


中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生数学建模竞赛

学　　校	同济大学
参赛队号	**0102470089**
队员姓名	**1.**杨柏萃
	2.刘宇航
	3.李嘉伟

中国研究生创新实践系列大赛
“华为杯”第十七届中国研究生
数学建模竞赛

题 目 汽油辛烷值优化建模

摘 要：

汽油辛烷值含量对汽油的燃烧性能具有重要的影响。在保证硫含量较低的情况下，应尽量降低辛烷值在汽油精炼过程中的损失。本文采用数据挖掘技术，研究了辛烷值优化建模等问题，具有一定的现实意义。

针对问题一，首先参考附件一中的预处理结果，按照附件二中的数据处理要求对 285 和 313 号样本的原始数据进行遍历，检验其是否存在如数据值缺失、超出操作变量范围和不满足拉依达准则等异常值并剔除。之后判断每位点的数据缺失值比率是否超出设定的阈值，根据判断结果采取删除位点或对空缺值进行拉格朗日插值操作。最后对预处理过的数据组取平均值，获得 285 和 313 号样本的数据处理结果。两样本共检测出异常数据 477 个，按照步骤进行预处理之后的结果与附件一已给数据对比，发现仅存在微小差异，原因可能是选择的处理标准略有不同。

针对问题二，为从 367 个变量中选取建模的主要变量，需要进行降维处理。首先对附件一的预处理数据进行分析，剔除掉存在大量异常数据和物理意义不合理的位点。然后分析了各变量之间的相关性，得出各变量之间具有较强的相关性。为较好地筛选主要变量，采用机器学习算法中适用于处理非线性关系的随机森林对样本数据进行分析，得出了影响产品辛烷值的各自变量的重要性排序。初步选取了排序前 60 的自变量。考虑到某些自变量之间也具有高度的相关性，为保证选取了变量具有独立性，采用距离相关系数对 60 个变量进行分析，逐步剔除自变量之间相关性较高的变量。最终选取 27 个主要变量。再次对选取的 27 变量计算两两相关性，结果表明，选取的变量之间关系较弱，具有很好的独立性。同时，选取的 27 个变量在物理意义上具有很好的可解释性，说明 27 个主要变量的选取是合理的。

针对问题三，为建立辛烷值损失预测模型，需要准确地预测产品中的辛烷值。由于上述选取的主要变量和产品辛烷值之间具有较强的非线性关系，一般的线性模型并不完全适用。考虑到神经网络具有很强的非线性映射能力，且精度很高，本文建立了三层 BP 神经网络对产品辛烷值进行预测。输入层具有 27 个神经元，输出层具有 1 个神经元。选取 80%的样本作为训练集，20%的样本作为测试集对网络进行训练。经过反复实验得出隐含层神经元的最佳个数为 50。结果显示，建立的预测模型的均方误差（MSE）为 0.0146，误差百分比最大仅为 0.25%，具有较高的精度，能够精确地预测产品中辛烷值含量。由于原料中

辛烷值含量已知，于是可以精确地预测辛烷值损失。

针对问题四，为分析辛烷值优化后损失降幅大于 30%的样本，并得出主要变量的操作条件，建立了优化模型。以辛烷值损失最小为目标函数，选取 23 个可变的操作变量为决策变量.由于操作变量具有取值范围，因此决策变量存在约束。同时考虑到实际的辛烷值不能超过原料中的辛烷值，硫含量的范围在 0—5 微克/克，建立了综合的约束条件。针对这一目标优化的问题，考虑到其搜索空间大，全局最优解寻找难度大的特点，采用了粒子群算法对这一优化问题进行求解。通过 Python 编写程序，求解得出：在 325 个样本中，有 103 个样本经过优化后其辛烷值损失降幅大于 30%，硫含量均低于 5 微克/克，同时获得了 23 个操作变量的操作条件，说明建立的模型是合理有效的。

针对问题五，首先根据问题四中得出的优化模型计算 133 号样本对应的最优操作条件。以各变量达到最优操作条件为目标，使各个操作变量自初始值按照各自调整步长 Δ 向最优条件逐步前进。在每一步前进的状态点，使用建立的辛烷值和硫含量的预测模型预测此时的辛烷值和硫的含量，并记录。根据记录的辛烷值和硫含量的数据即可绘制随调整次数变化的辛烷值和硫含量的轨迹。绘制的轨迹表明，133 号样本的产品辛烷值的优化结果为 88.48，辛烷值损失降低幅度达到 30.54%，产品硫含量最终低于 5 微克/克，效果较好。此外，分析了产品中辛烷值含量和硫含量的变化趋势，其中产品辛烷值的轨迹逐渐提高符合预期，硫含量的变化趋势呈现出一定的随机性，对此现象作出了合理性的解释。

关键词：汽油辛烷值，相关性分析，随机森林，BP 神经网络，粒子群算法

目录

1 问题重述 0
1.1 问题背景 0
1.2 问题重述 1
2 模型假设 1
3 符号说明 2
4 问题一：数据处理 2
4.1 问题一分析 2
4.2 数据处理 4
4.2.1 异常数据的剔除 4
4.2.2 缺失数据的处理 6
4.2.3 样本确定 7
5 问题二：寻找建模主要变量 8
5.1 问题二分析 8
5.2 变量降维 9
5.2.1 剔除含有过多异常值的变量 9
5.2.2 随机森林算法——评价变量重要度 9
5.2.3 变量高相关性滤波算法——去耦合 11
5.3 变量选择合理性评价 14
6 问题三：辛烷值（RON）损失预测模型 16
6.1 问题三分析 16
6.2 辛烷值损失预测模型的建立 17
6.2.1 BP 神经网络基本原理 17
6.2.2 辛烷值损失预测模型建立 21
6.2.3 模型求解结果 22
6.2.4 硫含量预测模型建立及求解 23
6.3 模型验证 23
6.3.1 辛烷值损失预测模型验证 23
6.3.2 硫含量预测模型验证 24
7 问题四：主要变量操作方案的优化 25
7.1 问题四分析 25
7.2 主要变量操作方案的优化模型建立 26
7.2.1 粒子群算法 26
7.2.2 优化目标及约束设定 29
7.2.3 模型参数设定 30

7.2.4 模型求解 31
7.3 模型评价（结果分析） 32
8 问题五：模型的可视化展示 32
8.1 问题五分析 32
8.2 建模结果 33
8.3 结果分析 35
9 模型评价与改进 36
9.1 模型优点 36
9.2 模型缺点 36
9.3 模型的改进与推广 36
10 参考文献 0
11 附录 0

1 问题重述

1.1 问题背景

汽油是小型车辆的主要燃料，汽油燃烧产生的尾气排放对大气环境有重要影响[1]。为控制汽车尾气排放造成的大气污染，世界各国都制定了日益严格的汽油质量标准（见表1.1）。我国在2014年全面实施国IV汽油标准的基础上，将国V汽油标准的执行时间由原定的2018年1月1日提前至2017年1月1日，并将分别于2019年1月1日和2023年1月1日起执行国VI-A和国VI-B汽油质量标准[2]。从我国已制定的各阶段汽油质量标准与欧洲标准的对比可以看出，汽油质量升级的趋势是在控制硫含量≤10mg/kg 的同时不断降低烯烃和芳烃含量，另要尽量保持其辛烷值[3]。

表1.1 欧盟和我国车用汽油主要规格

车用汽油标准	辛烷值	硫含量/(μg/g)≯	苯含量/%≯	芳烃含量/%≯	烯烃含量/%≯
国III（2010年）	90-97	150	1	40	30
国IV（2014年）	90-97	50	1	40	28
国V（2017年）	89-95	10	1	40	24
国VI-A（2019年）	89-95	10	0.8	35	18
国VI-B（2023年）	89-95	10	0.8	35	15
欧V（2009年）	95	10	1	35	18
欧VI（2013年）	95	10	1	35	18
世界燃油规范（V类汽油）	95	10	1	35	10

我国目前石油业面临的问题是：石油供求失衡不断扩大，对石油进口的依赖越来越严重[4]，原油对外依存度超过70%，且大部分是中东地区的含硫和高硫原油。原油中的重油通常占比40-60%，这部分重油（以硫为代表的杂质含量也高）难以直接利用。为了有效利用重油资源，我国大力发展了以催化裂化为核心的重油轻质化工艺技术，将重油转化为汽油、柴油和低碳烯烃，超过70%的汽油是由催化裂化生产得到，因此成品汽油中95%以上的硫和烯烃来自催化裂化汽油。故必须对催化裂化汽油进行精制处理，以满足对汽油质量要求。

辛烷值（以 RON 表示）是反映汽油燃烧性能的最重要指标，并作为汽油的商品牌号（例如 89#、92#、95#）。现有技术在对催化裂化汽油进行脱硫和降烯烃过程中，普遍降低了汽油辛烷值。若能建立反应其化工过程的数学模型，对操作变量采取研究，优化过程响应，使其RON损失降低，可给企业带来巨大的经济效益。

1.2 问题重述

基于上述研究背景，本文需研究和解决以下问题：

问题一：数据处理

参考近 4 年的工业数据(见附件一“325 个数据样本数据.xlsx”)的预处理结果，依“样本确定方法”（附件二）对 285 号和 313 号数据样本进行预处理（原始数据见附件三“285 号和 313 号样本原始数据.xlsx”）并将处理后的数据分别加入到附件一中相应的样本号中，供下面研究使用。

问题二：寻找建模主要变量

根据预处理过的 325 个样本数据（见附件一），通过降维的方法，从 7 个原料性质、2 个待生吸附剂性质、2 个再生吸附剂性质、2 个产品性质等变量以及另外 354 个操作变量（共计 367 个变量）筛选出影响建立降低辛烷值损失模型的主要变量与因素（为了工程应用方便，建议降维后的主要变量在 30 个以下，并考虑将原料的辛烷值作为建模变量之一）。需尽可能使选取的建模变量具有代表性、独立性，并详细说明建模主要变量的筛选过程及其合理性。

问题三：建立辛烷值（RON）损失预测模型

采用上述样本和建模主要变量，通过数据挖掘技术建立辛烷值（RON）损失预测模型，并进行模型验证。

问题四：主要变量操作方案的优化

在每个样本原料、待生吸附剂、再生吸附剂的性质不变（以它们在样本中的数据为准）的条件下，对操作变量进行优化调整。给出在保证产品硫含量不大于 5 μg/g 的前提下，利用建立的 RON 损失预测模型获得 325 个数据样本中，辛烷值（RON）损失降幅大于 30%的样本对应的主要变量优化后的操作条件。

问题五：模型的可视化展示

工业装置为了平稳生产，优化后的主要操作变量往往只能逐步调整到位（各主要操作变量每次允许调整幅度值 Δ 见附件四“354 个操作变量信息.xlsx”）。在原料性质、待生吸附剂和再生吸附剂的性质数据保持不变的情况下，利用图形展示 133 号样本的主要操作变量优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹。

2 模型假设

假设 1：所有样本的原料性质、操作位点的数据测量正确、数据处理步骤正确；

假设 2：所有位点均是可以操作改变的控制变量；

假设 3：在优化主要变量时认为所提出的预测模型结果准确。

3 符号说明

序号	符号	含义
1	T_i	第 i 组的数据采集时间
2	$variable_i$	第 j 个变量的操作范围
3	v_h	剩余误差
4	$Missing\_per$	数据缺失比例
5	K	决策树个数
6	M	特征总数
7	n_k	某一节点 k 的重要性
8	f_i	第 i 个特征的重要性
9	f_{ni}	第 i 个特征的归一化重要性
10	$R^2(x,y)$	样本 x 与 y 间的距离相关系数
11	W	神经元网络层全职矩阵
12	E	神经元误差函数
13	x_i^t	粒子 i 在 t 时刻的位置
14	l_d	粒子搜索空间的下限
15	u_d	粒子搜索空间的上限
16	c_1、c_2	粒子群算法学习因子
17	ω	粒子群算法惯性权重
18	RON_{pred}	产品辛烷值含量的预测值
19	S_{pred}	产品硫含量的预测值

4 问题一：数据处理

4.1 问题一分析

根据问题一要求，由于采集到的原始数据中部分位点存在问题，为保证数据挖掘结果，须按附件二“样本确定方法.doc”对原始数据进行处理。总体的数据处理方法要求如下：

（1）对于只含有部分时间点的位点，如果其残缺数据较多，无法补充，将此类位点删除；

（2）删除 325 个样本中数据全部为空值的位点；

（3）对于部分数据为空值的位点，空值处用其前后两个小时数据的平均值代替；

（4）根据工艺要求与操作经验，总结出原始数据变量的操作范围，然后采用最大最小的限幅方法剔除一部分不在此范围的样本；

（5）根据拉依达准则（3σ 准则）去除异常值。

题目给出了 285 号和 313 号样本所采集的原始数据，包括其各自的 7 个原料性质、2 个待生吸附剂性质、2 个再生吸附剂性质、2 个产品性质等变量以及另外 354 个操作变量的原始数据。本题需按附件二的数据处理方法对两样本中 354 个操作变量的数据进行数据清洗，并将处理后的数据分别加入到附件一中相应的样本号中。

问题一数据处理的思路流程图如图 4.1 **错误!未找到引用源。**所示。首先按照数据处理方法的要求对初始数据集进行初步分析，判断其是否存在不良数据，将不良数据按照标准剔除，得到数据集 1。之后判断数据集 1 中的缺失值是否满足插值条件，对满足条件的缺失值做拉格朗日插值处理，得到数据集 2。然后根据拉依达准则对含有粗大误差值的坏值进行剔除得到数据集 3，其次根据以辛烷值数据测定的时间点为基准时间，取其前 2 个小时的操作变量数据的平均值作为对应辛烷值的操作变量数据的样本确定方法，对数据集 3 中的数据进行整合计算，得到 285 和 313 号样本的数据预处理结果。最后将预处理得到的结果填入附件一“325 个数据样本数据.xlsx”中。

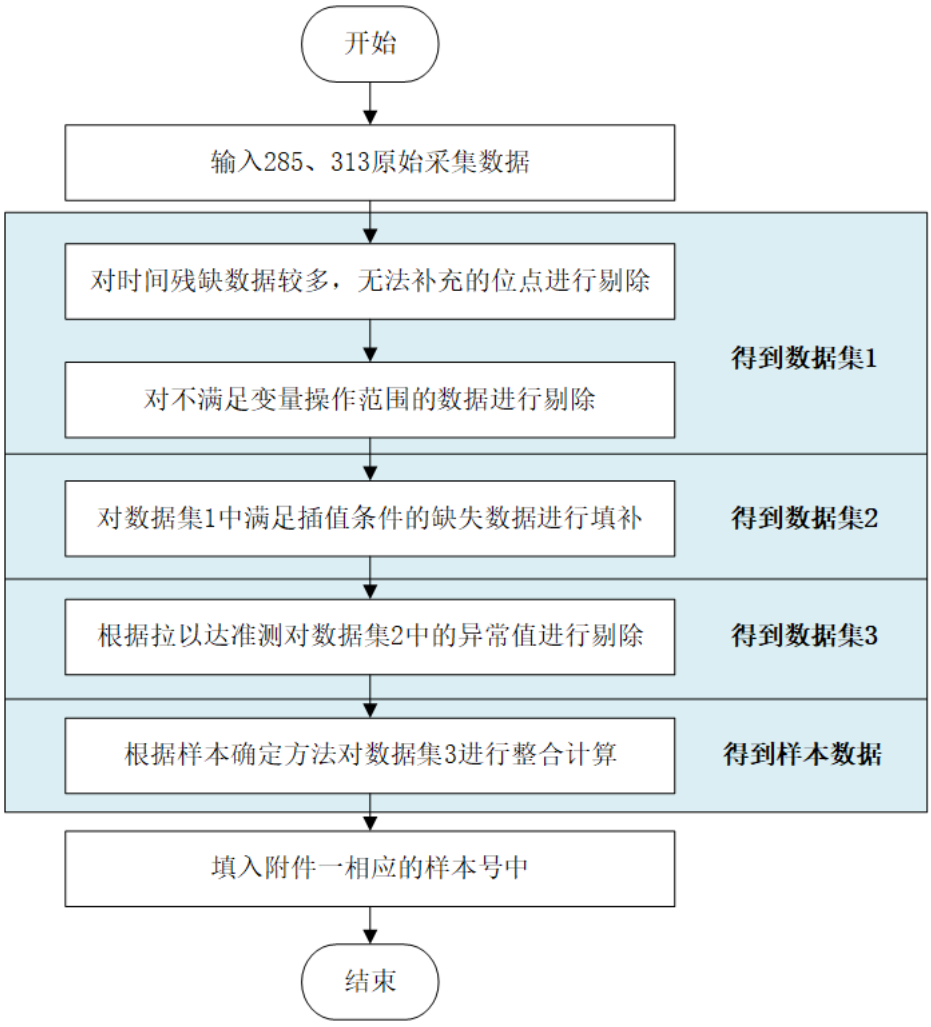

图 4.1 问题一思路流程图

4.2 数据处理

原始数据采自中石化高桥石化实时数据库（霍尼韦尔 PHD）及 LIMS 实验数据库。其中原料、产品和催化剂数据来自于 LIMS 实验数据库，操作变量数据来自于实时数据库，采集操作位点数共 354 个。各数据的时间范围与采集频次如表 4.1 所示。

