果如下表：

**表 10　Friedman 检验结果表**

| 数据集 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | 平均序值 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| XGBoost | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2.5 | 1 | 1.15 |
| CatBoost | 2 | 2.5 | 2 | 2 | 3 | 3 | 2.5 | 2 | 2.5 | 2 | 2.35 |
| 随机森林 | 3 | 2.5 | 2 | 3 | 2 | 2 | 2.5 | 3 | 1 | 3 | 2.4 |

基于上表中数据计算可得 $F = 24.421$，$p = 0.01$，$F$ 大于 $\alpha = 0.05$ 时的 $F$ 检验临界值 5.143，因此拒绝” 所有算法性能相同” 这个假设。由上表可知，XGBoost 分类模型的平均序值为 1.15，与其余两种方法的临界值域没有交叠，说明 XGBoost 分类模型显著优于 CatBoost 与随机森林分类模型。上述结果说明了本文选择模型的合理性以及 XGBoost 分类模型的优越性与预测结果可靠性。

# 七、 问题四的模型建立与求解

## 7.1 问题四的描述与分析

问题四要求分析在指定合格率条件下的系统设定温度，并进行敏感性分析以及准确性分析。首先，本文以最小温度变化量为目标建立**温度调控规划模型**，结合问题三的XGBoost分类模型以及题干条件对规划模型进行约束。接着利用帝国竞争算法对规划模型进行求解。之后，本文从**宏观**角度对温度调控规划模型进行**敏感性分析**，从**微观**角度对模型进行**准确性分析**。

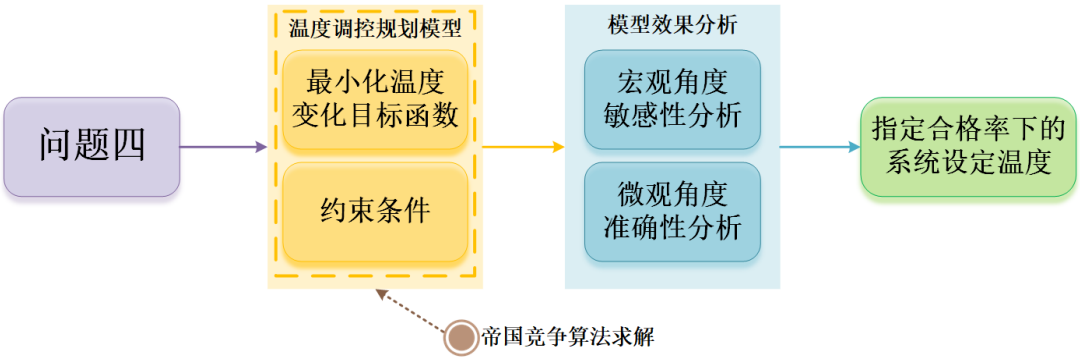


**图 18　问题四思路图**

## 7.2 温度调控规划模型建立

本问题要求对达到合格率要求对应的系统设定温度进行求解，由此本文建立温度调控规划模型，求解得到符合条件的系统 I 和系统 II 设定温度。设第 $i$ 天对应系统 $j$ 设定温度为 $t_{i,j}$，邻日的温度变化值 $\triangle t_{i,j}$ 满足：

$$\triangle t_{i,j} = t_{i,j} - t_{i-1,j}, \quad j = 1, 2. \tag{11}$$

其中，$t_{i-1,j}$ 表示第 $i-1$ 天对应系统 $j$ 设定温度。

本文建立温度调控规划模型的约束条件以及目标函数如下：

- **约束条件**

依据问题三中的 XGBoost 分类模型可得第 $i$ 天产品合格率 $\tilde{u_i}$ 满足下式：

$$\tilde{u_i} = \sum_{k=1}^{K} g_k(t_{i,j}), \quad g_k \in \Gamma; j = 1, 2, \cdots, 10. \tag{12}$$

其中，$t_{i,3}, \cdots, t_{i,10}$ 分别表示第 $i$ 天的四项原矿参数和四项过程数据指标。

由题知产品合格率需达到表 5 中的产品合格率要求，即有：

$$\tilde{u_i} \geq c. \tag{13}$$

其中，$c$ 为题干中的合格率要求。

- **目标函数**

由于产品加工过程中，较大的温度变化会导致较长的时间消耗，并且产生不必要的成本。为了高效地进行矿石加工，以最小的消耗达到较好的合格效果，本文以最小化温度变化为目标，目标函数如下：

$$\min w_i = \sum_{j=1}^{2} \triangle t_{i,j}. \tag{14}$$

其中，$w_i$ 表示第 $i$ 天两系统设定温度变化之和。

综上所述，本文建立温度调控规划模型如下：

$$\min w_i = \sum_{j=1}^{2} \triangle t_{i,j}.$$

$$st. \begin{cases} \tilde{u_i} = \sum_{k=1}^{K} g_k(t_{i,j}), & g_k \in \Gamma; j = 1, 2, \cdots, 10, \\ \triangle t_{i,j} = t_{i,j} - t_{i-1,j}, & j = 1, 2, \\ \tilde{u_i} \geq c. \end{cases} \tag{15}$$

### 7.3 温度调控模型的效果分析

#### 7.3.1 基于宏观角度的敏感性分析

本文以 4 月 9 日的数据为参考进行温度调控规划，已知改日产品合格率为 0.151。本文通过调节指定合格率，分析系统 I 和系统 II 的设定温度变化情况，从宏观的角度对模型的敏感性进行分析。不同合格率下的温度变化情况如下图19：

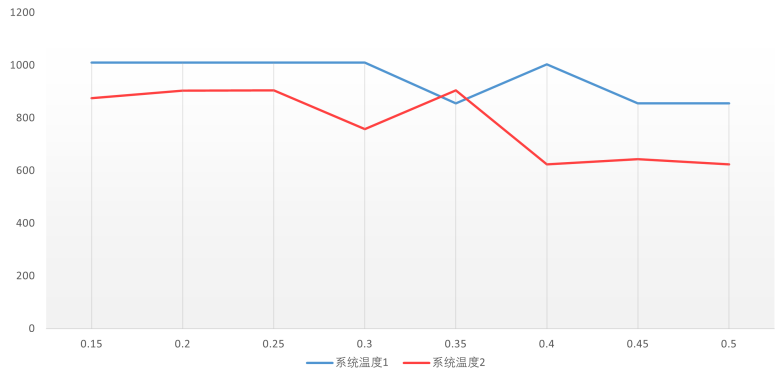


**图 19　温度调控模型敏感性分析图**

由上图可知，合格率与温度大小间并没有明显的线性关系，而且可以发现合格率越大，温度呈现出下降趋势。其中，系统 II 设定温度受合格率变化的影响较大，合格率对系统 II 温度的波动更为敏感。由合格率为 0.2、0.25、0.3 处可以看出，系统 I 和系统 II 的设定温度变化对合格率有着交互影响。

### 7.3.2 基于微观角度的准确性分析

本文以日合格率为 0.25 为例，提取出该日每分钟合格率，从微观的角度分析温度调控规划模型的准确性，分析结果如下图20：

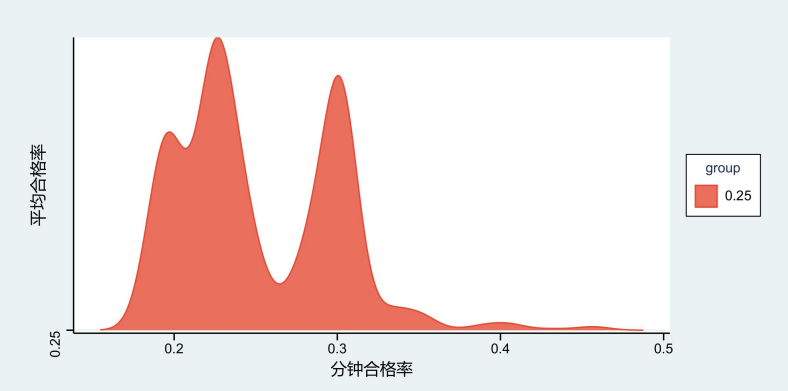


**图 20　温度调控模型准确性分析图**

由上图可知，每分钟合格率为 0.1 和 0.2 时分布较为集中，0.3 和 0.4 时合格率的分布较为分散，说明合格率呈现出向下集中的趋势。同时，分钟合格率比日合格率更为精确微观，上述结果印证了温度调控规划模型的准确性。

