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	北京邮电大学
参赛队号	20100130029
队员姓名	1. 唐麒淳
	2. 段祥卿
	3. 戴维

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　降低汽油精制过程中的辛烷值损失模型

摘　　要：

本文的建立对于优化汽油精制过程中的操作条件，从而带来巨大经济效益和环境保护效益具有重要参考意义。本文运用机器学习与贝叶斯优化等方法对 RON 损失模型建模问题展开研究，首先考虑不同场景设计多种策略对数据进行处理，并用特征筛选方法找到建模的 12 个主要变量，然后构建机器学习管线对主要变量进行建模，训练得到具有泛化能力且高鲁棒的机器学习模型，其在 5 折交叉验证过程中在验证集的 r^2 评级指标为 0.373。用这个机器学习模型作为评价函数，通过贝叶斯优化算法的 TPE 算法对 325 个样本的 RON 损失值进行优化，在产品硫含量不大于 5μg/g 的约束条件下使得 16.92% 的变量 RON 损失降幅大于 30%。对 133 号样本优化后其 RON 损失降幅为 32.61% 。在问题五中，使用创新性的两步 A* 算法找到了逐步调整主要操作变量的调整路径。

问题一：本问题首先根据时序上是否平稳来决定 313 与 285 号样本的空值填充策略，然后通过公式定义操作变量越界程度并以此为阈值删除采样样本，用 3σ 准则删除异常的采样样本。最后根据不同场景制定相应的空值处理策略。通过后续建模过程较高的评价指标验证了前导数据处理流程基本正确。

问题二：本问题采用特征筛选方法来寻找建模主要变量，用嵌入式的特征打分模型梯度提升树对特征进行评价，并通过对比不同的特征筛选次数进一步限定主要变量的范围，最后筛选得到了 12 个主要变量。通过对所得主要变量的数据分析结果和特征重要度可视化结果可以验证主要变量筛选过程的正确性。

问题三：本问题基于分布转换与 LightGBM 模型在 12 个主要变量的基础上建立 RON 损失模型。首先甄别出离群的异常样本并将其删除，提升了模型的综合表现。然后建立机器学习管线并进行交叉验证，通过交叉验证的结果可以验证所得模型的鲁棒性与可泛化性。

问题四：本问题首先根据问题三的方法建立一个对硫的预测模型，并根据产品硫含量不大于 5μg/g 的约束条件重新设计了评价函数，通过贝叶斯优化的 TPE 算法对 325 个样本的 RON 损失值进行优化，并得到了 RON 损失降幅大于 30% 样本主要变量优化后的操作条件。

问题五：本问题首先将 11 个主要操作变量视为高维空间中的坐标点，初始坐标即为原始操作变量值，目的坐标即为问题三所求的优化值。通过 A* 算法逐步调整初始坐标点在每一维上的坐标值，最终找到一条由起始点到达终点的路径。在此过程中，本队创新性地将 A* 算法分为两步进行，这样极大地减少了运算量。

关键字：　基于模型的特征筛选　梯度提升树　贝叶斯优化　TPE 算法　A* 算法

目录

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。为此，世界各国都制定了日益严格的汽油质量标准。汽油清洁化重点是降低汽油中的硫、烯烃含量，同时尽量保持其辛烷值。

我国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。辛烷值每降低 1 个单位，相当于损失约 150 元/吨。以一个 100 万吨/年催化裂化汽油精制装置为例，若能降低 RON 损失 0.3 个单位，其经济效益将达到四千五百万元。

化工过程的建模一般是通过数据关联或机理建模的方法来实现的，取得了一定的成果。但是由于炼油工艺过程的复杂性以及设备的多样性，它们的操作变量（控制变量）之间具有高度非线性和相互强耦联的关系，而且传统的数据关联模型中变量相对较少、机理建模对原料的分析要求较高，对过程优化的响应不及时，所以效果并不理想。

某石化企业的催化裂化汽油精制脱硫装置运行 4 年，积累了大量历史数据，其汽油产品辛烷值损失平均为 1.37 个单位，而同类装置的最小损失值只有 0.6 个单位。故有较大的优化空间。请参赛研究生探索利用数据挖掘技术来解决化工过程建模问题。

1.2 问题提出

问题 1：数据处理

请参考近 4 年的工业数据的预处理结果，依“样本确定方法”对 285 号和 313 号数据样本进行预处理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 2：寻找建模主要变量

由于催化裂化汽油精制过程是连续的，虽然操作变量每 3 分钟就采样一次，但辛烷值（因变量）的测量比较麻烦，一周仅 2 次无法对应。但根据实际情况可以认为辛烷值的测量值是测量时刻前两小时内操作变量的综合效果，因此预处理中取操作变量两小时内的平均值与辛烷值的测量值对应。这样产生了 325 个样本。建立降低辛烷值损失模型涉及包括 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 个产品性质等变量以及另外 354 个操作变量（共计 367 个变量），工程技术应用中经常使用先降维后建模的方法，这

有利于忽略次要因素，发现并分析影响模型的主要变量与因素。因此，请你们根据提供的325 个样本数据，通过降维的方法从 367 个操作变量中筛选出建模主要变量，使之尽可能具有代表性、独立性（为了工程应用方便，建议降维后的主要变量在 30 个以下），并请详细说明建模主要变量的筛选过程及其合理性。