表 4.1 各数据的采集时间范围及采集频次

数据类别	数据采集时间范围	数据采集频次
操作变量(354 个)	2017.04-2019.09	3 min /1 次
	2019.10-2020.05	6 min /1 次
原料及产品的辛烷值	2017.04-2020.05	1 week /2 次

根据附件三“285 号和 313 号样本原始数据”可知，两样本中的主要数据如表 4.2 所示。

表 4.2 285 号和 313 号样本主要原始数据

类别	285 号	313 号
采样日期	2017.07.17	2017.05.15
操作变量采样时段	6:00:00-8:00:00	6:00:00-8:00:00
辛烷值采样时间点	8:00:00	8:00:00
操作变量数据采集量	14160(40x354)	14160(40x354)

4.2.1 异常数据的剔除

1）时间不连续数据

因 285 号和 313 号的采样日期处于 2017.04-2019.09 时间范围内，故其数据采集频次应为 3min/次。编历两样本原始数据，检测第 i 组和第 $i+1$ 组数据采集时间 T 的差值，若存在：

$$\left|T_{i+1}-T_i\right|>3\,\text{min}$$

则认为该时间点数据缺失。之后判断时间段缺失程度，若其残缺数据较多，无法补充，则将此类位点删除。若只有小段数据缺失，可将此部分数据标为 NaN，之后按后一节的插值方法进行插值处理。

经检查，附件三中不存在因时间采集时间不连续导致的缺失数据。

2）超范围数据

此部分数据处理工作需根据工艺要求与操作经验总结出原始数据变量的操作范围，然后采用最大最小的限幅方法剔除一部分不在此范围的样本。

首先依据附件四“354 个操作变量信息”确定第 j 个数据变量 $variable_j$ 对应操作范围的最小值 $variable_{j\min}$ 及最大值 $variable_{j\max}$。向 Matlab 中导入附件三中的 285 号和 313 号样

本的各原始数据 $x_{i,j}$，判断其是否超出操作变量范围，若存在：

$$x_{i,j} < variable_{j\min} \quad \text{or} \quad x_{i,j} > variable_{j\max}$$

则认为该 $x_{i,j}$ 数据不合理，需对其进行剔除，标为 NaN。

按上述原则编写 Matlab 程序对数据进行遍历检验、剔除各操作位点中的异常数据，处理结果如表 4.3 所示。

表 4.3 超范围异常数据剔除结果

操作类别	样本 285	样本 313
剔除部分数据	49 号位点（40 个） 84 号位点（40 个） 111 号位点（40 个）	22 号位点（38 个） 33 号位点（31 个） ⋮ 323 号位点（3 个）
总计	涉及 3 个位点（120 个数据点）	涉及 30 个位点（299 个数据点）

3）拉以达准则检验

设对被测量变量进行等精度测量，得到 $x_1, x_2, ..., x_n$，算出其算术平均值 x 及剩余误差 $v_i = x_i - x(i = 1, 2, ..., n)$，

按照贝塞尔公式

$$\sigma = \left[\frac{1}{n-1} \sum_{i=1}^{n} v_i^2 \right]^{\frac{1}{2}} = \left\{ \left[\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2 / n \right] / (n-1) \right\}^{\frac{1}{2}}$$

算出标准误差 σ，若某个测量值 x_b 的剩余误差 $v_b\ (1 \le b \le n)$，满足

$$|v_b| = |x_b - x| > 3\sigma$$

则认为 x_b 是含有粗大误差值的坏值，应予剔除。

按 3σ 此准则对数据集进行检验，将含有出大误差值的坏值予以剔除，NaN，数据处理结果如表 4.4 所示。

表 4.4 拉以达准则检验异常数据剔除结果

操作类别	样本 285	样本 313
剔除部分数据	无	14 号位点（2 个） 40 号位点（1 个） ⋮ 339 号位点（1 个）
总计	无	涉及 42 个位点（58 个数据点）

4.2.2 缺失数据的处理

在 4.2.1 中，我们完成了对原始数据集中异常数据剔除工作。在此步中，我们进一步需检查数据集中缺失值的分布情况。一般情况下，当某变量的缺失值在数据集中的占比过高时，往往说明其包含的信息过少，需对其进行删除操作。若缺失值占比较低，则可通过拉格朗日插值对数据进行补充。定义数据缺失值比率 $Missing\_per$ 为：

$$Missing\_per=\frac{\text{位点的缺失数据个数}}{\text{位点的全部数据个数}}$$

首先计算各位点数据的缺失值比率。将计算值与缺失值比率的阈值（20%）相比，按照其是否超过阈值将缺失数据分为两类：（1）缺失值比率低的数据；（2）数据缺失值比率高的数据。接下来对两类数据的处理方法分别进行讨论。

1）数据缺失值比率低

对于缺失值比率较低的数据缺失情况，可认为各位点变量在较短的时间间隔内是按照线性关系变化的，故可利用数据缺失前后的数据进行两点拉格朗日插值，实现数据的补充。设某缺失段有 n 个缺失数据，缺失段前临数据为 x_a，后临数据为 x_b，则第 i 个填充数据 x_i（$i=1,2,...,n$）的计算公式为：

$$x_i = x_a + \frac{(x_b - x_a)}{n+1} i$$

根据此公式对符合条件的数据进行插值处理，处理结果如表 4.5 所示。

表 4.5 缺失值比率低（<20%）的缺失数据处理结果

操作类别	样本 285	样本 313
拉格朗日插值	无	33 号位点（3 个） 39 号位点（1 个） ⋮ 323 号位点（3 个）
总计	无	涉及 23 个位点（92 个数据点）

2）数据缺失值比率高

如果缺失值比例高于阈值，说明其缺失的数据量信息较多，仅通过拉格朗日插值、取均值等操作都不能较好地还原正常的位点采集数据。对于这种当数据的缺失量过大而对整个数据分析造成影响的情况，应该剔除数据缺失值比例过高的位点，以减少该段数据后续的数据挖掘工作带来干扰，增加分析难度。

缺失值比例高于阈值的位点进行剔除，处理结果如表 4.6 所示。

表 4.6 缺失值比率高（≥20%）的缺失数据处理结果

操作类别	样本 285	样本 313
删除位点	49 号位点 84 号位点 111 号位点	22 号位点 43 号位点 ⋮ 217 号位点
总计	涉及 3 个位点	涉及 7 个位点

4.2.3 样本确定

本题目标为降低 S Zorb 装置产品辛烷值损失，故确定样本的主要依据为样品的辛烷值数据。由于辛烷值的测定数据相对于操作变量数据而言相对较少，而且辛烷值的测定往往滞后，故采取以辛烷值数据测定的时间点为基准时间，取其前 2 个小时的操作变量数据的平均值作为对应辛烷值的操作变量数据作为确定某个样本的方法。

由表 4.2 可知 285 号和 313 号样本的辛烷值数据测定的时间点均为 8:00:00，故需取前两小时（6:00:00-8:00:00）各位点的数据的平均值作为 285 号和 313 号样本值。即本步需对进行了上述预处理操作后得到数据集中每个位点的数据取平均值并填入附件一。285 号和 313 号的部分样本值如表 4.7 所示。

表 4.7 经过整合的 285 号和 313 号样本值

编号	氢油比	反应过滤器差	还原器压力	…	D101 原料缓冲罐压力
285	0.260973	23.29049979	2.55482891	…	-85.02263508
313	0.261936	17.18346725	2.41705575	…	-113.3759175

至此，285 号和 313 号样本数据处理完毕。样本 285 中共检测出异常数据 120 个，补充数据 0 个，删除位点 3 个。样本 313 中共检测出异常数据 357 个，补充数据 92 个，删除位点 7 个。与附件一给出的预处理对比，存在微小差异，可能是因选取的处理标准有所不同，对后续建模过程的影响不大。

5　问题二：寻找建模主要变量

5.1 问题二分析

因辛烷值的测量较为复杂，根据实际情况认为某时刻测量的辛烷值是其前两小时内各个变量变化产生的综合效果，取操作变量两小时内的平均值与辛烷值的测量值对应，由此得到 325 个样本数据（附件一）。

每条样本数据中共包括 7 个原料性质、2 个待生吸附剂性质、2 个再生吸附剂性质、2 个产品性质变量以及另外 354 个操作变量（共计 367 个变量），变量的变化会对产品辛烷值造成不同程度的影响。本问希望对变量进行降维操作，从 367 个变量中提取出寻找建立辛烷值损失模型的主要变量，使挑选出的主要变量具有代表性和独立性，另外从工程应用角度来看，降维后的变量个数应小于 30，并考虑将原料的辛烷值作为建模变量之一。

本题的难点在于：（1）各操作变量与产品辛烷值之间具有高度非线性关系，判定因、自变量相关程度较为困难，同时为了后续的操作条件优化，选择的变量必须是原有变量，这是特征选取问题，无法使用较为常规的特征提取方法；（2）由于变量过多，变量与变量之间可能存在相互强耦连的关系，故选取变量的独立性问题较难处理。

针对难点（1）——主要变量代表性问题，数据降维算法分为特征选择与特征变换两类。特征选择为从给定的特征中直接选择若干重要特征，特征变换为通过某种变换将原始输入空间数据映射到一个新空间中。因为问题四中要对操作条件进行优化，所选取的主要变量必须为客观的物理量，数据提取的降维算法（如非负矩阵分析、因子分析、主成分分析、奇异值分解和独立成分分析）不适用于此问题。故最后采用基于有监督机器学习的随机森林算法获取到各变量对产品辛烷值贡献度的排名，依此实现对选取变量代表性的判断。

针对难点（2）——高耦合变量独立性问题，根据随机森林得到变量的贡献度排名后，依据从高到低的顺序对变量进行基于距离相关性系数的高相关性滤波，过滤掉耦合变量，以此实现提取具有独立性的特征变量。

最后，从变量降维过程中采用的算法及处理流程以及变量降维的最终结果两方面对所选择变量的合理性进行评价。问题二的思路流程图如图 5.1 所示。

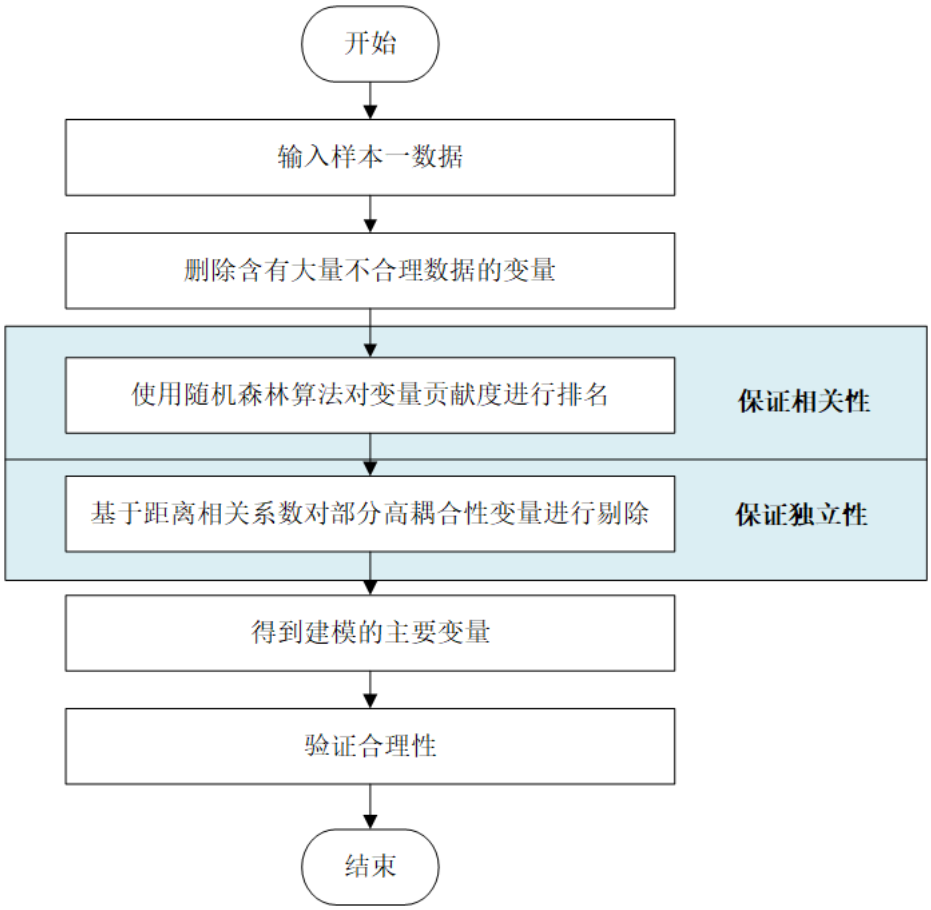

图 5.1 问题二思路流程图

5.2 变量降维

5.2.1 剔除含有过多异常值的变量

通过对附件一中的 325 组样本数据进行检测分析，发现其存在如部分变量值明显超出其物理意义所对应的正常范围、变量所对应的物理量在附件一和附件四中的单位定义矛盾等问题。若保留含有此类问题的变量，会对后续相关性及独立性分析结果的准确性造成影响。故在问题二中，应先对含有过多异常值的变量进行剔除，以保证后续数据挖掘的准确性。

5.2.2 随机森林算法——评价变量重要度

随机森林(Random forest)[5]是由美国科学家 Leo Breiman 将其在 1996 年提出的 Bagging 集成学习理论[6]与 Ho 在 1998 年提出的随机子空间方法[7]相结合，于 2001 年发表的一种机器学习算法。

随机森林是以 K 个决策树 $\{h(X,\theta_k),k=1,2,...,K\}$ 为基本分类器，进行集成学习后得到的一个组合分类器。当输入待分类样本时，随机森林输出的分类结果由每个决策树的分类结果简单投票决定。这里的 $\{\theta_k,k=1,2,...,K\}$ 是一个随机变量序列，它是由随机森林的两大

随机化思想决定的：

（1）Bagging 思想：从原样本集 X 中有放回地随机抽取 K 个与原样本集同样大小的训练样本集 $\{T_k, k=1,2,...,K\}$，每个训练样本集 T_k 构造一个对应的决策树。（本文设定 $K=500$）

（2）特征子空间思想：在对决策树每个节点进行分裂时，从全部属性中等概率随机抽取一个属性子集（通常取 $\log_2(\mathrm{M})+1$ 个属性，M 为特征总数，），再从这个子集中选择一个最优属性来分裂节点。（本文设定 M=150）