### 7.4 模型求解

本文利用帝国竞争算法求解温度调控规划模型，帝国竞争算法的伪代码如下[3]：

**Algorithm 2** Function Imperialist Competitve Algorithm(x)

1: **Initialize** /* set multiple initial national solutions. $X_n, n = 1, \cdots, n_{max}$。Select $N$ solutions with the best quality from them as colonial countries of $N$ empires, $X_1, X_2, \cdots, X_N$, and distribute them to national colonies according to the proportion of national power, thus obtaining $N$ empires $E_1, E_2, \cdots, E_N$ */

2: iteration = 1; **Repeat for** $i \in [1, N]$ **do**

3: Empire Assimilate($E_i$); Colonies Revolve($E_i$); Empire Posses($E_i$)

4: **end**

5: Empire Unite($E_1, E_2, \cdots, E_N$) EmpireCompetition($E_1, E_2, \cdots, E_N$)

6: EmpireCompetition($E_1, E_2, \cdots, E_N$)

7: iteration ← iteration + 1; Make a new iteration.

8: until There is only one empire left or the number of iterations has been reached.

利用帝国竞争算法求解温度调控规划模型的迭代曲线如下图21：

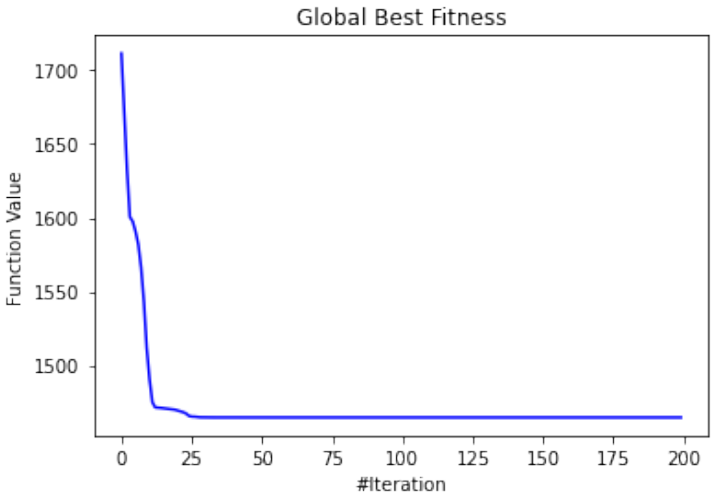


**图 21　规划模型求解迭代曲线图**

### 7.5 求解结果与分析

对温度调控规划模型进行求解得到 2022 年 4 月 10 日 80% 的合格率可以达到，而 4 月 11 日 99% 的合格率无法达到，具体结果如下表：

**表 11　问题四结果表**

| 时间 | 合格率 | 能否达到 | 系统 I 设定温度 | 系统 II 设定温度 |
|---|---|---|---|---|
| 2020-4-10 | 80% | 能达到 | 850.11730321 | 513.46985986 |
| 2022-4-11 | 99% | 不能达到 | \ | \ |

由上表中数据可知，过高的合格率如 99% 难以达到，合格率为 80% 时能够通过调控系统设定温度达到。而此时系统 I、II 的设定温度均较低，该结果为追求较高合格率的矿石加工温度调控提供了参考，即加工过程中各系统温度不宜过高，要合理进行调控。

## 八、 模型评价

### 8.1 模型总结

本文利用小波降噪和 DBSCAN 聚类对附件一和二的数据进行处理，利用 XGBoost 回归模型预测附件一中的相应指标，并预测出其对应的置信区间以反映其可能性，之后建立基于广义回归神经网络 (GRNN) 的模型对温度进行双输出变量的预测，之后利用 XGBoost 分类模型对合格率进行分类预测，再建立温度调控规划模型结合帝国竞争算法求解出系统最佳温度，最后综合分析结果，提出有针对性意见。

## 8.2 模型优缺点分析

优点：

（1）第一问对预测值和可能性分别求解，建立基于置信区间预测的 XGBoost 回归模型，利用置信区间和残差正态检验使结果的展示更为全面；

（2）本文在第二问中利用在双输出变量和小样本变现优异的广义回归神经网络，并用 10-fold 交叉验证和残差正态检验以完善模型；

（3）在第三问中建立基于 XGBoost 分类的预测模型，从多维度对分类的效果进行评价，使结果更加合理完善；

（4）在第四问中结合分类效果，建立基于帝国竞争算法的温度调控规划模型，通过调整合格率的约束从宏微观进行灵敏性和准确性的分析，得出最佳的调控温度。

缺点：

（1）在 DBSCAN 聚类中并没有对参数的确定进行严格的探讨；

（2）在广义回归的神经网络预测模型中并没有考虑变量之中的交互影响。

## 8.3 模型改进与展望

首先，本文对数据进行小波降噪和 DBSCAN 聚类，在聚类中可以对不同的聚类参数之间进行比较，得到更合理的聚类参数。在广义回归神经网络模型中可以引入变量之间的交互影响使结果更加完善。在基于 XGBOOST 分类预测模型中，可以引入更多的集成学习算法进行综合分析，使结果更加完善。规划模型中可以以小时为单位对温度的调控进行更详细的探讨，使结果更具有现实意义。

# 参考文献


[1] Goswami Agam Das,Mishra M K,Patra Dipti. Evaluation of machine learning algorithms for grade estimation using GRNN and SVR[J]. Engineering Research Express,2022,4(3):33-35.

[2] Li ZhuoXuan,Shi XinLi,Cao JinDe,Wang XuDong,Huang Wei.CPSO-XGBoost segmented regression model for asphalt pavement deflection basin area prediction[J].Science China(Technological Sciences),2022,65(07):1470-1481.

[3] Salam Karim, Jasim Saade, Mireya Romero Parra Rosario. Optimization of doubly-fed induction generator (dfig)based wind turbine to achieve maximum power generation with imperialist competitive algorithm (ica). [J]. Science progress, 2022, 105(3):1-3.

[4] Shehadeh Ali,Alshboul Odey,Al Mamlook Rabia Emhamed,Hamedat Ola. Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression[J]. Automation in Construction,2021,5(9):9-12.

[5] Wu Jiaju,Kong Linggang,Yi Ming,Chen Qiuxian,Cheng Zheng,Zuo Hongfu. Prediction and Screening Model for Products Based on Fusion Regression and XGBoost Classification[J]. Computational Intelligence and Neuroscience,2022,6(5):10-11.

[6] 陈文龙, 时宏伟. 基于 KD 树改进的 DBSCAN 聚类算法 [J]. 计算机系统应用,2022,31(2):305-310.

[7] 高虹雷, 门昌骞, 王文剑. 多核贝叶斯优化的模型决策树算法 [J]. 国防科技大学学报,2022,44(03):67-76.

[8] 罗学辉, 张勇, 陈占生, 李玄辉, 陈雪. 金矿石加工及测试质量过程控制 [J]. 黄金,2012,33(03):61-64.

[9] 马景义, 谢邦昌. 用于分类的随机森林和 Bagging 分类树比较 [J]. 统计与信息论坛,2010,25(10):18-22.

[10] 饶运章, 袁博云, 吴卫强, 孙翔, 陈斌. 基于 GRNN 模型的硫化矿石堆氧化自热温度预测 [J]. 金属山,2016,47(06):149-152.

[11] 杨琪威. 线性模型平均中惩罚因子选择的交叉验证法 [D]. 华东师范大学,2022,20(22):48-50.