训练随机森林的过程就是训练各个决策树的过程，由于各个决策树的训练是相互独立的，因此随机森林的训练可以通过并行处理来实现，这将大大提高生成模型的效率。随机森林中第 k 个决策树 $h(X,\theta_k)$ 的训练过程如图 5.2 所示。

将以同样的方式训练得到 K 个决策树组合起来，就可以得到一个随机森林。当输入待分类的样本时，随机森林输出的分类结果由每个决策树的输出结果进行简单投票（即取众数）决定。随机森林分类流程如　　图 5.3 所示。

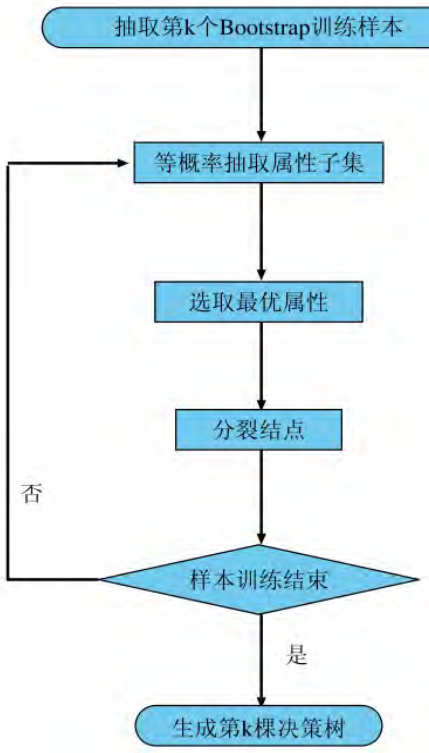

图 5.2 随机森林中单个决策树训练过程

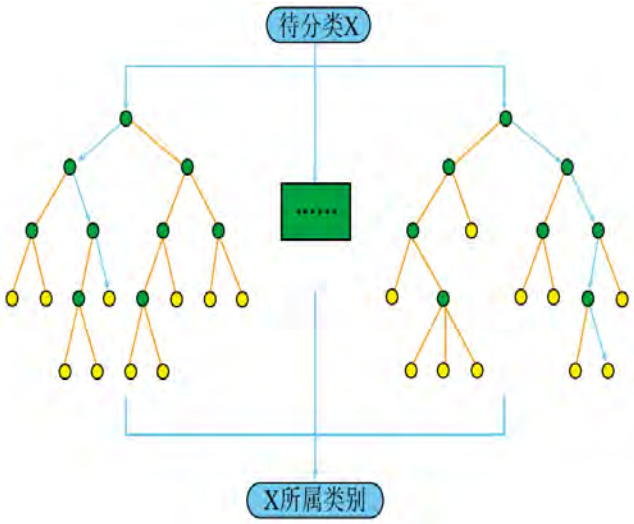

图 5.3 随机森林分类过程

随机森林对噪声和异常值有较好的容忍性，能够在不需要降维的条件下处理具有高维特征的输入样本，而且能够评估各个特征在分类问题上的重要性，具有良好的可扩展性和并行性。

随机森林算法可以在分类的基础上进行回归分析，通过将样本分类的结果进行一定的运算可以获得各个特征重要性特征的重要性表示特征对预测结果影响程度，某一特征重要

性越大，表明该特征对预测结果的影响越大，重要性越小，表明该特征对预测结果越小。随机森林算法中某一特征的重要性，是该特征在内部所有决策树重要性的平均值，而在决策树中，计算某一个特征的重要性可以采用以下方法：

即某一节点 k 的重要性为：

$$n_k = \omega_k * G_k - \omega_{left} * G_{left} - \omega_{right} * G_{right}$$

其中，$\omega_k, \omega_{left,} \omega_{right}$ 分别为节点 k 以及其左右节点中训练样本与总训练样本数目的比例，G_k, G_{left}, G_{right} 分别为节点 k 以及其左右子节点的不纯度。知道每一个节点的重要性之后，即通过公式得出某一特征的重要性。

$$f_i = \frac{\sum_{j \in nodes\ split\ on\ feature\ i} n_j}{\sum_{k \in all\ nodes} n_k}$$

为了使所有特征的重要性加起来等于 1，需要对每一个特征重要性进行归一化处理，如下式：

$$f_{ni} = \frac{f_i}{\sum_{j \in all\ features} f_j}$$

使用 Python 应用随机森林算法，综合考虑到算法速度和算法准确率，设定 $K = 500$，M=150。运行程序得到 359 个变量的贡献度排名，将贡献度由大到小排序。考虑到下一步的高相关性滤波操作会对进一步变量进行降维，故在这一步中先筛选出排名处于前 60 的变量，如表 5.1 所示。可以发现排名为 60 的变量对辛烷值的归一化贡献度只有 0.18%，说明选取贡献度排名前 60 名的变量已经可以有足够的信息来预测辛烷值。

表 5.1 贡献度排名前 60 的变量名称及贡献度大小

排名	变量名称	贡献度大小
1	辛烷值 RON	36.83%
2	饱和烃（烷烃+环烷烃）	5.66%
3	硫含量	3.66%
⋮	⋮	⋮
60	S-ZORB.PT_2801.PV（还原器压力）	0.18%

5.2.3 变量高相关性滤波算法——去耦合

考虑到此题自变量之间具有非线性和高耦合的特征，有必要对其进行高相关性滤波，从互相高度相关的变量中仅保留贡献值最大的变量，认为它能代表所在高耦合变量组的全部信息。

传统的 Person 相关系数常用来衡量两个数据集合是否在一条线上面，即衡量定距变量

间的线性关系，只适用于两变量呈线性关系的情况。本题中各变量间为非线性关系，Person相关性分析并不适用，为此我们选择**距离相关系数（Distance Correlation）**作为衡量变量间相关性的指标。

Szekely 等人[8]定义了距离相关系数的概念，克服传统的 Pearson 相关系数的缺点，可以衡量非线性相关变量之间的关系。在某些情况下，即便 Pearson 相关系数为 0，也无法将两个变量判定为线性无关（有可能非线性相关），但是，如果距离相关系数为 0，则可将两个变量判定为互相独立。对于两个随机向量 $x \in R^p, y \in R^q$,记 $(x,y)=\{(x_i,y_i): i=1,...,n\}$ 为观察到的随机样本，x,y 间的距离相关系数（dCor）可以定义为：

$$R^2(x,y)=\frac{v^2(x,y)}{\sqrt{v^2(x,x)v^2(y,y)}}$$

其中：

$$v^2(x,y)=\frac{1}{n^2}\sum_{i,j=1}^{n}A_{i,j}B_{i,j}$$

$$A_{i,j}=\| x_i-x_j \|_2-\frac{1}{n}\sum_{k=1}^{n}\| x_k-x_j \|_2-\frac{1}{n}\sum_{l=1}^{n}\| x_i-x_l \|_2+\frac{1}{n^2}\sum_{k,l=1}^{n}\| x_k-x_l \|_2$$

$$B_{i,j}=\| y_i-y_j \|_2-\frac{1}{n}\sum_{k=1}^{n}\| y_k-y_j \|_2-\frac{1}{n}\sum_{l=1}^{n}\| y_i-y_l \|_2+\frac{1}{n^2}\sum_{k,l=1}^{n}\| y_k-y_l \|_2$$

同理可计算：

$$v^2(x,x)=\frac{1}{n}\sum_{i,j=1}^{n}A_{i,j}^2$$

$$v^2(y,y)=\frac{1}{n}\sum_{i,j=1}^{n}B_{i,j}^2$$

距离相关系数越大，表示两者相关性越强。在统计学中，对相关性强弱有如下约定，如表 5.2 所示。

表 5.2 相关程度度量表

相关性	相关系数
极强相关	0.8~1.0
强相关	0.6~0.8
中相关	0.4~0.6
弱相关	0.2~0.4
极弱或无相关	0~0.2

根据题目要求，去耦合后的变量之间应具有中相关及以下的相关性，故将距离相关系数（dCor）的阈值设为 0.6。基于此，对表 5.1 中的变量进行相关性检验。首先计算两两变量间的距离相关系数，然后判断两者的相关系数是否大于所设阈值。若大于，则说明两变量相关性高，对两变量间贡献值排名靠后的那一变量采取删除操作。若小于，则说明两变量间并不强相关，则将两变量都暂时予以保留。之后重复上述操作，直至遍历结束。使用 Matlab 编写此去耦合程序，程序流程图如图 5.4 所示。

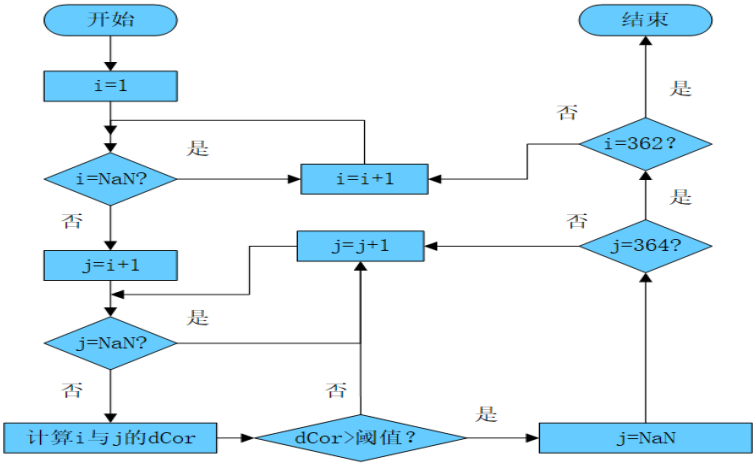

图 5.4 去耦合程序流程图

根据此去耦合程序，对 5.2.2 中得到的表 5.1 中变量进行高相关性滤波处理，前 60 个贡献值高的变量经滤波处理后剩余 27 个变量，如表 5.3 所示。将这 27 个变量作为降维后得到的主要变量，供下文建立模型时使用。

表 5.3 进行高相关性滤波处理过后的剩余变量（降维结果）

排名	名称	排名	名称
1	原料辛烷值	15	再生风流量
2	原料饱和烃百分比	16	E-206 壳程出口管温度
3	原料硫含量	17	再生器顶部/再生器接收器差压
4	还原器温度	18	稳定塔底出口温度
5	催化汽油进装置总流量	19	紧急氢气去 R-101 流量
6	净化风进装置流量	20	闭锁料斗氧含量
7	轻烃出装置流量	21	火炬气排放流量
8	K-101B 进气压力	22	火炬罐 D-206 液位
9	循环水进装置流量	23	EH-103 加热元件温度
10	E203 重沸器管程出口凝结水流量	24	D-203 顶部出口管温度
11	D121 去稳定塔流量	25	密度 kg/m³
12	D-110 底流化 N2 流量	26	E-101 管程入口总管压力
13	P-101B 入口过滤器差压	27	R-102 底滑阀差压
14	K-103 出口去 K-101 出口管流量	/	/

5.3 变量选择合理性评价

1）从变量降维过程中采用的算法及处理流程来看：

根据随机森林得出的变量贡献度排名，保留排名靠前的变量从而保证了所选取变量与因变量之间的相关性，而设计的基于距离相关性高相关度变量滤波算法则保证了降维后变量之间的独立性，可根据所提取的特征变量之间的距离相关系数计算结果对变量之间的相关程度进行可视化，如图 5.5 所示。可见选取的 27 个变量之间相关性低，独立性较好。

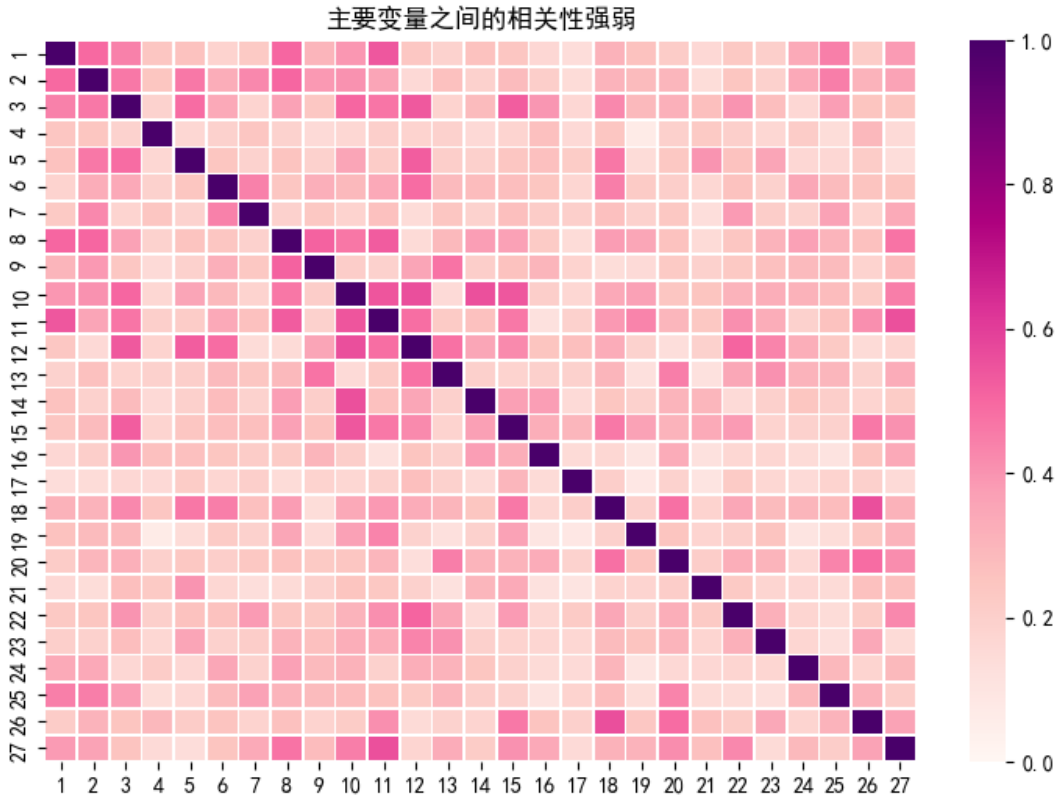

图 5.5 27 个主要变量之间的相关度分布图

2）从变量降维的最终结果来看：

通过查阅资料可知，S-Zorb 装置的工艺流程如图 5.6[3]所示。

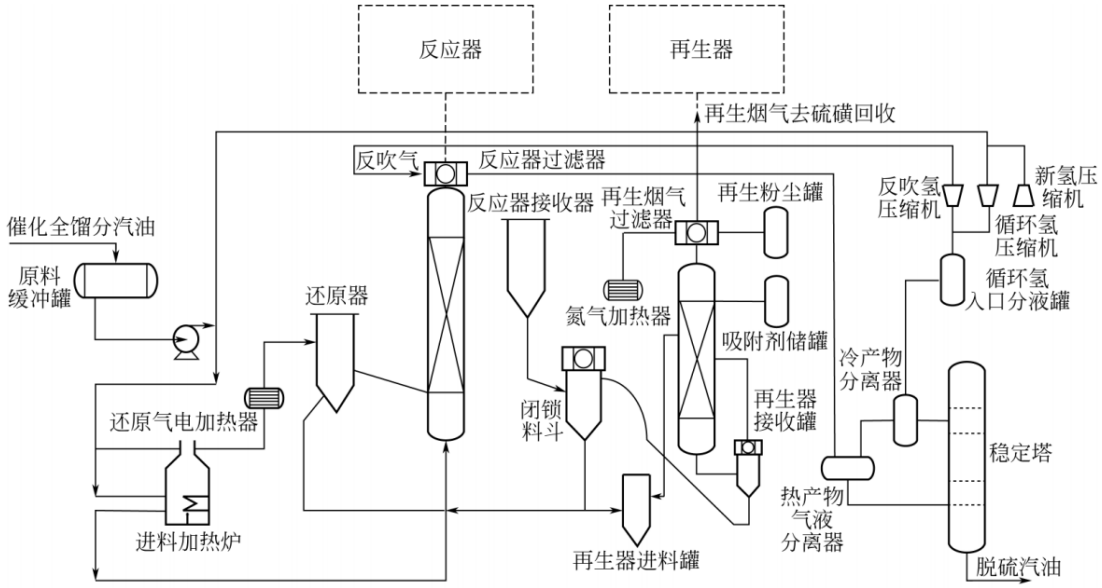

图 5.6 S-Zorb 装置工艺流程图[3]

根据此工艺流程图，结合变量的实际物理意义，可将题目给出的 365 个变量做进一步分为 6 类，各分类定义如表 5.4 所示：

表 5.4 根据变量实际物理意义的分类结果

序号	变量类别名称	含义
1	原料性质	所用原料的性质，即题目描述中给出的 7 个变量
2	温度情况	各个装置、反应过程中的温度情况
3	压力情况	各个装置、反应过程中的压力及压差情况
4	输入流量	加入参加反应原料的流量，如原油、再生风和循环水的流量等
5	输出流量	反应后需要排出的废物及分馏产生的产品的流量
6	过程流量	各步骤之间传递的物质流量

根据上述的变量分类定义，对在 5.2.3 中得到的 27 个主要变量进行分类，统计每个类别下的主要变量个数，结果如图 5.7 所示。

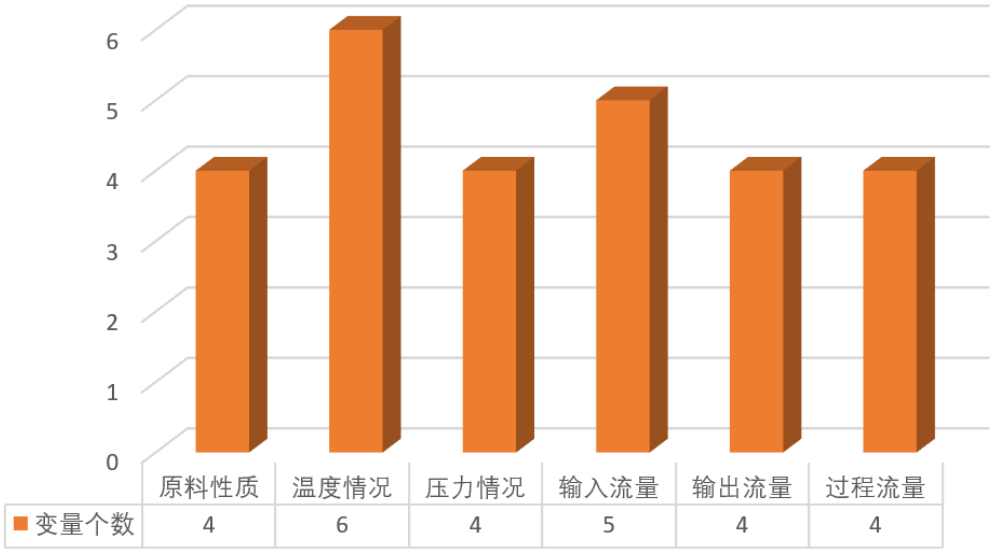

图 5.7 27 个主要变量的分类统计结果

从各变量所属类的统计结果可以看出：（1）所提取变量包括了原料特征、温度特征、压力特征、输入物质数量上的特征、输出物质数量上的特征及各个过程之间物质传递的数量特征，对工艺流程的反应较为全面。（2）从每类所包含变量个数分布情况上来看，方差较小，说明所提取变量对各类别的侧重程度较为均衡。

6　问题三：辛烷值（RON）损失预测模型

6.1 问题三分析

问题三要求采用上述样本和建模主要变量，通过数据挖掘技术建立辛烷值（RON）损失预测模型，并进行模型验证。从数据类型上来看，变量之间可能具有高度非线性关系。BP 神经网络是人工智能网络的一个典型算法，其本身具有很强的非线性映射能力，非常适合解决一些非线性问题。另外其网络拓扑结构简单，而且具有较高的计算精度。因此在此问中，我们将样本数据集数据进行标准化处理之后，采用从 325 组样本数据中抽取 80%的数据用于 BP 神经网络模型训练，使用剩下的 20%的数据对模型精度进行验证以评价模型合理性。问题三的思路流程图如图 6.1 所示。

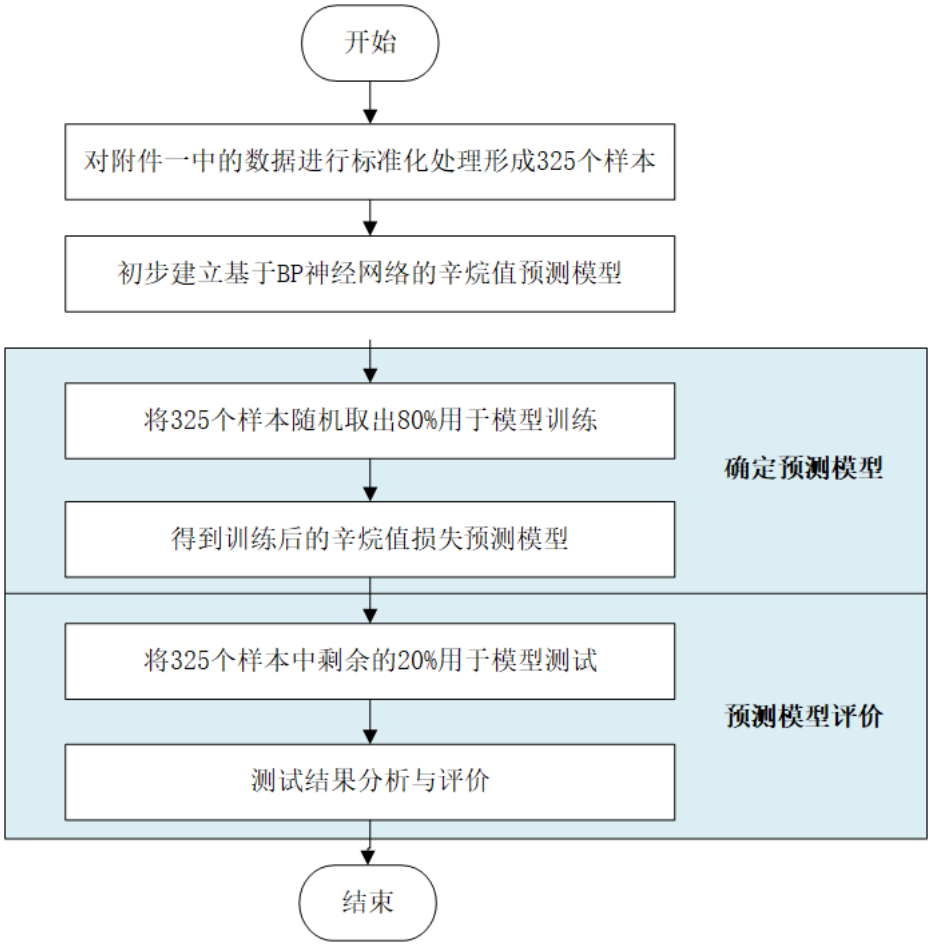

图 6.1 问题三思路流程图

6.2 辛烷值损失预测模型的建立

6.2.1 BP 神经网络基本原理

BP(Back Propagation)神经网络是 1986 年由 Rumelhart 和 McClelland 为首的科学家提出的概念[9]，是一种按照误差逆向传播算法训练的多层前馈神经网络，是应用最广泛的神经网络，其基本原理如下。

（1）神经网络

神经网络是由大量简单神经元相互连接构成的复杂网络。一个典型的具有 n 维输入、S 个神经元的单层神经网络模型可用图 6.2 加以描述。

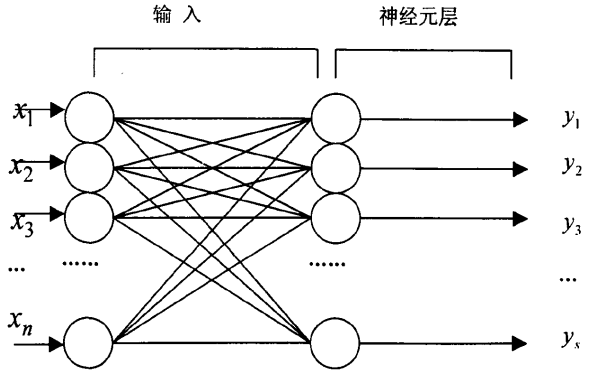

图 6.2 单层神经元网络模型

在图 6.2 中 X 为 n*1 维的输入矢量，网络层由权值矩阵$W(S\times n)$阈值矢量$b(s\times 1)$，S 个神经元的输出组成了 S*1 维的神经网络输出矢量 y：

$$y = f(wx+b)$$

其中，输入层网络权值矩阵 w 和阈值矢量 b 的具体形式如下：

$$w=\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{S1} & w_{S2} & \cdots & w_{Sn} \end{bmatrix}, b=\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_S \end{bmatrix}$$

在单层神经网络基础上可以构造多层神经网络。一个典型的三层神经网络模型如图 6.3 所示。

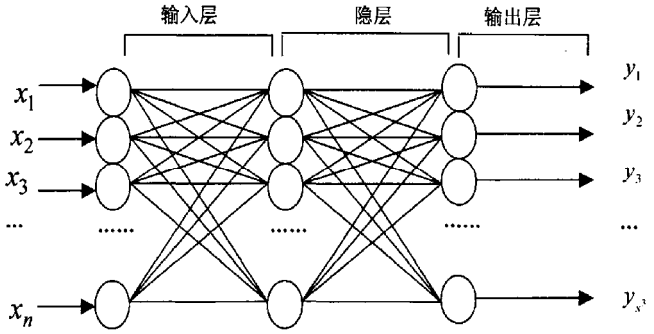

图 6.3 图三层神经网络

产生神经网络最终输出的网络层称为输出层，中间层称为隐层。图 6.3 所示神经网络三层神经元的数目分别为：S^1,S^2,S^3。为了方便描述，采用上标法对神经网络中相关元素加以标记，其中$IW^{1,1}(S^1\times n)$表示输入层权值矩阵，$LW^{2,1}(S^2\times S^1)$和$LW^{3,2}(S^3\times S^2)$分别表示第一层到第二层、第二层到第三层的网络层权值矩阵。b^1,b^2,b^3分别表示各层的网络域值矢量，则神经网络的输出为：

$$y^3 = f^3(LW^{3,2}f^2(LW^{2,1}f^1(Iw^{1,1}+b^1)+b^2)+b^3) = y$$

（2）BP 算法

BP 算法由正向传播和反向传播两部分组成。在正向传播过程中输入信息从输入层经隐层单元处理后传至输出层。每一层神经元的状态只影响下一层神经元的状态。如果在输出层得不到期望输出就转为反向传播，即把误差信号沿连接路径返回并通过修改各层神经元之间的连接权值使误差信号最小。

假设该网络有 M 层而第 M 层仅含输出节点，第一层为输入节点。另外，假设有 N 个标准样本对 $\{x_1(p), x_2(p), ..., x_n(p); x_i(p) \mid p = 1, 2, ..., N\}$，对网络 M 层输出为 $O(p)$。

选择隐层节点和输出层节点的作用函数，如 Sigmoid 函数：

$$f(x) = \frac{1}{1+e^{-Qx}}$$

其中：Q 为表示神经元非线性的参数，称之为增益值，也可称调节参数。Q 值越大，S 曲线越陡峭；反之，Q 值越小，S 曲线越平坦；取 Q=1；

假设对某一输入 $x(p)$，网络输出为 $O(p)$，节点 i 的输出为 x_i，节点 j 的输入为、$net_j = \sum_i w_{ji} x_i$，节点 j 的输出为 $y_i = f(net_j)$。

定义误差函数为：

$$E = \frac{1}{2N} \sum_{p=1}^{N} (O_p - T_p)^2$$

其中：T_p 为网络目标输出。

现定义 $E = \frac{1}{2}(O(p) - T(p))^2, \delta_j = \frac{\partial E}{\partial net_j}$，且 $y_i = f(net_i)$，于是有：

$$\frac{\partial E}{\partial W_{ji}} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial W_{ji}} = \frac{\partial E}{\partial net_j} \cdot x_i = \delta_j x_i$$

然后分两种情况来讨论：

（i）当 j 为输出节点时，$y_j = O$，有：

$$\delta_j = \frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial net_j} = -(T - O) f'(net_j)$$

（ii）若 j 为隐层节点时，有：

$$\partial_j = \frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} = \frac{\partial E}{\partial y_j} \cdot f'(net_j)$$

其中： y_j 为送到下层 $(l+1)$ 的输入，计算 $\frac{\partial E}{\partial y_i}$ 要从 $(l+1)$ 层送回。

在 $(l+1)$ 层第 m 个节点：

$$\frac{\partial E}{\partial y_j}=\sum_m \frac{\partial E}{\partial net_j}\cdot\frac{\partial net_j}{\partial y_j}=\sum_m \frac{\partial E}{\partial net_m}\cdot\frac{\partial}{\partial y_j}\sum_j W_{jm}O_j=\sum_m \frac{\partial E}{\partial net_m}\cdot\sum_j W_{jm}=\sum_m \delta_m W_{jm}$$

由上面两式可以得到：

$$\begin{cases}\delta_j = f^{'}(net_j)\sum_m \delta_m W_{jm}\\ \dfrac{\partial E}{\partial W_{jm}}=\delta_j y_j\end{cases}$$

有 $f^{'}(net_j)=\frac{dE}{dnet_j}[\frac{1}{1+e^{-Qnet_j}}]=Q(1+e^{-Qnet_j})^{-2}\cdot e^{-Qnet_j}=Q\cdot f(net_j)(1+f(net_j))=Q\cdot y_j\cdot(1+y_j)$

根据上述推导，具体的 BP 算法步骤可概括如下：

Step1：选取初始权值、阈值。

Step2：重复下述过程直至满足性能要求为止：

Move1：对于学习样本 $p=1$ 到 N ，包括：

①正向过程——计算每层各节点 j 的 y_j, net_j 和 O 的值；

②反向过程——对各层 M 层到第二层，对每层个节点，反向计算 δ_j 。

Move2：修正权值：

$$W_{ji}(t+1)=W_{ji}(t)-\eta\frac{\partial E}{\partial W_{ji}},\eta>0$$

上述算法所对应的程序框图如图 6.4 所示。可见 BP 神经网络模型把一组输入输出样本的函数问题转变为了一个非线性优化问题，使用了优化技术中的梯度下降法。如果把神经网络看成是输入到输出的映射，则这个映射是一个高度非线性映射，适合用来解决本题中特征变量与辛烷值关系的非线性问题。