# 附录 A　支撑材料清单

- **图表**

  -附件 1 训练集数据表 (1)

  -附件 1 训练集数据表 (2)

  -附件 1 训练集数据表 (3)

  -附件 2 训练集数据表 (1)

  -附件 2 训练集数据表 (2)

  -附件 2 训练集数据表 (3)

  -附件 2 训练集数据表 (4)

- **代码**

  -Python 源程序

  -Matlab 源程序

# 附录 B　图表

**附件 1 训练集数据表 (1)**

| 指标 A | 指标 B | 指标 C | 指标 D | 参数 1 | 参数 2 | 参数 3 | 参数 4 | I 温度 | II 温度 | 加工 2h |
|---|---|---|---|---|---|---|---|---|---|---|
| 78.39 | 25.22 | 12.93 | 14.28 | 49.24 | 90.38 | 46.13 | 28.16 | 853.57 | 766.2 | 是 |
| 79.22 | 24.6 | 12.41 | 13.7 | 49.24 | 90.38 | 46.13 | 28.16 | 854.81 | 768.08 | 是 |
| 80.51 | 22.41 | 11.72 | 13.38 | 49.24 | 90.38 | 46.13 | 28.16 | 595.22 | 686.44 | 是 |
| 79.18 | 23.97 | 12.24 | 13.79 | 49.24 | 90.38 | 46.13 | 28.16 | 594.69 | 686.45 | 是 |
| 79.29 | 23.93 | 12.24 | 17.05 | 49.24 | 90.38 | 46.13 | 28.16 | 876.82 | 758.58 | 是 |
| 79.45 | 23.34 | 11.72 | 17.23 | 49.24 | 90.38 | 46.13 | 28.16 | 876.96 | 758.34 | 是 |
| 79.28 | 23.3 | 11.37 | 16.52 | 49.24 | 90.38 | 46.13 | 28.16 | 876.86 | 758.4 | 是 |
| 79.96 | 22.81 | 11.37 | 16.47 | 49.24 | 90.38 | 46.13 | 28.16 | 877.04 | 758.42 | 是 |
| 80.41 | 21.91 | 10.68 | 17.72 | 49.24 | 90.38 | 46.13 | 28.16 | 876.82 | 758.41 | 是 |
| 80.26 | 23.21 | 10.34 | 16.65 | 49.24 | 90.38 | 46.13 | 28.16 | 1374.23 | 829.53 | 是 |
| 80 | 23.21 | 10.68 | 16.78 | 49.24 | 90.38 | 46.13 | 28.16 | 1376.26 | 830.27 | 是 |
| 79.56 | 24.19 | 11.89 | 20.45 | 49.24 | 90.38 | 46.13 | 28.16 | 1389.12 | 841.28 | 是 |
| 79.07 | 24.64 | 12.07 | 15.22 | 49.24 | 90.38 | 46.13 | 28.16 | 1389.16 | 840.5 | 是 |
| 80.16 | 23.12 | 11.2 | 13.12 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.64 | 841.52 | 是 |
| 80.19 | 22.76 | 11.03 | 13.83 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.09 | 840.9 | 是 |
| 79.37 | 24.02 | 12.07 | 13.52 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.68 | 841.41 | 是 |
| 79.18 | 24.37 | 11.89 | 14.1 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.88 | 841.5 | 是 |
| 80.65 | 21.87 | 9.82 | 16.43 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.93 | 841.5 | 是 |
| 80.14 | 22.94 | 10.51 | 15.89 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.5 | 841.63 | 是 |
| 79.07 | 23.66 | 11.2 | 17.68 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.98 | 841.8 | 是 |
| 79.72 | 22.09 | 10.34 | 18.44 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.59 | 841.88 | 是 |
| 79.45 | 23.08 | 11.03 | 16.07 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.11 | 841.74 | 是 |

**附件 1 训练集数据表 (2)**

| 指标 A | 指标 B | 指标 C | 指标 D | 参数 1 | 参数 2 | 参数 3 | 参数 4 | I 温度 | II 温度 | 加工 2h |
|---|---|---|---|---|---|---|---|---|---|---|
| 80.45 | 21.33 | 10.16 | 16.69 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.15 | 841.42 | 是 |
| 80.61 | 20.62 | 9.99 | 17.23 | 47.95 | 86.09 | 45.89 | 25.99 | 1397.3 | 841.6 | 是 |
| 80.07 | 22.85 | 11.55 | 17.36 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.19 | 841.61 | 是 |
| 80.88 | 22.45 | 11.03 | 18.8 | 47.95 | 86.09 | 45.89 | 25.99 | 1396.64 | 841.77 | 是 |
| 80.26 | 22.76 | 11.37 | 19.91 | 47.95 | 86.09 | 45.89 | 25.99 | 1398.01 | 842.51 | 是 |
| 80.57 | 23.03 | 11.55 | 15.31 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.75 | 841.38 | 是 |
| 80.44 | 22.99 | 11.37 | 15.04 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.57 | 841.44 | 是 |
| 80.21 | 22.49 | 11.37 | 14.01 | 47.95 | 86.09 | 45.89 | 25.99 | 1395.84 | 841.41 | 是 |
| 79.51 | 22.54 | 11.2 | 13.7 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.1 | 841.57 | 是 |
| 80.77 | 22.14 | 11.37 | 13.3 | 54.04 | 97.16 | 48.97 | 22.14 | 1397.16 | 841.93 | 是 |
| 80.48 | 22.54 | 11.03 | 13.03 | 54.04 | 97.16 | 48.97 | 22.14 | 1395.7 | 841.03 | 是 |
| 80.3 | 22.63 | 10.68 | 13.34 | 54.04 | 97.16 | 48.97 | 22.14 | 1395.75 | 841.36 | 是 |
| 80.96 | 21.78 | 10.34 | 12.49 | 54.04 | 97.16 | 48.97 | 22.14 | 1395.79 | 840.82 | 是 |
| 81.05 | 21.6 | 10.51 | 12.49 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.63 | 841.5 | 是 |
| 81.27 | 21.42 | 10.51 | 12.89 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.72 | 841.61 | 是 |
| 81.67 | 21.82 | 10.68 | 12.85 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.28 | 841.36 | 是 |
| 81.12 | 23.39 | 11.2 | 13.34 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.63 | 841.79 | 是 |
| 80.44 | 24.69 | 10.34 | 18.3 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.32 | 841.51 | 是 |
| 79.38 | 25.31 | 11.03 | 19.78 | 54.04 | 97.16 | 48.97 | 22.14 | 1396.37 | 841.72 | 是 |
| 78.36 | 25.22 | 12.41 | 16.07 | 54.04 | 97.16 | 48.97 | 22.14 | 1140.17 | 804.66 | 是 |
| 78.52 | 25.27 | 11.89 | 16.96 | 54.04 | 97.16 | 48.97 | 22.14 | 580.59 | 633.81 | 是 |
| 77.73 | 25.22 | 12.24 | 16.25 | 54.04 | 97.16 | 48.97 | 22.14 | 580.24 | 633.71 | 是 |