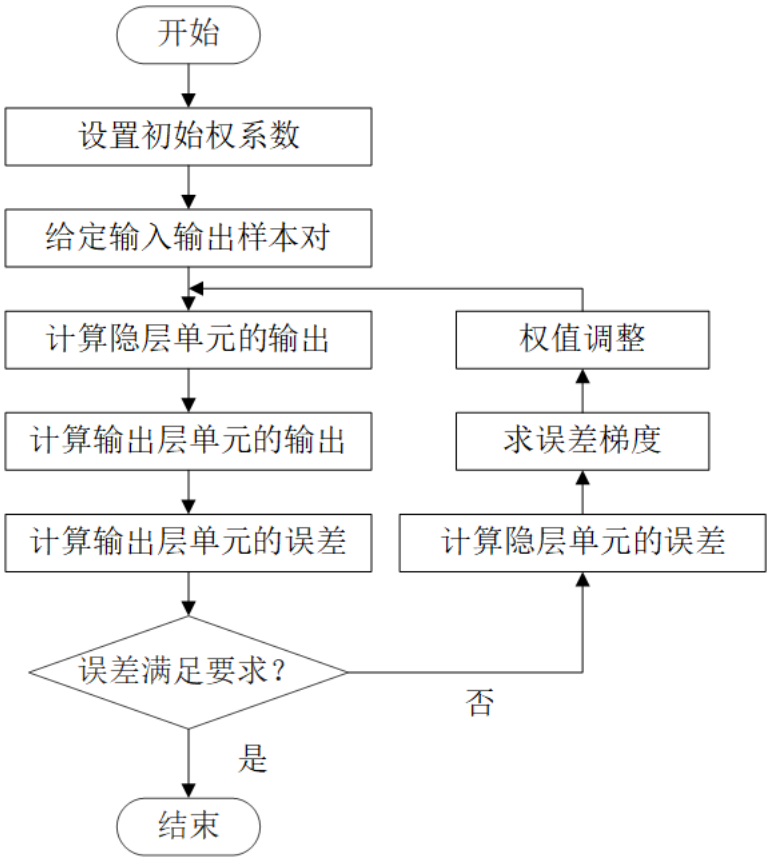

图 6.4 BP 学习算法框图

6.2.2 辛烷值损失预测模型建立

本文采用含有一个隐层三层多输入单输出结构建立的用于辛烷值预测的 BP 神经网络模型。

（1）数据整理

此步需对附件一中的 325 组数据进行整理，保留前文提取出的 27 项主要变量及产品辛烷值这一变量，用 Python 对这些数据进行标准化处理，将每一行作为一组输入训练集，得到数据集 Data_RON，将数据集中的 325 组数据按 80%和 20%比例分为训练集数据（80%）和测试集数据（20%）。

（2）设计网络结构层

1）输入输出层：将前文提取 27 项主要变量作为输入，产品辛烷值作为输出。故输入层神经元个数 $n=27$，输出层神经元个数 $m=1$。

2）隐层：在网络设计过程中，隐层神经元数的确定十分重要。隐层神经元个数过多，会加大网络计算量并容易产生过度拟合问题；神经元个数过少，则会影响网络性能，达不到预期效果。网络中隐层神经元的数目与实际问题的复杂程度、输入和输出层的神经元数以及对期望误差的设定有着直接的联系。目前，对于隐层中神经元数目的确定并没有明确的公式。本文出于保证精度的考虑，经反复调试，最终确定隐层神经元个数 $l=50$。

本文设计的网络结构示意图如图 6.5 所示：

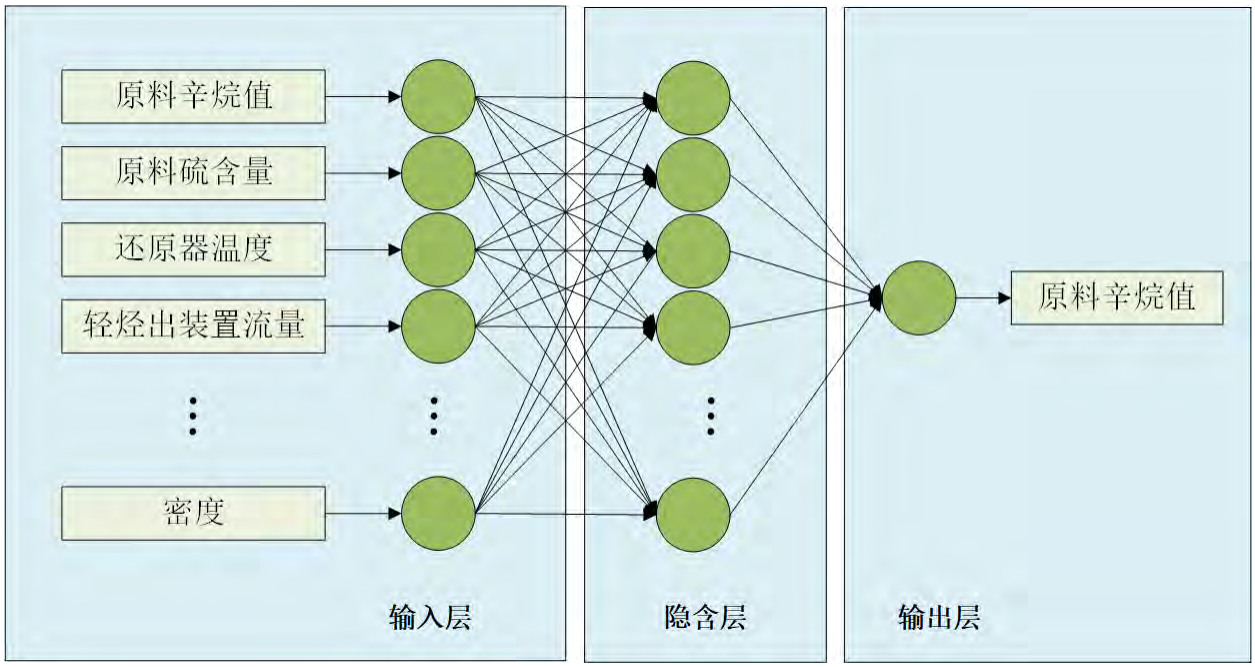

图 6.5 本文设计的神经网络结构示意图

（3）激励函数的选取

激励函数的作用是提供规模化的非线性化能力，使得神经网络可以任意逼近任何非线性函数，模拟神经元被激发的状态变化。如果不用激励函数，每一层输出都是上层输入的线性函数，无论神经网络有多少层，输出都是输入的线性组合。

目前主要有三种常用的激励函数：Sigmoid、Thah 和 ReLU 激励函数。ReLU 使得 SGD 的收敛速度比 Sigmoid 和 Thah 快很多，使过程计算量减少，此外还解决了梯度消失问题。出于此种考虑，本文最后采用 ReLU（Rectified Linear Uni）激励函数作为本神经网络的激励函数。其形式如下：

$$f(x)=\begin{cases}0 & x\le 0\\ x & x>0\end{cases}$$

（4）模型实现

最终选用 Python 编写程序进行神经网络训练。采用三层结构 BP 神经网络模型，将输入层神经元个数等于输入变量数为 27；输出层神经元个数等于输出变量数为 1；隐含层神经元个数设置为 50；采用 ReLU 激励函数；设置最大迭代次数为 3000 次，期望误差为 0.00000001，学习速率设置为 0.001。

6.2.3 模型求解结果

该网络通过 15 次重复学习即可达到期望误差，之后结束学习。该网络训练完成后，只需将各项主要变量值输入网络即可得到预测数据。故将此网络作为辛烷值 RON 预测模型，而辛烷值 RON 损失模型只需计算 RON 的预测值与实际值的差值，即：

$$\mathrm{RON}_{\text{损失}} = \mathrm{RON}_{\text{原料值}} - \mathrm{RON}_{\text{预测值}}$$

至此，得到基于 BP 神经网络算法的 RON 损失预测模型，可将此模型定义为一个映射：

$$BPNN\_RON : X_{corr} \rightarrow \delta_{RON}$$

其中：

X_{corr} ——含有所选 27 个变量的集合；

δ_{RON} ——辛烷值（RON）损失值。

6.2.4 硫含量预测模型建立及求解

因为问题四中要考虑产品硫含量的范围限制，所以依照上一小节建立产品辛烷值预测模型的方法，同样使用 27 个主要变量建立产品硫含量的 BP 神经网络预测模型，此硫含量预测模型定义为一个新映射:

$$BPNN\_S : X_{corr} \rightarrow S$$

其中：

X_{corr} ——含有所选 27 个变量的集合；

S ——产品硫含量。

6.3 模型验证

6.3.1 辛烷值损失预测模型验证

将测试集导入训练好的 RON 损失预测模型中，将预测得到的结果与其真实值进行对比，结果如图 6.6 所示：

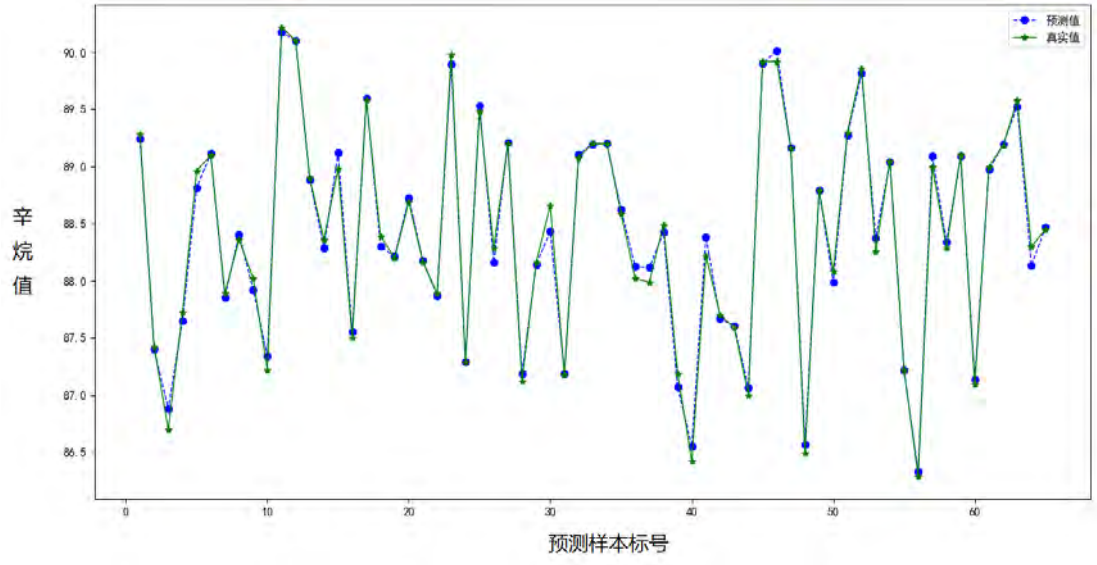

图 6.6 BP 神经网络预测结果与真实值对比——RON

从图 6.6 中，我们可以看出预测值与其对应的真实值十分相近，说明该模型具有较好回归结果，可较为真实地反应辛烷值损失。

之后计算各样本的预测误差百分比，计算结果如图 6.7 所示：

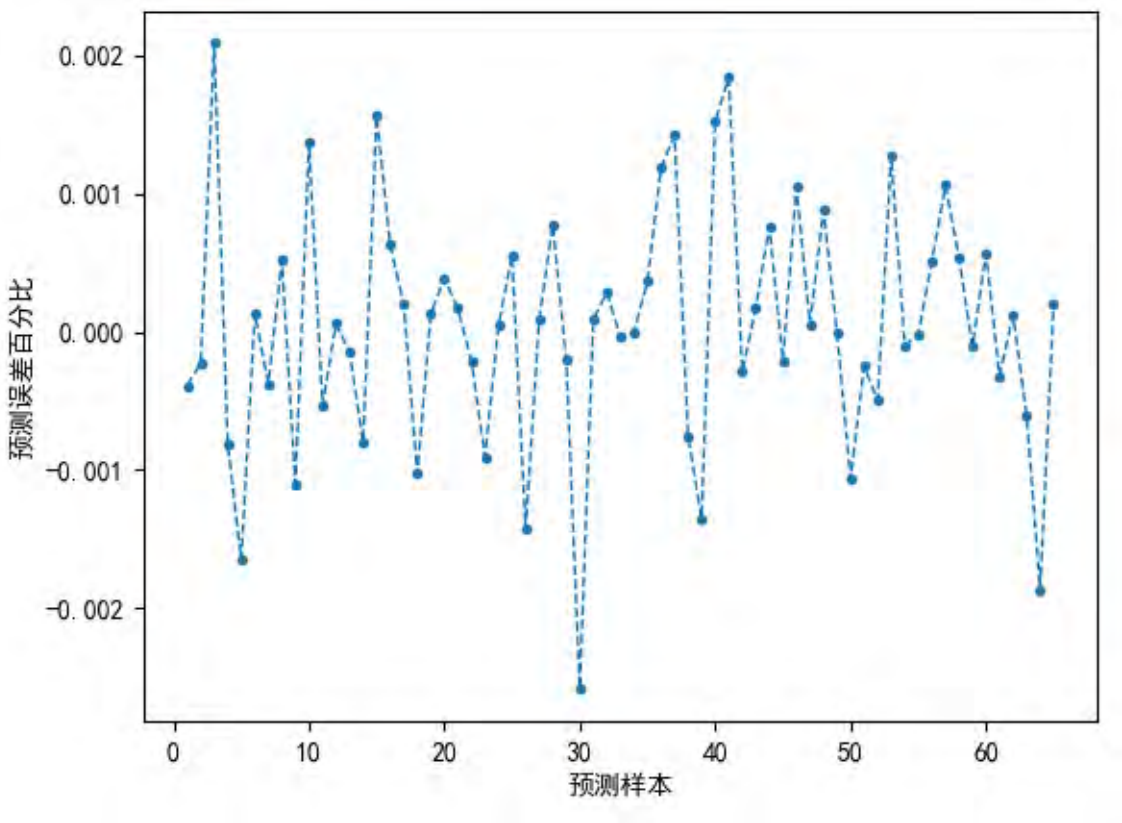

图 6.7 BP 神经网络预测误差百分比——RON

从图 6.7 中我们可以看出训练得到的 BP 神经网络模型预测结果较为准确，样本中神经网络预测的误差百分比最大只有 0.25%。

另外均方误差（Mean Squared Error，MSE）是预测值与真实值之差的平方和的平均值，计算公式为：

$$\frac{1}{n}\sum_{i=1}^{n}[f(x_i)-y_i]^2$$

均方误差被用来作为衡量预测结果的一个指标，本模型结果的均方误差为 0.0146，误差值较小，说明该预测模型预测精度较高。

6.3.2 硫含量预测模型验证

模型验证步骤与上一小节相同，验证结果如图 6.8 和图 6.9 所示：

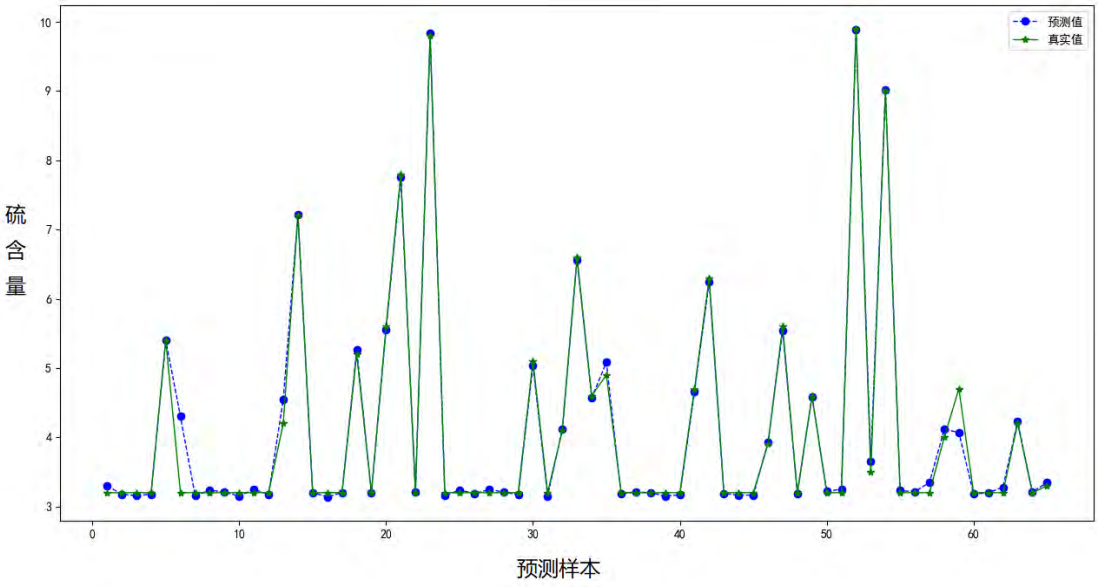

图 6.8 BP 神经网络预测结果与真实值对比——S

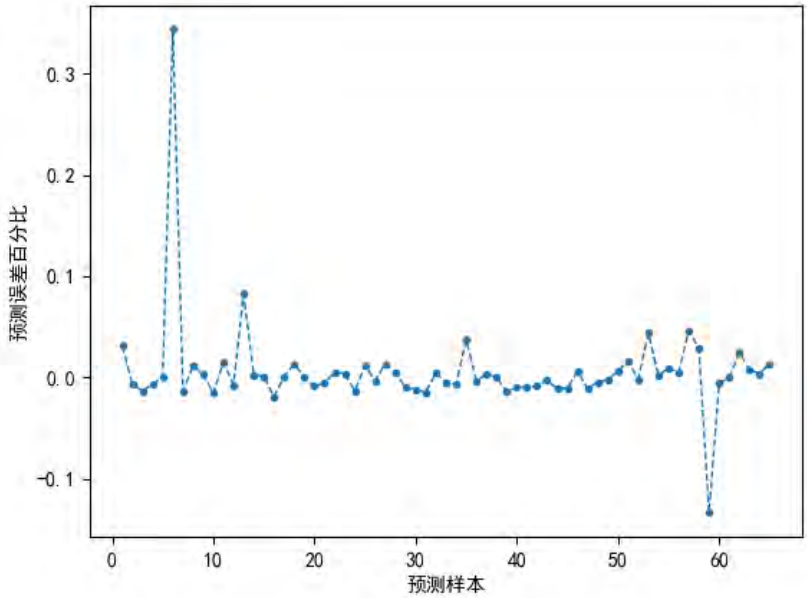

图 6.9 BP 神经网络预测误差百分比——S

由图可知，产品硫含量预测模型的最大相对误差仅为 0.34%，预测效果优秀。

7　问题四：主要变量操作方案的优化

7.1 问题四分析

本文要求对主要变量操作方案进行优化。在保证产品硫含量不大于 5 μg/g 的前提下，利用第 6 章中的模型获取优化后的 325 个样本的辛烷值预测值，将其与之前的产品辛烷值进行对比，筛选出辛烷值（RON）损失降幅大于 30%的样本对应的主要变量优化后的操作