**附件 1 训练集数据表 (3)**

| 指标 A | 指标 B | 指标 C | 指标 D | 参数 1 | 参数 2 | 参数 3 | 参数 4 | I 温度 | II 温度 | 加工 2h |
|---|---|---|---|---|---|---|---|---|---|---|
| 78.97 | 25.09 | 11.89 | 22.11 | 54.04 | 97.16 | 48.97 | 22.14 | 305.53 | 574.32 | 是 |
| 79.21 | 23.93 | 11.72 | 26.17 | 54.04 | 97.16 | 48.97 | 22.14 | 304.24 | 574.39 | 是 |
| 79.35 | 23.21 | 11.55 | 27.38 | 54.04 | 97.16 | 48.97 | 22.14 | 306.81 | 574.65 | 是 |
| 80.02 | 22.9 | 11.55 | 23.04 | 54.04 | 97.16 | 48.97 | 22.14 | 305.34 | 574.2 | 是 |
| 78.74 | 23.52 | 11.55 | 14.46 | 54.55 | 99.46 | 45.04 | 22.19 | 1151.09 | 768.58 | 是 |
| 78.31 | 25.13 | 12.07 | 16.11 | 54.55 | 99.46 | 45.04 | 22.19 | 1151.4 | 768.89 | 是 |
| 77.36 | 26.25 | 12.41 | 15.08 | 54.55 | 99.46 | 45.04 | 22.19 | 1151.22 | 768.58 | 是 |
| 80.04 | 21.96 | 11.37 | 15.26 | 54.55 | 99.46 | 45.04 | 22.19 | 717.94 | 633.04 | 是 |
| 78.76 | 24.64 | 12.59 | 14.23 | 54.55 | 99.46 | 45.04 | 22.19 | 717.28 | 734.49 | 是 |
| 79.75 | 24.42 | 12.59 | 15.89 | 53.21 | 100.56 | 46.31 | 21.65 | 999.28 | 759.03 | 是 |
| 79.89 | 23.93 | 12.59 | 16.02 | 53.21 | 100.56 | 46.31 | 21.65 | 998.89 | 759.24 | 是 |
| 79.79 | 25.04 | 12.93 | 18.66 | 53.21 | 100.56 | 46.31 | 21.65 | 998.27 | 758.83 | 是 |
| 79.59 | 23.43 | 12.59 | 15.8 | 53.21 | 100.56 | 46.31 | 21.65 | 861.7 | 729.2 | 是 |
| 80.44 | 22.18 | 12.07 | 13.61 | 53.21 | 100.56 | 46.31 | 21.65 | 862.54 | 729.48 | 是 |
| 79.75 | 22.85 | 12.76 | 13.97 | 53.21 | 100.56 | 46.31 | 21.65 | 998.66 | 786.74 | 是 |
| 79.71 | 22.41 | 12.59 | 15.75 | 53.21 | 100.56 | 46.31 | 21.65 | 998.84 | 786.85 | 是 |
| 80.13 | 21.82 | 12.41 | 16.56 | 53.21 | 100.56 | 46.31 | 21.65 | 999.02 | 786.86 | 是 |
| 79.58 | 24.73 | 12.93 | 16.11 | 53.21 | 100.56 | 46.31 | 21.65 | 999.06 | 787.05 | 是 |
| 78.63 | 24.24 | 12.76 | 15.53 | 53.21 | 100.56 | 46.31 | 21.65 | 1136.94 | 816.71 | 是 |
| 78.91 | 23.61 | 12.24 | 15.93 | 53.21 | 100.56 | 46.31 | 21.65 | 1136.72 | 816.69 | 是 |
| 79.07 | 23.57 | 12.76 | 15.49 | 53.21 | 100.56 | 46.31 | 21.65 | 1135.71 | 816.02 | 是 |
| 79.28 | 23.17 | 11.89 | 14.28 | 53.21 | 100.56 | 46.31 | 21.65 | 1135.98 | 816.55 | 是 |

**附件 2 训练集数据表 (1)**

| I 温度 | II 温度 | 合格 | 参数 1 | 参数 2 | 参数 3 | 参数 4 | 过程 1 | 过程 2 | 过程 3 | 过程 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1273.51 | 937.49 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 233.31 | 173.87 |
| 1272.84 | 936.67 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 238.73 | 168.27 |
| 1404.72 | 960.09 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 237.92 | 150.74 |
| 1405.16 | 960.24 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 228 | 151.76 |
| 1405.6 | 960.43 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 221.47 | 155.6 |
| 1405.87 | 961.04 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 224.07 | 154.51 |
| 1405.12 | 960.23 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 229.44 | 151.69 |
| 1404.54 | 959.95 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 236.17 | 150.5 |
| 1402.95 | 959.22 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 238.51 | 151.07 |
| 1404.41 | 960.26 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 237.03 | 156.24 |
| 1405.69 | 960.63 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 234.69 | 159.88 |
| 1406.17 | 960.9 | 0 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 242.39 | 154.71 |
| 1405.03 | 960.22 | 1 | 55.26 | 108.03 | 43.29 | 20.92 | 1.25 | 3.09 | 249.45 | 148.89 |
| 1404.85 | 960.18 | 1 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 249.6 | 144.25 |
| 1404.28 | 959.53 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 246.05 | 143.75 |
| 1404.19 | 959.9 | 1 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 251.45 | 142.51 |
| 1404.5 | 959.97 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 257.39 | 141.83 |
| 1404.85 | 960.31 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 255.8 | 142.84 |
| 1404.72 | 935.78 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 247.43 | 148.17 |
| 1404.54 | 935.56 | 1 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 246.56 | 150.78 |
| 1404.85 | 935.87 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 239.3 | 154.6 |
| 1403.35 | 934.66 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 234.24 | 156.15 |

**附件 2 训练集数据表 (2)**

| I 温度 | II 温度 | 合格 | 参数 1 | 参数 2 | 参数 3 | 参数 4 | 过程 1 | 过程 2 | 过程 3 | 过程 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1404.94 | 935.49 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 225.73 | 161.52 |
| 1404.85 | 935.44 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 221.9 | 164.81 |
| 1404.32 | 935.28 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 226.88 | 165 |
| 1403.93 | 935.5 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 232.65 | 163.44 |
| 1404.58 | 935.48 | 0 | 55.28 | 102.38 | 46.13 | 20.1 | 1.25 | 3.09 | 237.62 | 160.1 |
| 1404.58 | 935.63 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 236.88 | 158.56 |
| 1404.1 | 935.49 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 235.32 | 159.59 |
| 1406.13 | 936.3 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 235.28 | 158.94 |
| 1290.92 | 915.76 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 248.96 | 145.74 |
| 1290.39 | 915.52 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 252.11 | 144.05 |
| 1291.09 | 916.08 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 244.98 | 152.58 |
| 1291.63 | 916.2 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 237.92 | 159.94 |
| 1003.61 | 866.44 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 237.72 | 165.88 |
| 1003.53 | 866.23 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 241.22 | 166.58 |
| 427.28 | 738.71 | 0 | 54.04 | 102.21 | 47.94 | 21.3 | 1.25 | 3.09 | 265.84 | 152.3 |
| 574.67 | 764.19 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 273.79 | 148.19 |
| 576.7 | 764.51 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 272.08 | 149.39 |
| 576.34 | 765.48 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 265.76 | 150.61 |
| 576.21 | 764.3 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 260.82 | 145.83 |
| 585.41 | 794.13 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 257.86 | 144.89 |
| 585.76 | 794.43 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 254.03 | 147.32 |
| 585.14 | 793.78 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 243.89 | 150.79 |

**附件 2 训练集数据表 (3)**

| I 温度 | II 温度 | 合格 | 参数 1 | 参数 2 | 参数 3 | 参数 4 | 过程 1 | 过程 2 | 过程 3 | 过程 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 585.85 | 793.47 | 0 | 56.43 | 112.74 | 43.54 | 20.14 | 1.25 | 3.09 | 238.17 | 152.16 |
| 1390.48 | 934.1 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 245.14 | 154.34 |
| 1388.59 | 932.93 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 249.86 | 154.51 |
| 1132.4 | 859.53 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 255.7 | 149.48 |
| 1132.93 | 859.63 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 257.69 | 145.56 |
| 1179.98 | 818.74 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 285.9 | 153.02 |
| 1178.63 | 818.12 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 288.78 | 153.37 |
| 739.96 | 628.08 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 267.87 | 154.24 |
| 740.01 | 627.58 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 262.63 | 157.84 |
| 739.92 | 628.01 | 0 | 54.89 | 109.21 | 43.6 | 21.64 | 1.25 | 3.09 | 271.87 | 147.15 |
| 1146.77 | 804.04 | 1 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 253.43 | 148.39 |
| 1385.74 | 907.87 | 0 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 237.98 | 151.12 |
| 1385.95 | 907.34 | 0 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 240.63 | 153.21 |
| 1376.35 | 912.32 | 0 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 262.33 | 146.37 |
| 1375.81 | 911.69 | 0 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 258.61 | 142.05 |
| 1376.79 | 881.5 | 0 | 57.34 | 112.91 | 42.87 | 18.8 | 1.25 | 3.09 | 260.22 | 147.52 |
| 1377.01 | 881.77 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 270.41 | 149.03 |
| 1376.48 | 881.63 | 1 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 277.21 | 148.28 |
| 1376.12 | 881.58 | 1 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 243.15 | 161.5 |
| 1376.17 | 881.84 | 1 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 255.45 | 154.07 |
| 1376.3 | 881.68 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 255.03 | 167.37 |
| 1376.75 | 881.76 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 257.42 | 155.24 |

**附件 2 训练集数据表 (4)**

| I 温度 | II 温度 | 合格 | 参数 1 | 参数 2 | 参数 3 | 参数 4 | 过程 1 | 过程 2 | 过程 3 | 过程 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1376.48 | 881.7 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 257.48 | 147.85 |
| 1378.2 | 883.5 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 256.29 | 148.53 |
| 1376.84 | 882.63 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 246.67 | 154.63 |
| 1376.96 | 881.57 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 239.99 | 158.92 |
| 1376.97 | 881.85 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 242.65 | 158.23 |
| 1377.28 | 881.77 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 248.14 | 155.48 |
| 1376.53 | 881.81 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 265.15 | 143.39 |
| 1376.88 | 881.51 | 0 | 58.81 | 109.09 | 45.89 | 17.47 | 1.25 | 3.09 | 275.23 | 135.64 |
| 1376.26 | 881.65 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 277.29 | 134.74 |
| 1376.04 | 881.56 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 273.62 | 138.08 |
| 1376.61 | 881.69 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 264.77 | 143.03 |
| 1376.83 | 881.86 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 259.95 | 145.06 |
| 1376.75 | 881.77 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 265.78 | 138.8 |
| 1376.09 | 881.67 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 272.71 | 133 |
| 1376.53 | 881.78 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 280.04 | 126.7 |
| 1376.04 | 881.69 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 282 | 124.92 |
| 1377.06 | 880.85 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 275.64 | 141.58 |
| 1377.1 | 881.75 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 269.79 | 155.17 |
| 1376.31 | 881.84 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 270.27 | 151.8 |
| 1377.41 | 882.16 | 1 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 273.67 | 142.19 |
| 1376.57 | 882.03 | 1 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 270.67 | 150.1 |
| 1376.17 | 881.89 | 0 | 62.29 | 133.06 | 38.95 | 16.36 | 1.25 | 3.09 | 266.62 | 161.1 |