条件（优化过程中原料、待生吸附剂、再生吸附剂的性质保持不变，以它们在样本中的数据为准）。

本问可简化为一个基于约束的目标优化搜索问题。难点在于：本题约束较多，不仅对自变量有所约束（操作条件需要在限定范围内进行调整），而且对因变量也有所约束（辛烷值损失降幅要大于 30%），这使搜索最优解。针对这一难点，因为选取的因变量为连续变量，故不能通过穷举得搜索方式找到最优解，另外普通的求极值方法往往会陷入得到的结果是局部最优解的困境。经分析，本文决定采用粒子群算法，以保证获取的解是全局最优解，另外粒子群算法速度较快，可在短时间内找到最优解。

本文的思路如下：首先依据题意，设定优化目标和约束，之后调整粒子群算法的主要参数，对 325 个数据样本进行求解，得到满足题意的样本对应的主要变量优化后的操作条件，以 EXCEL 表格列出。思路流程图如图 7.1 所示。

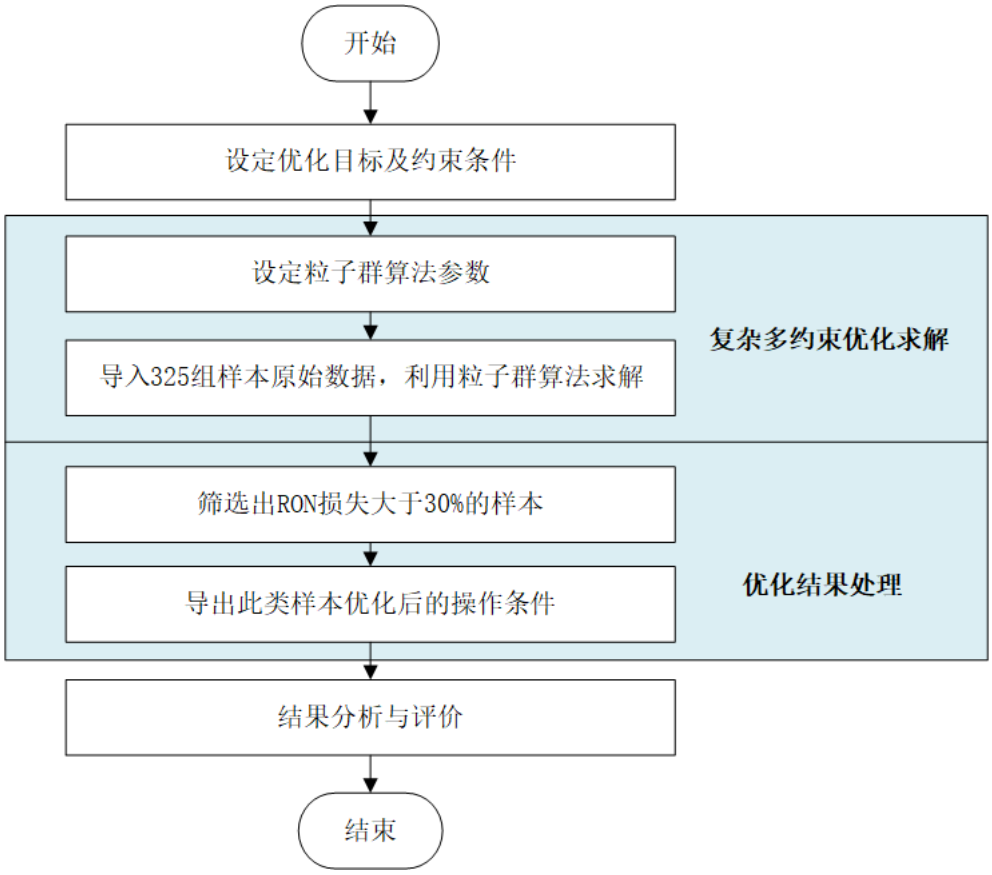

图 7.1 问题四思路流程图

7.2 主要变量操作方案的优化模型建立

7.2.1 粒子群算法

（1）粒子群算法原理

粒子群优化算法来源于 1987 年 Reynolds 提出的 Boid 模型。在该模型中，鸟群的飞行空间即为解空间，鸟群即为粒子群，通过群体信息的共享和更新不断向优化目标方向飞行。

粒子没有质量和体积，只有速度和位置信息，粒子通过速度和位置信息的更新逐渐趋向优化目标，在这个过程中，粒子跟踪个体历史最优位置和群体历史最优位置。粒子群优化算法的迭代示意图如图 7.2 所示

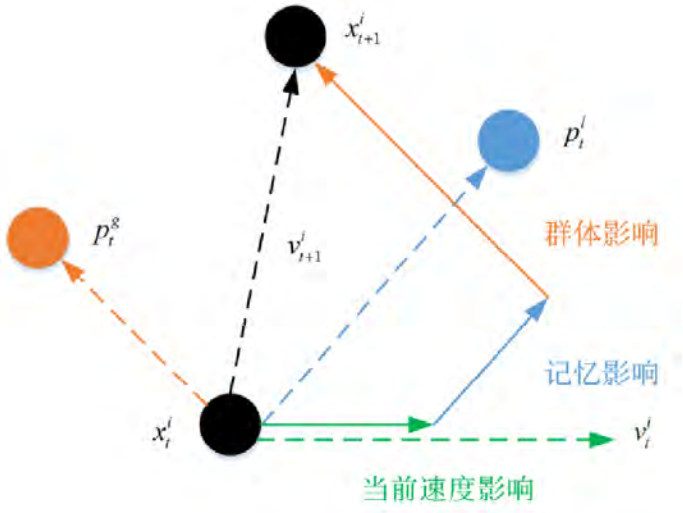

图 7.2 粒子迭代示意图

假设优化变量的数量为 D，则粒子群的搜索空间为 D 维，在此空间中，有 M 个粒子组成的粒子群。

粒子 i 在 t 时刻的位置：

$$x_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{id}^t), x_{id}^t \in [l_d, u_d]$$

粒子 i 在 t 时刻的速度：

$$v_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{id}^t)， \quad v_{id}^t \in [v_{\min}, v_{\max}]$$

其中：

$v_{\min}$ ——粒子速度的最小值；

$v_{\max}$ ——粒子速度的最大值；

l_d ——粒子搜索空间的下限；

u_d ——粒子搜索空间的上限；

粒子的每次迭代记录两个位置：

一个是个体最优位置：

$$p_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{id}^t)$$

另一个是种群最优位置：

$$p_g^t = (p_{g1}^t, p_{g2}^t, ..., p_{gd}^t)；$$

其中：$1 \le i \le m, 1 \le d \le D$。

粒子i在$t+1$时刻的速度、位置更新公式如下：

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_1 (p_{id}^{t} - x_{id}^{t}) + c_2 r_2 (p_{gd}^{t} - x_{gd}^{t})$$

$$x_{td}^{t+1} = x_{td}^{t} + v_{id}^{t+1}$$

其中：

v_{id}^{t+1}——在$t+1$时刻迭代粒子i飞行速度矢量的第d维分量；

x_{td}^{t+1}——在$t+1$时刻迭代粒子i位置矢量的第d维分量；

c_1、c_2——学习因子，用以调节学习最大步长；

r_1、r_2——两个的随机数，取值范围[0,1]，用以增加搜索随机性；

ω——惯性权重，非负数，调节对解空间的搜索范围，根据经验，当$\omega \in [0.9,1.2]$时，算法具有比较好的搜索性能。

粒子速度更新公式包含三部分：

（1）第一部分为粒子先前的速度；

（2）第二部分为“认知”部分，表示粒子本身的思考，可以理解为粒子i当前的位置与自己最好位置之间的距离；

（3）第三部分为“社会”部分，表示粒子间的信息共享与合作，可理解为粒子i当前位置与群体最好位置之间的距离。

如图 7.3 所示，粒子群优化算法的流程可总结如下：

Step1：初始化粒子群体（群体规模为 m），包括随机位置和速度。

Step2：根据适应度公式，计算每个粒子的适应值。

Step3：对每个粒子，将其当前适应值与其个体历史最佳位置对应的适应值作比较，如果当前的适应值更高，则将用当前位置更新历史最佳位置。

Step4：对每个粒子，将其前适应值与其个全局最佳位置对应的适应值作比较，如果当前的适应值更高，则将用当前位置更新全局最佳位置。

Step5：根据公式更新每个粒子的速度与位置

Step6：判断是否满足结束条件，如未满足，则返回 Step2。（通常算法达到最大迭代次数G_{max}或者最佳适应度值的增量小于某个给定的阈值时算法停止。）

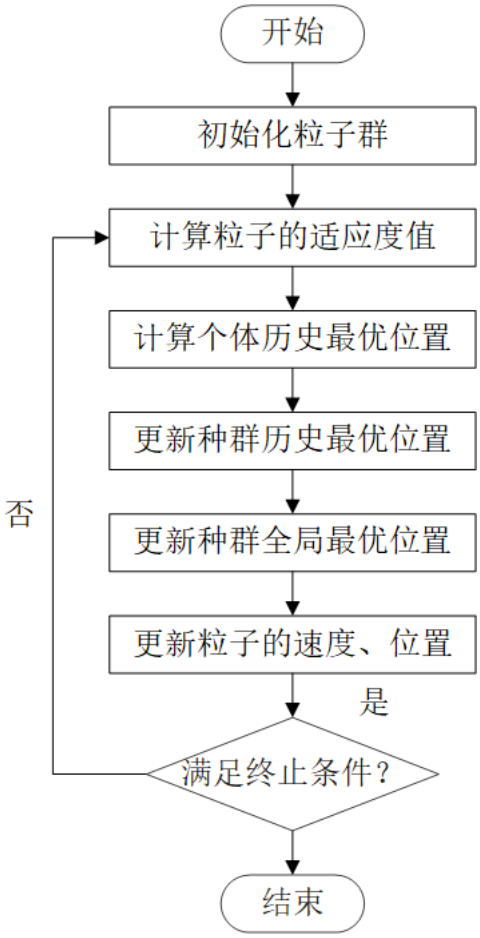

图 7.3 粒子群算法流程图

7.2.2 优化目标及约束设定

（1）决策变量：

本文所建立的模型中影响辛烷值含量的主要变量一共有 27 个，其中 4 个变量属于原料性质，是不可变的量。因此，该粒子群算法模型中共有 23 个可变变量，即这里的决策变量有 23 个，记为：

$$X = \{x_1, x_2, x_3, ...x_{23}\}$$

（2）目标函数：

问题要求在保证产品硫含量不大于 5 μg/g 的前提下，获得 325 个数据样本中，辛烷值（RON）损失降幅大于 30%的样本对应的主要变量优化后的操作条件。这里我们以辛烷值的损失为主要的优化目标，要求辛烷值的损失尽量小，从而保证优化后的损失相对于优化之前的降幅尽量大。基于该分析，本文提出以下目标函数：

$$\min RON_{loss} = RON_{原料} - RON_{pred}$$

（3）约束条件：

对于该优化问题，由于提出的 23 个决策变量在实际操作中存在一定的取值范围，因

此，存在约束 1：

$$lb_i < x_i < ub_i, i = 1,2,3,...,23$$

此外，考虑到预测的辛烷值含量不能超过原来中的辛烷值含量，因此有约束 2：

$$RON_{pred} < RON_{原料}$$

另外，在本问题中，需要保证产品的硫含量不大于 5μg/g，同时，在实际中，硫含量应大于等于 0ug/g，于是，提出约束 3：

$$0 \le S_{pred} \le 5$$

综上所述，约束条件如下：

$$s.t. = \begin{cases} lb_i < x_i < ub_i, i = 1,2,3,...,23 \\ RON_{pred} < RON_{原料} \\ 0 \le S_{pred} \le 5 \end{cases}$$

其中：辛烷值和硫含量的 BP 神经网络预测模型可用下列非线性函数描述：

$$RON_{pred} = f\left(x_1, x_2, x_3, ...x_{23}\right)$$
$$S_{pred} = g\left(x_1, x_2, x_3, ...x_{23}\right)$$

7.2.3 模型参数设定

依据题目要求，选用局部粒子群算法，结合相关因素，确定粒子群算法的主要变量设定值如表 7.1 粒子群算法要素设定所示

表 7.1 粒子群算法要素设定

要素名称	符号	值
种群大小	m	300
维数	D	23
权重因子	ω	0.8
学习因子	c_1, c_2	500
粒子的最大速度	$v_{\max}$	根据附件四中取值范围设定
最大迭代步数	t	500
初始化各粒子的位置	x_0	随机数
初始化各粒子的速度	v_0	随机数

7.2.4 模型求解

针对上述建立的复杂多约束优化模型，本文使用 Python 语言编写粒子群算法的程序对模型进行求解。本文对 325 个数据样本求解辛烷值（RON）损失最小时的主要变量对应的操作条件。可求得在满足约束条件下，每个样品辛烷值损失下降的最大百分比额，统计结果如表 7.2 所示。其中，满足辛烷值（RON）损失降幅大于 30%的样本数共有 103 个。

表 7.2 每个样品辛烷值损失下降的最大百分比额的统计结果

产品辛烷值损失下降百分比	包含样本数	具体样本号
50%以上	12	14、20、24、63、19、38、31 等
40%-50%	34	80、209、85、65、56、13、等
30%-40%	57	7、53、29、44、155、224 等
20%-30%	70	129、285、116、160、358、279 等
10%-20%	71	170、148、133、144、119、313 等
10%以下	81	232、270、214、267、112 等

从表中可以看出，经过优化后，31.69%的样本辛烷值损失下降百分比超过 30%，53.23%的样本辛烷值损失降低 20%以上，75.77%的样本辛烷值损失下降百分比超过 10%。

本文筛选出辛烷值（RON）损失降幅大于 30%的样本，可得到 103 组满足条件的样本，将其对应的主要变量优化后的操作条件进行汇总输出，输出结果放于“优化操作数据.xlsx”中。图 7.4 展现了其中的部分数据。