# 附录 C　Python 源程序

```
# 问题一数据处理
# 导入数据处理包
import numpy as np
import pandas as pd

from sklearn.neighbors import LocalOutlierFactor as LOF


# 读取附件一
xls_one = pd.ExcelFile("附件1(Attachment 1)2022Problem B.xlsx")

# 各个sheet中的数据
temp_data = pd.read_excel(xls_one,'温度(temperature)')
quality_data = pd.read_excel(xls_one,'产品质量(quality of the products)', index_col=0)
mineral_data = pd.read_excel(xls_one,'原矿参数(mineral parameter)', index_col=0)
print(temp_data.shape)
temp_data.head(2)

temp_data["时间 (Time)"] = temp_data["时间 (Time)"].astype("datetime64")
data_range = pd.date_range(start=temp_data["时间 (Time)"].min(),end=temp_data["时间
    (Time)"].max(),freq="T")
temp_data = temp_data.set_index("时间 (Time)").reindex(index = data_range)
print(temp_data.shape)
temp_data.head(2)

temp_data["系统I温度 (Temperature of system I)"] = temp_data["系统I温度 (Temperature of system
    I)"].interpolate(method='spline',order=3)
temp_data["系统II温度 (Temperature of system II)"] = temp_data["系统II温度 (Temperature of
    system II)"].interpolate(method='spline',order=3)

temp_data.to_excel("afterFill.xlsx")

# 问题一：机器学习
# 数据处理
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 机器学习
from sklearn.model_selection import train_test_split,ShuffleSplit
from xgboost import XGBRegressor
from sklearn.metrics import r2_score

# 贝叶斯调参
```

```
from bayes_opt import BayesianOptimization
from sklearn.model_selection import cross_val_score

# 数据引入
data_one = pd.read_excel('forML_Q1.xlsx')
data_one.head(2)

x_train_all = data_one.iloc[:,1:7]

def xgb_cv(max_depth, learning_rate, n_estimators, min_child_weight, subsample,
     colsample_bytree, reg_alpha, gamma):
    val = cross_val_score(estimator=XGBRegressor(max_depth=int(max_depth),
                                          learning_rate=learning_rate,
                                          n_estimators=int(n_estimators),
                                          min_child_weight=min_child_weight,
                                          subsample=max(min(subsample, 1), 0),
                                          colsample_bytree=max(min(colsample_bytree, 1), 0),
                                          reg_alpha=max(reg_alpha, 0), gamma=gamma,
                                              objective='reg:squarederror',
                                          booster='gbtree',
                                          seed=2022), X=x_train_all, y=y_train_all,
                        cv=10).mean()
    return val

xgb_bo = BayesianOptimization(xgb_cv, pbounds={'max_depth': (1, 10),
                                        'learning_rate': (0.01, 0.3),
                                        'n_estimators': (1, 200),
                                        'min_child_weight': (0, 20),
                                        'subsample': (0.001, 1),
                                        'colsample_bytree': (0.01, 1),
                                        'reg_alpha': (0.001, 20),
                                        'gamma': (0.001, 10)})
xgb_bo.maximize(n_iter=100, init_points=10)

# 划分训练集和测试集
y_train_all = data_one.iloc[:,7]
X_train, X_test, y_train, y_test = train_test_split(x_train_all,
                                            y_train_all,
                                            random_state=1,test_size=0.20)
# 进行训练
xgbA = XGBRegressor(
    max_depth= 10,
    learning_rate= 0.1,
    n_estimators=98,
    min_child_weight=6.711332187325609,
    subsample=1.0,
    colsample_bytree=1.0,
```

```
    reg_alpha= 2.3,
    objective='reg:squarederror',
    booster='gbtree',
    seed=2022
)

xgbA.fit(X_train, y_train)

y_pred = xgbA.predict(X_test)
r2_score(y_true=y_test,y_pred=y_pred)

y_train_all = data_one.iloc[:,8]
X_train, X_test, y_train, y_test = train_test_split(x_train_all,
                                                    y_train_all,
                                                    random_state=2,test_size=0.20)
xgbB = XGBRegressor(
    max_depth= 1,
    learning_rate= 0.74,
    n_estimators=75,
    min_child_weight=7,
    subsample=0.55,
    colsample_bytree=1.0,
    reg_alpha= 1.6,
    reg_lambda=0.9,
    objective='reg:squarederror',
    booster='gbtree',
    seed=2022
)

xgbB.fit(X_train, y_train)

y_pred = xgbB.predict(X_test)
r2_score(y_true=y_test,y_pred=y_pred)

y_train_all = data_one.iloc[:,9]
X_train, X_test, y_train, y_test = train_test_split(x_train_all,
                                                    y_train_all,
                                                    random_state=5,test_size=0.20)
xgbC = XGBRegressor(
    max_depth= 1,
    learning_rate= 0.74,
    n_estimators=82,
    min_child_weight=7,
    subsample=0.55,
    colsample_bytree=1.0,
    reg_alpha= 1.6,
    reg_lambda=0.61,
```

```
    objective='reg:squarederror',
    booster='gbtree',
    seed=2022
)

xgbC.fit(X_train, y_train)

y_pred = xgbC.predict(X_test)
r2_score(y_true=y_test,y_pred=y_pred)

y_train_all = data_one.iloc[:,10]
X_train, X_test, y_train, y_test = train_test_split(x_train_all,
                                                    y_train_all,
                                                    random_state=3,test_size=0.20)

xgbD = XGBRegressor(
    max_depth= 10,
    learning_rate= 0.21,
    n_estimators=200,
    min_child_weight=0.2,
    subsample=0.801,
    colsample_bytree=1,
    reg_alpha= 2.0,
    reg_lambda=2.0,
    objective='reg:squarederror',
    booster='gbtree',
    seed=2022
)

xgbD.fit(X_train, y_train)

y_pred = xgbD.predict(X_test)
r2_score(y_true=y_test,y_pred=y_pred)

# 引入预测数据
pre_data = pd.read_excel("p1_predect.xlsx")
pre_data
pre_A = xgbA.predict(pre_data)

pre_B = xgbB.predict(pre_data)

pre_C = xgbC.predict(pre_data)

pre_D = xgbD.predict(pre_data)

pre_res = np.array([pre_A,pre_B,pre_C,pre_D]).T
pre_res_pd = pd.DataFrame(data=pre_res,columns=["指标A (index A)","指标B (index B)","指标C
```

```
    (index C)","指标D (index D)"])
pre_res_pd

pre_res_pd.to_excel("问题一预测结果.xlsx")


# 分位数回归目标
def log_cosh_quantile(alpha):
    def _log_cosh_quantile(y_true, y_pred):
        err = y_pred - y_true
        err = np.where(err < 0, alpha * err, (1 - alpha) * err)
        grad = np.tanh(err)
        hess = 1 / np.cosh(err)**2
        return grad, hess
    return _log_cosh_quantile

splitter = ShuffleSplit(n_splits=1,test_size=.20,random_state=0)

# alpha设置
alpha = 0.95
pre = pd.read_excel("p1_predect.xlsx")

for train_index, test_index in splitter.split(data_one):
    train = data_one.iloc[train_index]
    test = data_one.iloc[test_index]
    # x,y准备训练集
    X = train.iloc[:,1:7]
    y = train.iloc[:,-1] - 17
    # 测试集
    X_test = test.iloc[:,1:7]
    y_test = test.iloc[:,-1] - 17

    # over_predict
    model = XGBRegressor(
        objective=log_cosh_quantile(alpha),
        max_depth= 10,
        learning_rate= 0.21,
        n_estimators=200,
        min_child_weight=0.2,
        subsample=0.801,
        colsample_bytree=1,
        reg_alpha= 2.0,
        reg_lambda=2.0,
        booster='gbtree',
        seed=2022
    )
    model.fit(X,y)
```

```
    y_upper_smooth = model.predict(X_test)

    y_prefor1 = model.predict(pre)

    # under predict
    model = XGBRegressor(
        objective=log_cosh_quantile(1-alpha),
        max_depth= 1,
        learning_rate= 0.74,
        n_estimators=82,
        min_child_weight=7,
        subsample=0.55,
        colsample_bytree=1.0,
        reg_alpha= 1.6,
        reg_lambda=0.61,
        booster='gbtree',
        seed=2022
    )
    model.fit(X, y)

    y_lower_smooth = model.predict(X_test)

    y_prefor2 = model.predict(pre)

    res = pd.DataFrame({'lower_bound':y_lower_smooth, 'true_duration':y_test,'upper_bound':
        y_upper_smooth})


# 问题三: 数据处理
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 读取附件二
xls_two = pd.ExcelFile("附件2(Attachment 2)2022Problem B.xlsx")

# 表中的各个sheet数据
temp_data = pd.read_excel(xls_two,'温度(temperature)')
quality_data = pd.read_excel(xls_two,'产品质量(quality of the products)')
mineral_data = pd.read_excel(xls_two,'原矿参数(mineral parameter)')
process_data = pd.read_excel(xls_two,'过程数据(process parameter)')
process_data["时间 (Time)"] = process_data["时间 (Time)"].dt.floor('T')

# 缺失值检测,查看数据缺失值
nan_pre = temp_data[temp_data.isnull().T.any()]
nan_pre.shape
```

```
# 查找时间序列缺失值
def nan_find(data, interval,timeName):
    data = data.copy()
    nan_pre = data[data.isnull().T.any()]
    print("初始数据shape:",data.shape)
    print("初始中缺失值",nan_pre.shape[0])
    # 进行时间补全
    data[timeName] = data[timeName].astype("datetime64")
    data_range =
        pd.date_range(start=data[timeName].min(),end=temp_data[timeName].max(),freq=interval)
    data_after = data.set_index(timeName).reindex(index = data_range)
    print("补全后的shape:",data_after.shape)
    nan_aft = data_after[data_after.isnull().T.any()]
    print("补全后的缺失值:",nan_aft.shape[0])
    print("-------------------------")
    print("时间缺失值:",nan_aft.shape[0] - nan_pre.shape[0])
    # 返回补全时间后的值
    return data_after

# temperature
temp_data_check = nan_find(temp_data, "T", "时间 (Time)")

# 数据处理
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 机器学习
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score,roc_curve,auc
from sklearn import metrics
from sklearn.model_selection import train_test_split

# 贝叶斯调参
from bayes_opt import BayesianOptimization
from sklearn.model_selection import cross_val_score

# smote过采样
from imblearn.over_sampling import SMOTEN

# 保存机器学习模型
import joblib
```

```
data_three = pd.read_excel("forML_Q3.xlsx",index_col=0)
data_three.head(2)

data_three_af = data_three.iloc[:,[0,1,2,3,4,5,8,9,10]].copy()

# 进行bayes调参
def xgb_cv(max_depth, learning_rate, n_estimators, min_child_weight, subsample,
     colsample_bytree, reg_alpha, gamma):
    val = cross_val_score(estimator=XGBClassifier(max_depth=int(max_depth),
                                          learning_rate=learning_rate,
                                          n_estimators=int(n_estimators),
                                          min_child_weight=min_child_weight,
                                          subsample=max(min(subsample, 1), 0),
                                          colsample_bytree=max(min(colsample_bytree, 1), 0),
                                          reg_alpha=max(reg_alpha, 0), gamma=gamma,
                                          booster='gbtree',
                                          seed=42), X=data_three.drop(['是否合格'],axis=1),
                                              y=data_three['是否合格'], scoring="roc_auc",
                        cv=10).mean()
    return val

xgb_bo = BayesianOptimization(xgb_cv, pbounds={'max_depth': (1, 10),
                                       'learning_rate': (0.01, 0.3),
                                       'n_estimators': (1, 200),
                                       'min_child_weight': (0, 20),
                                       'subsample': (0.001, 1),
                                       'colsample_bytree': (0.01, 1),
                                       'reg_alpha': (0.001, 20),
                                       'gamma': (0.001, 10)})
xgb_bo.maximize(n_iter=100, init_points=10)

xgb_bo.max

# 过采样准备
sm = SMOTEN(random_state=0)



X_train, X_test, y_train, y_test = train_test_split(data_three_af.drop(['是否合格'],axis=1),
                                          data_three_af['是否合格'],
                                          random_state=5, test_size=0.2)

X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
xgbtest = XGBClassifier(
    seed=42,
    learning_rate = 0.21,
    max_depth = 3,
```

```
    reg_alpha = 1,
    n_estimators = 100,
    gamma = 0.13,
    min_child_weight = 12,
    subsample = 0.83,
    reg_lambda = 1,
    eval_metric='logloss'
)

xgbtest.fit(X_train_res, y_train_res)
# xgbtest.fit(X_train, y_train)
y_pred = xgbtest.predict(X_test)
y_pred_proba = xgbtest.predict_proba(X_test)
print(classification_report(y_test, y_pred))
fpr_xgb, tpr_xgb, thresholds = metrics.roc_curve(y_test,y_pred_proba[:,1], pos_label=1)
print(auc(fpr_xgb, tpr_xgb))
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)

joblib.dump(xgbtest, "xgb2.joblib.dat")

a = pd.Series(y_train)
df=pd.concat([X_train,a],axis=1,ignore_index=True)
df.to_excel("过采样前.xlsx")

a = pd.Series(y_train_res)
df=pd.concat([X_train_res,a],axis=1,ignore_index=True)
df.to_excel("过采样结果.xlsx")

# catboost
cat = CatBoostClassifier(n_estimators=10)
cat.fit(X_train_res, y_train_res)
y_pred_cat = cat.predict(X_test)
print(classification_report(y_test, y_pred_cat))

# randomForestClassifier
rf = RandomForestClassifier(n_estimators=10)
rf.fit(X_train_res, y_train_res)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))

# cat_pre
y_pred_proba_cat = cat.predict_proba(X_test)
fpr_cat, tpr_cat, thresholds = metrics.roc_curve(y_test,y_pred_proba_cat[:,1], pos_label=1)
roc_auc_cat = auc(fpr_cat, tpr_cat)

# rf_pre
y_pred_proba_rf = rf.predict_proba(X_test)
```

```
fpr_rf, tpr_rf, thresholds = metrics.roc_curve(y_test,y_pred_proba_rf[:,1], pos_label=1)
roc_auc_rf = auc(fpr_rf, tpr_rf)

plt.style.use('seaborn-darkgrid')
# 4.绘图
plt.figure(dpi=1000)
lw = 2
plt.plot(
    fpr_xgb, tpr_xgb,
    color="blue",
    lw=lw,
    label="XgBoost (area = %0.3f)" % roc_auc_xgb
)
plt.plot(
    fpr_cat, tpr_cat,
    color="darkorange",
    lw=lw,
    label="CatBoost (area = %0.3f)" % roc_auc_cat,
)
plt.plot(
    fpr_rf, tpr_rf,
    color="green",
    lw=lw,
    label="RandomForest (area = %0.3f)" % roc_auc_rf
)

plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([-0.05, 1.0])
plt.ylim([-0.05, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
# plt.savefig('auc_roc.pdf')
plt.show()

X_train_res.head()

# 预测结果
pre_data = pd.read_excel("p3_predect.xlsx",index_col=0)
pre_data_for = pre_data.iloc[:,[0,1,2,3,4,5,8,9]]
pre_data_one = pre_data_for['2022-04-08'].iloc[121:,:]
pre_data_two = pre_data_for['2022-04-09'].iloc[121:,:]
# pre_data = pd.read_excel("p3_predect.xlsx")
# pre_data_for = pre_data.iloc[:,[1,2,3,4,5,6,9,10]]

pre_res_one = xgbtest.predict_proba(pre_data_one)
```

```
pre_res_two = xgbtest.predict_proba(pre_data_two)

pre_res_one[:,1].mean()
pre_res_two[:,1].mean()



# 问题四
# 数据处理
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import joblib

# 智能算法
from mealpy.human_based.ICA import BaseICA



import warnings
warnings.filterwarnings('ignore')

# 导入机器学习的数据结构
xgbtest = joblib.load("xgb.joblib.dat")

data_pre = pd.read_excel("p4_predict.xlsx",index_col=0)
# 分成两天
data_pre_10 = data_pre['2022-04-10'].iloc[:,[0,1,2,3,6,7]].copy()
data_pre_11 = data_pre['2022-04-11'].iloc[:,[0,1,2,3,6,7]].copy()

# 适应度函数/目标函数
def fitness_function(solution):
    data_pre_11["系统I温度"] = solution[0]
    data_pre_11["系统II温度"] = solution[1]

    pre_xgb = xgbtest.predict_proba(data_pre_11)

    prob = pre_xgb[:,1].mean()

    def g1():
        if prob < 0.80:
            return (0.80- prob)
        return 0

    def violate(value):
        if value > 0:
            return 1e8*value
        else:
            return 0
```

```
    fx = np.absolute(solution.sum() - 1010.32 - 874.47)

    return fx + violate(g1())

epoch = 200
pop_size = 45
empire_count = 4
assimilation_coeff = 2.0
revolution_prob = 0.05
revolution_rate = 0.05
revolution_step_size = 0.15
zeta = 0.2


problem = {
    "fit_func": fitness_function,
    "lb": [0, 0],
    "ub": [3000, 3000],
    "minmax": "min",
}

model = BaseICA(problem, epoch, pop_size, empire_count, assimilation_coeff, revolution_prob,
    revolution_rate, revolution_step_size, zeta)
best_position, best_fitness = model.solve()
print(f"Solution: {best_position}, Fitness: {best_fitness}")

model.history.save_global_best_fitness_chart(filename="problem4/first")

data_pre_p3 = pd.read_excel("p3_predect.xlsx",index_col=0)
data_pre_p3 = data_pre_p3['2022-04-09'].iloc[:,[0,1,2,3,4,5,8,9]]
data_pre_p3.head()

st_res = np.zeros((12,2))
fit_res = np.zeros(12)

for i in range(12):
    # 适应度函数/目标函数
    h_rate = 0.15
    def fitness_function(solution):
        data_pre_p3["系统I温度"] = solution[0]
        data_pre_p3["系统II温度"] = solution[1]

        pre_xgb = xgbtest.predict_proba(data_pre_p3)

        prob = pre_xgb[:,1].mean()
```

```
    def g1():
        if prob < (h_rate + (i)*0.05):
            return ((h_rate + (i+1)*0.05)- prob)
        return 0

    def violate(value):
        if value > 0:
            return 1e8*value
        else:
            return 0

    fx = np.absolute(solution[0] - 1010.32) + np.absolute(solution[1] - 874.47)

    return fx + violate(g1())

epoch = 100
pop_size = 80
empire_count = 6
assimilation_coeff = 2.0
revolution_prob = 0.05
revolution_rate = 0.05
revolution_step_size = 0.15
zeta = 0.2


problem = {
    "fit_func": fitness_function,
    "lb": [0, 0],
    "ub": [2000, 2000],
    "minmax": "min",
}

model = BaseICA(problem, epoch, pop_size, empire_count, assimilation_coeff,
     revolution_prob, revolution_rate, revolution_step_size, zeta)
best_position, best_fitness = model.solve()
st_res[i,:] = best_position
fit_res[i] = best_fitness
print(f"Solution: {best_position}, Fitness: {best_fitness}")
```