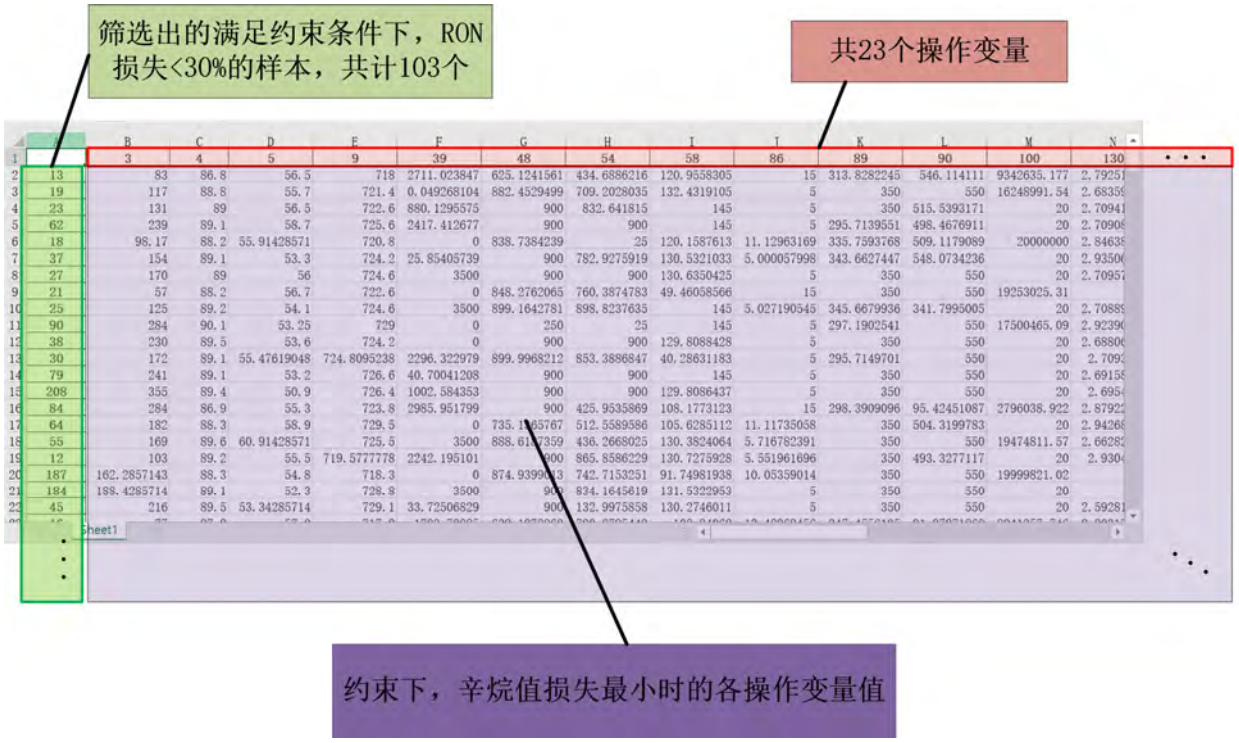

图 7.4 优化后辛烷值损失降幅大于 30%样本对应的操作变量值

7.3 模型评价（结果分析）

如图 7.5 所示是经过粒子群算法优化主要变量对应的操作条件后 325 组样本的辛烷值损失降幅情况。可以看出，样本经过粒子群算法优化，其辛烷值损失都有了不同的降幅。325 组数据中，有 31.7%的样本在优化后辛烷值损失降幅达到了 30%以上，说明这类样本的操作条件有较大的改进空间；另有 24.9%样本的辛烷值损失降幅较低，为 10%以下，可能受原料成分等因素的影响。总体来说，该粒子群优化搜索模型能够较好地对样本的主要变量的操作条件进行优化，且求解时间较短。

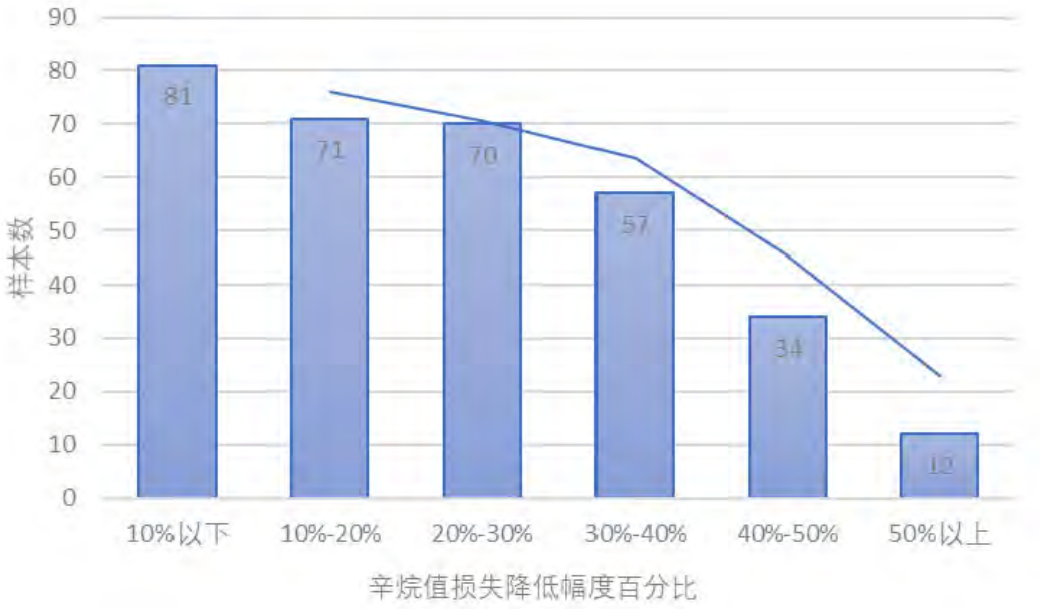

图 7.5 不同辛烷值降幅范围内所含样本数的统计

8 问题五：模型的可视化展示

8.1 问题五分析

本问要求以图形展示 133 号样本其主要操作变量优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹。实际上将前面几个问题的建模结果以图形的方式展示出来，是对整个建模结果的综合展示，只需不断调用前面开发的算法与模型绘制出相应轨迹即可。

解决这一问题的思路为：在任务三中建立了基于所提取 27 个主要变量的产品辛烷值、产品硫含量的预测模型，将样本的原料性质与操作变量输入预测模型就可以得到估计的产品辛烷值与产品硫含量。在任务四中通过优化算法确定了在样本原料性质及待生吸附剂剂和再生吸附剂性质数据保持不变前提下的最优操作操作条件。将预测模型与优化算法结合起来就可以实现模型的可视化展示，获得主要操作变量优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹。

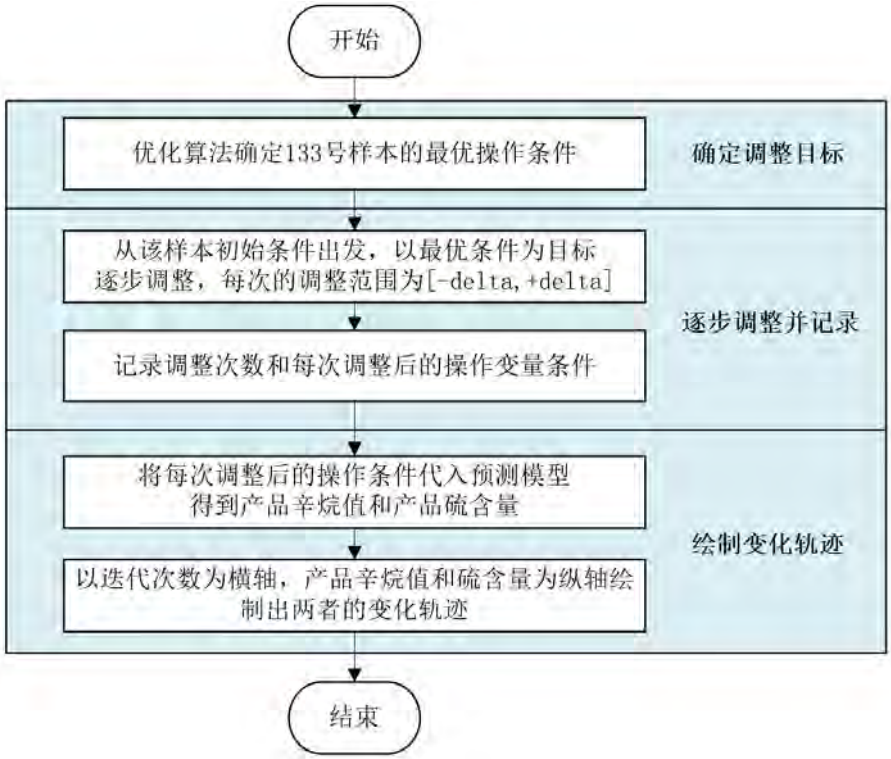

图 8.1 问题五思路流程图

具体过程如下：将 133 号样本的 27 个主要变量代入优化算法确定针对 133 号样本的最优操作条件OP_{best}，从 133 号样本的初始操作条件（记为OP_0）出发，以最优操作条件为目标逐步改变操作变量的值，各操作变量每次的改变范围的绝对值为 *0* 至附件四中提供的Δ值。第 i 次调整后的操作条件记为OP_i，其对应的产品辛烷值记为RON_i，产品硫含量记为S_i，持续调整操作变量直到所有的操作变量均达到最优值，调整结束。以 i 为横坐标，RON_i 和 S_i 为纵坐标画出要操作变量优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹。

8.2 建模结果

133 号样本的最优操作条件OP_{best}如表 8.1 133 号样本初始条件及最优条件所示。操作变量优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹如图 8.2 主要操作变量调整过程中产品辛烷值与硫含量的变化轨迹图所示。

表 8.1 133 号样本初始条件及最优条件

名称	原始条件	最优条件	操作范围	变化量
还原器温度	261.8852	349.78	200-350	1
催化汽油进装置总流量	122.7257	844.26	25-900	50
净化风进装置流量	111.8784	129.89	40-145	5
轻烃出装置流量	1756.7436	1412.56	0-3500	50
K-101B 进气压力	2.2379	0.9	0-2.5	0.05
循环水进装置流量	441.0867	900	250-900	50
E203 重沸器管程出口凝结水流量	2291.2150	831.56	500-3800	1
D121 去稳定塔流量	11.2819	5.82	5-15	1
D-110 底流化 N2 流量	40.7268	15	15-50	5
P-101B 入口过滤器差压	8.8463	5.16	-1.5-12	1
K-103 出口去 K-101 出口管流量	1352.1851	793.69	100-3500	100
再生风流量	280.3438	545.87	90-550	50
E-206 壳程出口管温度	88.8064	113.13	50-150	1
再生器顶部/再生器接收器差压	113.2382	150	-120-(150)	10
稳定塔底出口温度	134.05572	150	100-150	1
紧急氢气去 R-101 流量	-0.6188	-7.42	-8-75	5
闭锁料斗氧含量	0.1933	0.12	-0.5-5	1
火炬气排放流量	505.4965	20	20-20000000	1000
火炬罐 D-206 液位	7.1313	0.12	-5-(35)	5
EH-103 加热元件温度	358.9112	550	300-550	1
D-203 顶部出口管温度	31.0319	26.71	10-50	1
E-101 管程入口总管压力	2.6893	2.71	2.55-2.95	0.1
R-102 底滑阀差压	10.1016	-0.25	-0.5-20	1

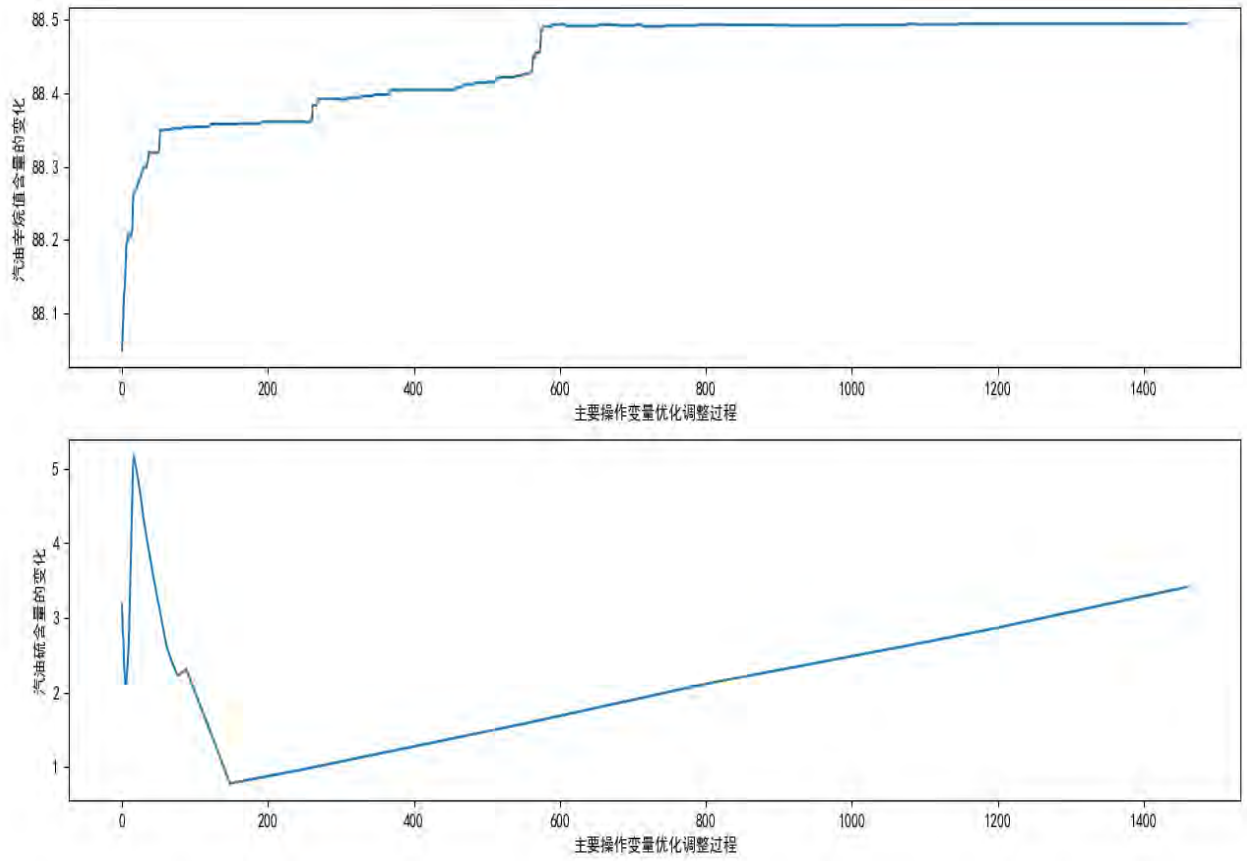

图 8.2 主要操作变量调整过程中产品辛烷值与硫含量的变化轨迹图

分析图 8.2 可知：133 号样本的产品辛烷值初始值为 88.09 并随着操作条件逐步调整，在其从 0 到第 60 次的调整过程中，产品辛烷值含量迅速上升达到约 88.35；在从 50 到第 600 次的调整过程中，此范围内产品辛烷值的含量缓慢提高；在调整次数到达第 600 次之后，产品辛烷值含量以极低的速率向最优值逼近，共经过约 1450 次调整后达到最优值。

对其优化效果分析：该样本的原料辛烷值为 89.4，产品辛烷值为 88.09，辛烷值损失为 1.3，经过优化后产品辛烷值为 88.48，辛烷值损失变为 0.92，相比原来的值降低了 30.54%，说明优化效果较为优秀。

8.3 结果分析

（1）产品辛烷值按照此趋势变化的原因可能为：在主要操作条件调整的初期，优化空间很大，各个操作变量每步都以最大调整幅度向最佳条件改变，这导致产品辛烷值迅速提高；随着调整次数的提高，一些关键操作变量已经调整到最优条件，这时每步调整仅能调整次关键的操作变量导致上升速度下降；到后期，几乎所有操作条件都已经达到最优值，仅剩最后一两个初始距离大的值需要更多调整次数来达到最优值，这个阶段产品的辛烷值以极小的速率上升，随着迭代结束到达最优值。

（2）产品硫含量按照此趋势变化的原因可能为：因优化算法的目标是找使产品辛烷值达到最大，产品硫含量小于 5 微克/克只是一个约束条件，只要求满足限定范围即可，同

时使产品辛烷值达到最优的条件不一定能降低产品的硫含量，24 个操作变量对产品硫含量量的影响错综复杂，所以产品硫含量随着调整次数的轨迹曲线存在很大的不确定性。

（3）关于图示调整次数的说明：E203 重沸器管程出口凝结水流量作为本文选择的主要变量之一，它的变化范围 500-3800。但是每次可调整的量仅为 1，133 样本中这个变量的初始条件为 2291.2150，而其最优条件为 831.56，按照其步长要求进行调整，约 1460 次才能达到其理想值，而其他变量所需调整次数均在 100 次以下，故导致了后期随着调整次数过多辛烷值不再明显改变。

9 模型评价与改进

9.1 模型优点

1) 充分考虑了各变量之间，各变量与产品辛烷值之间的非线性关系，使用了随机森林回归、距离相关系数等适用于处理非线性特征的方法。所获得的主要变量物理意义明确，符合实际。
2) 针对主要变量与产品辛烷值之间复杂的关系，选了 BP 神经网络机器学习算法，同时数据集与测试集分开，所得到的结果最大相对误差很小，所建立的预测模型精度、鲁棒性表现优秀。
3) 在进行粒子群算法的设计过程中充分考虑了变量范围、产品辛烷值范围和产品硫含量范围的约束，迭代搜寻的结果满足所有约束要求。
4) 算法速度快，响应性好。

9.2 模型缺点

1) 分析问题时认为所有操作变量均可以独立改变，这可能与实际不符。
2) 所选取主要变量只考虑了对产品辛烷值的贡献，未考虑对产品硫含量的贡献，将两者综合考虑完成变量选取的效果可能会更好。
3) 本文构建的 BP 神经网络预测模型训练样本数较少，日后可获得更多数据对其进行修正。

9.3 模型的改进与推广

本文提出的方法和模型可推广应用于其他工业生产流程中的操作变量优化问题：

1) 本文中数据预处理方法及操作因其分类细化且处理科学，在其他相关研究遭遇数据质量问题时均可对应采纳、使用。
2) 所采用的的降维和预测方法适用性强，应用面广，对以后的其他相关研究具有较强的现实参考意义。

10 参考文献

[1] 牟明仁, 李百舸, 刘冉, et al. 车用汽油标准 GB17930—2016 与 DB11/238—2016 内容对比分析[J]. 石油商技, 2017, 000(004): 42-45.

[2] 左明. 车用汽油的清洁化进程[J]. 炼油, 2001(1): 1-8.

[3] 王廷海, 李文涛, 常晓昕, et al. 催化裂化汽油清洁化技术研究开发进展[J]. 化工进展, 2019, 38(01): 203-214.

[4] 李亚芬. 我国石油供求现状,问题及改善对策[J]. 国际金融研究, 2005, 000(003): 65-70.

[5] Breiman L. Random Forests[J]. Machine Learning, 2001, 45(1): 5-32.

[6] Breiman L. Bagging Predictors[J]. Machine Learning, 1996, 24(2): 123-140.

[7] Ho T K. The random subspace method for constructing decision forests[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1998, 20(8): 832-844.

[8] Székely G J, Rizzo M L, Bakirov N K. Measuring and testing dependence by correlation of distances[C]. Acm Symposium on Virtual Reality Software & Technology, 2007.

[9] Rumelhart D E, Hinton G E, Williams R J. Learning Internal Representation by Back-Propagation Errors[J]. Nature, 1986, 323: 533-536.

11 附录

程序编号	1	程序说明	程序中的自定义工具函数	编程软件	Python

```
# 程序中使用到的自定义工具函数
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import joblib
import pandas as pd

# 读取文件的函数
def read_data(path: str):
    data = pd.read_excel(path, header=1)  # 原始数据，包含表头
    s = data[0]  # 硫含量
    RON = data[1]  # 辛烷值
    factors = data.drop(columns={0, 1, 2})  # 其它因素
    factors = factors.dropna(axis=1)  # 去除空值列
    return s, RON, factors
```

```
    # 获得选取的主要变量的数据
    def use_data(data, column_set: list):
        return pd.DataFrame(data, columns=column_set)

    # 训练神经网络模型
    def train_mlp(X_train, y_train, num=300):
        # 数据标准化
        scaler = StandardScaler()
        scaler.fit(X_train)
        X_train = scaler.transform(X_train)

        # 定义神经网络
        clf = MLPRegressor(hidden_layer_sizes=(num,), random_state=0, max_iter=10000,
shuffle=True)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_train)
        mse = mean_squared_error(y_train, y_pred)
        print(mse)
        print(clf.score(X_train, y_train))
        return clf, scaler

    def train_rf(X_train, y_train):
        # 定义随机森林
        clf = RandomForestRegressor(n_estimators=500,
                                    random_state=0,
                                    max_features="auto",
                                    n_jobs=2)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_train)
        mse = mean_squared_error(y_train, y_pred)
        print(mse)
        print(clf.score(X_train, y_train))
        return clf

    # 保存模型
    def save_model(clf, name, scaler=None):
        joblib.dump(clf, "./model_1/" + name + ".pkl")
        if scaler:
            joblib.dump(scaler, "./model_1/" + name + ".s")

    # 加载模型
    def load_model(name, scaler=None):
        model = joblib.load("./model_1/" + name + ".pkl")
        if scaler:
```

```
            s = joblib.load("./model_1/" + name + ".s")
            return model, s
        return model


    # 保存粒子群算法优化的样本操作参数
    def save_pso_data(pso, i: int):
        joblib.dump(pso, "optimization_data/" + "sample_" + str(i) + ".con")
```

程序编号	2	程序说明	距离相关系数计算	编程软件	Python

```
    # 变量之间的距离相关系数计算
    # 相关性分析
    from scipy.spatial.distance import pdist, squareform
    import numpy as np
    import pandas as pd


    def distcorr(X, Y):
        X = np.atleast_1d(X)
        Y = np.atleast_1d(Y)
        if np.prod(X.shape) == len(X):
            X = X[:, None]
        if np.prod(Y.shape) == len(Y):
            Y = Y[:, None]
        X = np.atleast_2d(X)
        Y = np.atleast_2d(Y)
        n = X.shape[0]
        if Y.shape[0] != X.shape[0]:
            raise ValueError('Number of samples must match')
        a = squareform(pdist(X))
        b = squareform(pdist(Y))
        A = a - a.mean(axis=0)[None, :] - a.mean(axis=1)[:, None] + a.mean()
        B = b - b.mean(axis=0)[None, :] - b.mean(axis=1)[:, None] + b.mean()

        dcov2_xy = (A * B).sum() / float(n * n)
        dcov2_xx = (A * A).sum() / float(n * n)
        dcov2_yy = (B * B).sum() / float(n * n)
        dcor = np.sqrt(dcov2_xy) / np.sqrt(np.sqrt(dcov2_xx) * np.sqrt(dcov2_yy))
        return dcor
```

```
path = r'E:\华为杯数学建模\data1.xlsx'
raw_data = pd.read_excel(path)  # 原始数据，包含表头
data = raw_data.values  # 去除表头的数据

s = data[:, 0]  # 硫含量
RON = data[:, 1]  # 辛烷值
factors = data[:, 3:]  # 其它因素
m, n = factors.shape
corrs = np.zeros((n, n))
for i in range(n):
    print(i)
    for j in range(i, n):
        corrs[i, j] = distcorr(factors[:, i], factors[:, j])

print(corrs)
frame = pd.DataFrame(corrs)
frame.to_excel(r'E:\华为杯数学建模\corr.xlsx', "Sheet2")
```

程序编号	3	程序说明	通过随机森林评价变量重要性	编程软件	Python

```
# 通过随机森林分离变量对辛烷值含量的重要性
from time import time
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import pandas as pd

path = r'E:\华为杯数学建模\325 个样本原始数据.xlsx'
data = pd.read_excel(path,header=1)  # 原始数据，包含表头
s = data[0]
RON = data[1]  # 辛烷值
factors = data.drop(columns={0, 1, 2})  # 其它因素
factors = factors.dropna(axis=1)



t0 = time()
forest = RandomForestRegressor(n_estimators=500,
                               random_state=0,
                               max_features=100,
                               n_jobs=2)
X_train, X_test, y_train, y_test = train_test_split(factors, RON,
test_size=0.3,shuffle=True,random_state=0)

print("done in %0.3fs" % (time() - t0))
```

```
    importances = forest.feature_importances_
    print(importances)
print(sorted(importances, reverse=True))
```

程序编号	4	程序说明	神经网络模型的训练	编程软件	Python

```
    import Utils

    path = r'E:\华为杯数学建模\325 个样本原始数据.xlsx'
    s, RON, factors = Utils.read_data(path)
    column_set = [4, 5, 3, 89, 58, 68, 54, 39, 290, 129, 181, 289, 287, 120, 83, 179, 288, 61, 48,
6, 63, 49, 319, 86, 77, 317, 291, 199, 160, 260]

    factors = Utils.use_data(factors, column_set)
    X_train,y_train=factors,RON
    clf = Utils.train_mlp(X_train, y_train)
    Utils.save_model(clf,"model_1")
```

程序编号	5	程序说明	粒子群算法优化样本的主要变量	编程软件	Python

```
    import pandas as pd
    import Utils
    import numpy as np
    from multiprocessing import Process
    import PSOUtils

    def demo_func(x, g_data):
        serch_factor = np.concatenate((g_data, x), axis=1)  # 搜索的特征（n_sample,n_dim）
        pred = model.predict(serch_factor)  # 辛烷预测值
        std_serch_factor = scaler.transform(serch_factor)  # 特征值标准化
        s_pred = mlp_model.predict(std_serch_factor)  # 硫的预测值
        loss = np.zeros(pred.shape)
        for i in range(len(pred)):
            if (pred[i] > g_data[0][1]) or (s_pred[i] > 5) or (s_pred[i] < 0):
                loss[i] = 100
            else:
                loss[i] = g_data[0][1] - pred[i]
        return loss

    # 样本数据路径
    sample_data_path = r'E:\华为杯数学建模\325 个样本原始数据.xlsx'
    # 特征集合
    column_set = [3, 4, 5, 9, 39, 48, 54, 58, 86, 89, 90, 100, 130, 138, 160, 184, 192, 209, 220,
222, 231, 241, 260,
                  270, 290, 309, 319]
```

```
    g_index = [x for x in column_set if x <= 13]  # 固定参数
    v_index = [x - 14 for x in column_set if x >= 14]  # 可变参数
    # 读取变量的范围数据
    range_file = r'E:\华为杯数学建模\变量取值范围.xlsx'
    variable_range = pd.read_excel(range_file).values  # 可变参数的取值范围
    lb = variable_range[v_index, 0]
    ub = variable_range[v_index, 1]
    s, RON, factors = Utils.read_data(sample_data_path)

    # 加载模型
    model = Utils.load_model("RON_model_rf_1")
    mlp_model, scaler = Utils.load_model("s_model_mlp_1", "s_model_mlp_1")

    def main(piece_index):
        # 优化参数
        pop_num = 300  # 种群数量
        gd_data = np.zeros((pop_num, 4))  # 固定参数
        data_piece = factors.loc[piece_index]  # 第 index 条样本
        # 填充固定参数值
        for k in range(pop_num):
            for i, each in enumerate(column_set):
                if each <= 13:
                    gd_data[k][i] = data_piece[each]

        pso = PSOUtils.PSO(data=gd_data, func=demo_func, dim=23, pop=pop_num,
max_iter=200, lb=lb, ub=ub, w=0.8, c1=0.5,
                           c2=0.5)
        pso.run()
        # print(pso.gbest_y)
        Utils.save_pso_data(pso, piece_index)

    sample1 = list(range(0, 100))
    sample2 = list(range(100, 200))
    sample3 = list(range(200, 325))
    print(len(sample1) + len(sample2) + len(sample3))

    def pro(samples):
        while len(samples) > 0:
            element = samples.pop()
            print(element)
            main(element)

    if __name__ == '__main__':
```

```
        print("开始执行.....")
        p1 = Process(target=pro, args=(sample1,))
        p2 = Process(target=pro, args=(sample2,))
        p3 = Process(target=pro, args=(sample3,))
        p1.start()
        p2.start()
        p3.start()
        p1.join()
        p2.join()
        p3.join()
```

程序编号	6	程序说明	133 号样本的辛烷值/硫含量轨迹	编程软件	Python

```
    import joblib
    import matplotlib.pyplot as plt
    import pandas as pd
    import numpy as np
    import Utils

    def demo_func(x, g_data):
        return None

    def load_pso_data(i: int):
        pso = joblib.load("optimization_data/sample_" + str(i) + ".con")
        return pso.gbest_x, pso.gbest_y

    pso = joblib.load("sample_" + str(132) + "_data")
    fanal_param = pso.gbest_x   # 最终需要的操作变量

    column_set = [3, 4, 5, 9, 39, 48, 54, 58, 86, 89, 90, 100, 130, 138, 160, 184, 192, 209, 220,
222, 231, 241, 260,
                  270, 290, 309, 319]
    cons_index = [x for x in column_set if x <= 13]
    var_index = [x for x in column_set if x >= 14]
    # 样本数据路径
    sample_data_path = r'E:\华为杯数学建模\325 个样本原始数据.xlsx'
    s, RON, factors = Utils.read_data(sample_data_path)
    cons_param = factors[cons_index].values[132]   # 固定参数值
    var_param = factors[var_index].values[132]   # 可变参数初始值

    delta_file = r"E:\华为杯数学建模\数模题\附件四：354 个操作变量信息.xlsx"
    data = pd.read_excel(delta_file)["Δ 值"].values
    data = [abs(x) for x in data]
```

```
var_delta = []    # 保存变量的单位变化
for each in column_set:
    if each >= 14:
        var_delta.append(data[each - 14])


def find_near_num(t0, t1, d):
    temp = [t0]
    if t0 < t1:
        while t0 < t1:
            t0 += d
            temp.append(t0)
        if temp[-1]>t1:
            temp[-1]=t1
        return temp
    if t0 > t1:
        while t0 > t1:
            t0 -= d
            temp.append(t0)
        if temp[-1] < t1:
            temp[-1] = t1
        return temp
    return temp

state = []
for i in range(len(fanal_param)):
    s1 = find_near_num(var_param[i], fanal_param[i], var_delta[i])
    state.append((var_index[i], s1))

state = dict(state)    # 保存了每个变量的状态变化

max_length = []
for each in var_index:
    max_length.append(len(state[each]))

max_length = max(max_length)

all_state = []
for i in range(max_length):
    temp = []
    for each in var_index:
        length = len(state[each]) - 1
        if i > length:
```

```
                    temp.append(state[each][length])
                else:
                    temp.append(state[each][i])
        temp = np.array(temp)
        all_state.append(np.concatenate((cons_param, temp), axis=0))
all_state = np.array(all_state)

# 加载模型
model = Utils.load_model("RON_model_rf_1")
mlp_model, scaler = Utils.load_model("s_model_mlp_1", "s_model_mlp_1")

RON_point = model.predict(all_state)

std_all_state = scaler.transform(all_state)
s_point = mlp_model.predict(std_all_state)

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title("汽油辛烷值和硫含量的变化轨迹")
plt.subplot(2, 1, 1)
plt.plot(RON_point)
plt.xlabel("主要操作变量优化调整过程")
plt.ylabel("汽油辛烷值含量的变化")
plt.subplot(2, 1, 2)
plt.plot(s_point)
plt.xlabel("主要操作变量优化调整过程")
plt.ylabel("汽油硫含量的变化")
plt.show()
```