# 附录 D　Matlab 源程序

```
%% 第一问%%

clc;clear
load data_tem
load data_tem1

tem=data_tem1;

%小波降噪
x1= tem(:,1);
x2= tem(:,2);
[xy1,xr1,xg1]=xb(x1);
[xy2,xr2,xg2]=xb(x2);
close%关闭窗口




tem1=xy1';%硬阈值去噪更好

tem2=xg2';%固定阈值去噪
tem=[tem1,tem2];%组合

%按天数划分


for i=1:10

   day1(:,i)=tem1([1440*i-1439:1440*i]);
   day2(:,i)=tem2([1440*i-1439:1440*i]);

end



for j=1:10

%对每一天进行分析，利用dbscan的阈值进行调温点的确定
 for i=1:1439

 if abs(day1(i,j)-day1(i+1,j))<6&&abs(day2(i,j)-day2(i+1,j))<6
    sw(i,j)=1;
 else
    sw(i,j)=0;
```

```
 end
 end
end


k=1;
t=zeros(30,10);%对持续稳定的温度段进行计数
for j=1:10
   for i=1:1439
      if sw(i,j)==1
        t(k,j)=t(k,j)+1;
      else
        k=k+1;%不稳定，换行
        t(k,j)=t(k,j)+1;
      end
   end
   k=1;
end




for j=1:10
   for i=1:30
   if i==1
       t(i,j)=t(i,j)+1;
   end

   if t(i,j)<120
     a(i,j)=0;
   else
     a(i,j)=1;
   end
   end

end

c1=0
c=zeros(1441,10);
for j=1:10

 for i=1:30


   b=repmat(a(i,j),t(i,j),1);
   c1=[c1;b];
```

```
 end
 c(:,j)=c1;
 c1=0;
end




c=c(2:1441,1:10);%去掉第一个空值
c=c(:) ;
c=c(2:14400);


C=c;%两小时后加工好




A=0;

for i=1:14399

   if C(i)==0
       A=0;
   end

    if C(i)==1

       A=A+1;
    end

    if A==120

    C(i-119:i)=0;

    end



end

%%小波分析代码
```

```
function [xy,xr,xg]=xb(YSJ)
t1=clock;

[c,l]=size(YSJ);
Y=[];
for i=1:c
   Y=[Y,YSJ(i,:)];
end
[c1,LL]=size(Y);
X=[1:LL];
 %% 绘制噪声信号图像
figure(1);
subplot(2,2,1)
plot(X,Y);
xlabel('横坐标');
ylabel('纵坐标');
title('原始信号');
 %% 硬阈值处理
lev=3;
xy=wden(Y,'heursure','h','one',lev,'db4');%硬阈值去噪处理后的信号序列

subplot(2,2,2)
plot(X,xy)
xlabel('横坐标');
ylabel('纵坐标');
title('硬阈值去噪处理')
set(gcf,'Color',[1 1 1])
%% 软阈值处理
lev=3;
xr=wden(Y,'heursure','s','one',lev,'db4');%软阈值去噪处理后的信号序列

subplot(2,2,3)
plot(X,xr)
xlabel('横坐标');
ylabel('纵坐标');
title('软阈值去噪处理')
set(gcf,'Color',[1 1 1])
%% 固定阈值后的去噪处理
lev=3;
xg=wden(Y,'sqtwolog','s','sln',lev,'db4');%固定阈值去噪处理后的信号序列

subplot(2,2,4)
plot(X,xg);
xlabel('横坐标');
ylabel('纵坐标');
title('固定阈值后的去噪处理')
set(gcf,'Color',[1 1 1])
```

```
%% 计算信噪比SNR
Psig=sum(Y*Y')/LL;
Pnoi1=sum((Y-xy)*(Y-xy)')/LL;
Pnoi2=sum((Y-xr)*(Y-xr)')/LL;
Pnoi3=sum((Y-xg)*(Y-xg)')/LL;
SNR1=10*log10(Psig/Pnoi1);
SNR2=10*log10(Psig/Pnoi2);
SNR3=10*log10(Psig/Pnoi3);
%% 计算均方根误差RMSE
RMSE1=sqrt(Pnoi1);
RMSE2=sqrt(Pnoi2);
RMSE3=sqrt(Pnoi3);
%% 输出结果
disp('-------------三种阈值设定方式的降噪处理结果---------------');
disp(['xy硬阈值去噪处理的SNR=',num2str(SNR1),', RMSE=',num2str(RMSE1)]);
disp(['xr软阈值去噪处理的SNR=',num2str(SNR2),', RMSE=',num2str(RMSE2)]);
disp(['xg固定阈值后的去噪处理SNR=',num2str(SNR3),', RMSE=',num2str(RMSE3)]);
t2=clock;
tim=etime(t2,t1);
disp(['------------------运行耗时',num2str(tim),'秒------------------'])

end



%%开始聚类
clc;
clear;
close all;
load X


%% 运行 DBSCAN Clustering Algorithm
epsilon= 0.8 ;
MinPts= 10  ;
IDX1=DBSCAN(X,epsilon,MinPts);
%% Plot Results
figure;
PlotClusterinResult(X, IDX1);
title(['DBSCAN 聚类' ]);
set(gcf,'position',[30 -10 500 500]);
xlabel('系统一的温度变化');

ylabel('系统二的温度变化');

epsilon= 0.25 ;
MinPts= 3  ;
```

```
IDX2=DBSCAN(X,epsilon,MinPts);
%% 画出结果%%
figure;
PlotClusterinResult(X, IDX2);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = ' num2str(MinPts) ')']);
set(gcf,'position',[530 -10 500 500]);

epsilon= 0.5 ;
MinPts= 3  ;
IDX3=DBSCAN(X,epsilon,MinPts);
%% 画图%%
figure;
PlotClusterinResult(X, IDX3);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = ' num2str(MinPts) ')']);
set(gcf,'position',[30 380 500 500]);


%%聚类代码


function [IDX, isnoise]=DBSCAN(X,epsilon,MinPts)

    C=0;

    n=size(X,1);
    IDX=zeros(n,1);

    D=pdist2(X,X);

    visited=false(n,1);
    isnoise=false(n,1);

    for i=1:n
        if ~visited(i)
            visited(i)=true;

            Neighbors=RegionQuery(i);
            if numel(Neighbors)<MinPts
                % X(i,:) is NOISE
                isnoise(i)=true;
            else
                C=C+1;
                ExpandCluster(i,Neighbors,C);
            end

        end
```

```
    end

    function ExpandCluster(i,Neighbors,C)
        IDX(i)=C;

        k = 1;
        while true
            j = Neighbors(k);

            if ~visited(j)
                visited(j)=true;
                Neighbors2=RegionQuery(j);
                if numel(Neighbors2)>=MinPts
                    Neighbors=[Neighbors Neighbors2]; %#ok
                end
            end
            if IDX(j)==0
                IDX(j)=C;
            end

            k = k + 1;
            if k > numel(Neighbors)
                break;
            end
        end
    end

    function Neighbors=RegionQuery(i)
        Neighbors=find(D(i,:)<=epsilon);
    end

end



function PlotClusterinResult(X, IDX)

    k=max(IDX);

    Colors=hsv(k);

    Legends = {};
    for i=0:k
        Xi=X(IDX==i,:);
        if i~=0
            Style = 'x';
            MarkerSize = 8;
```

```
            Color = Colors(i,:);
            Legends{end+1} = ['Cluster #' num2str(i)];
        else
            Style = 'o';
            MarkerSize = 6;
            Color = [0 0 0];
            if ~isempty(Xi)
                Legends{end+1} = 'Noise';
            end
        end
        if ~isempty(Xi)
            plot(Xi(:,1),Xi(:,2),Style,'MarkerSize',MarkerSize,'Color',Color,'LineWidth',2);
        end
        hold on;
    end
    hold off;
    axis equal;
    grid on;
    legend(Legends);
    legend('Location', 'NorthEastOutside');

end



%%第三问代码

clc;clear
load data_2

tem=data2;

%小波降噪
x1= tem(:,1);
x2= tem(:,2);
[xy1,xr1,xg1]=xb(x1);
[xy2,xr2,xg2]=xb(x2);
close%关闭窗口




tem1=xy1';%硬阈值去噪更好

tem2=xg2';%固定阈值去噪
tem=[tem1,tem2];%组合
```

```
%按天数划分
t=[0,0];
tem=[t;tem];
tem1=tem(:,1);
tem2=tem(:,2);


for i=1:73

    day1(:,i)=tem1([1440*i-1439:1440*i]);
    day2(:,i)=tem2([1440*i-1439:1440*i]);

end



for j=1:73

%对每一天进行分析，利用dbscan的阈值进行调温点的确定
  for i=1:1439

  if abs(day1(i,j)-day1(i+1,j))<6&&abs(day2(i,j)-day2(i+1,j))<6
      sw(i,j)=1;
  else
      sw(i,j)=0;
  end
  end
end


k=1;
t=zeros(201,73);%对持续稳定的温度段进行计数
for j=1:73
    for i=1:1439
       if sw(i,j)==1
         t(k,j)=t(k,j)+1;
       else
         k=k+1;%不稳定，换行
         t(k,j)=t(k,j)+1;
       end
    end
    k=1;
end



%判断是否大于120m
```

```
for j=1:73
    for i=1:201
    if i==1
        t(i,j)=t(i,j)+1;
    end

    if t(i,j)<120
      a(i,j)=0;
    else
      a(i,j)=1;
    end
    end

end

c1=0
c=zeros(1441,73);
for j=1:73

 for i=1:201


    b=repmat(a(i,j),t(i,j),1);
    c1=[c1;b];




 end
 c(:,j)=c1;
 c1=0;
end




c=c(2:1441,1:73);%去掉第一个空值
c=c(:) ;
c=c(2:end);


C=c;%两小时后加工好
```

```
A=0;

for i=1:105119

   if C(i)==0
       A=0;
   end

    if C(i)==1

       A=A+1;
    end

    if A==120

    C(i-119:i)=0;

    end



end


%% 对指标计算
load zhibiao
zb=zhibiao;
for i=1:1744
  if zb(i,1)<=80.33&&zb(i,1)>=77.78&&zb(i,2)<24.15&&zb(i,3)<17.15&&zb(i,4)<15.62
     h(i,1)=1 ;%合格
  else
     h(i,1)=0;%不合格
  end

end

%%开始聚类
clc;
clear;
close all;
load Y


X(:,1)=Y(1:end/6,2);
X(:,2)=Y(1:end/6,1);
%% 运行 DBSCAN Clustering Algorithm
epsilon= 0.8 ;
```

```
MinPts= 10  ;
IDX1=DBSCAN(X,epsilon,MinPts);
%% Plot Results
figure;
PlotClusterinResult(X, IDX1);
title(['DBSCAN 聚类' ]);
set(gcf,'position',[30 -10 500 500]);
xlabel('系统一的温度变化');

ylabel('系统二的温度变化');

epsilon= 0.25 ;
MinPts= 3  ;
IDX2=DBSCAN(X,epsilon,MinPts);
%% 画出结果%%
figure;
PlotClusterinResult(X, IDX2);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = ' num2str(MinPts) ')']);
set(gcf,'position',[530 -10 500 500]);

epsilon= 0.5 ;
MinPts= 3  ;
IDX3=DBSCAN(X,epsilon,MinPts);
%% 画图%%
figure;
PlotClusterinResult(X, IDX3);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ', MinPts = ' num2str(MinPts) ')']);
set(gcf,'position',[30 380 500 500]);
```