问题 3：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 4：主要变量操作方案的优化

要求在保证产品硫含量不大于 5μg/g 的前提下，利用模型获得 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 5：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，请你们对133 号样本，以图形展示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。

2. 模型假设

1. 题干描述的空值为 0。
2. 不考虑本题所给数据以外信息对辛烷值损失与硫含量的影响。
3. 变量的取值范围由附件四的规定值及附件一的实际取值共同决定。
4. 去除异常数据后，不影响数据的整体分布情况。
5. 选择的特征参数集能够反映原数据的主要特征。
6. 问题 3 训练的模型能反映操作变量与产品性质之间的关系。
7. 问题五假设前文搭建的硫产量及辛烷值损失预测模型可以进行准确预测。
8. 问题五假设变量的取值范围由附件四的规定值及附件一的实际取值共同决定

3. 符号说明

表 1　论文中用到的符号定义

符号	意义
r^2	r^2 评价指标
v	剩余误差
$\rho_{X,Y}$	X, Y 的相关系数
σ	标准差
$\mathbb{R}^n$	n 维实数域
$\mathbb{H}_S$	硫预测模型的参数空间
$\vec{d}$	起始点到终点的向量
$\vec{step}$	起始点到终点的步长向量
CV	交叉验证（Cross Validation）
GBDT	梯度提升数（Gradient Boosting Decision tree）
RFE	递归特征消除（Recursion Feature Elimination）
EI	期望提升（Expected Improve）

4. 问题分析与求解

4.1 问题一：数据处理

4.1.1 对含空值变量的分析

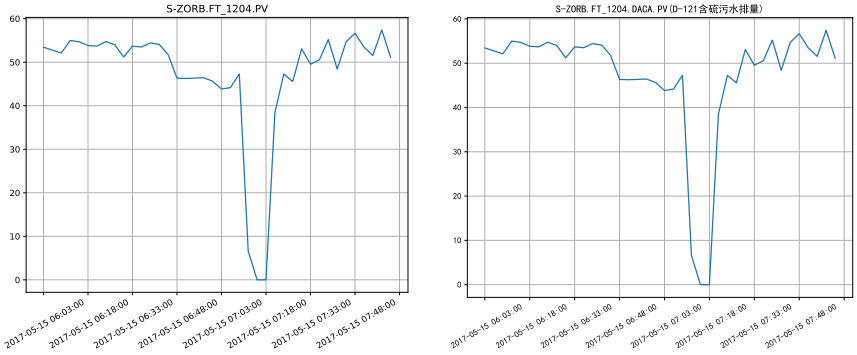
考虑到工业设备传感器失效或异常时记录值为 0，故在此认为 0 代表了工业数据中的空值。

经过分析可以发现 285 号样本的“新氢进装置流量”、“1# 催化汽油进装置流量”等操作变量全部为空值 0，但其余操作变量不含空值。考虑到最后需要取其前 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，所以最后汇总得到的相应的操作变量值仍是 0，故暂时不对这类变量做处理，留到章节4.1.4中处理。

经过分析，容易发现 313 号样本除了“新氢进装置流量”等操作变量与上文描述的一样也全部为空值 0 以外，还有部分操作变量只含有部分空值。对于这类操作变量，制定了 2 种处理策略：

处理 1：对时序上基本平稳的含空值变量做均值填充

在 313 号样本的数据中，容易发现 2 个时序上基本平稳但含有空值的变量。考虑到本题确定某个样本的方法为：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。故对于这两个变量的空值用 313 号样本数据的其余采样相应变量的均值来填充。

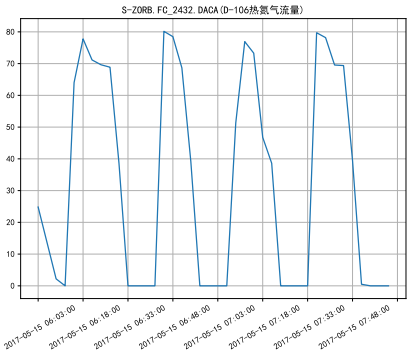


(a) S-ZORB.FT_1204.PV **(b) D-121 含硫污水排量**

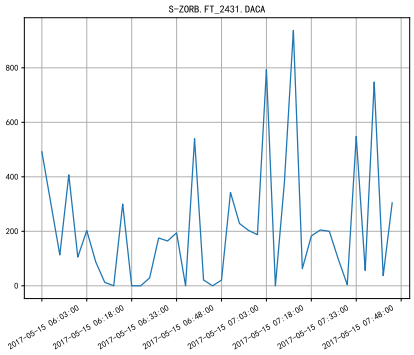
图 1　313 号样本处理 1 中的 2 个变量

处理 2：对时序上杂乱或呈周期趋势的含空值变量做删除操作

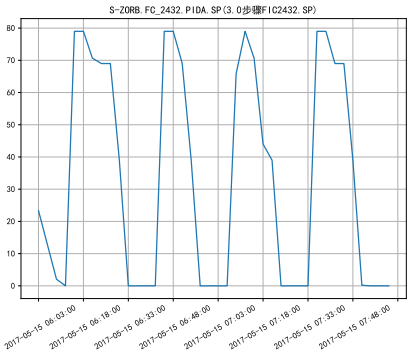
在 313 号样本的数据中，容易发现 3 个时序上杂乱或呈周期趋势的含空值变量但含有空值的变量。考虑到本题对于只含有部分时间点的位点，如果其残缺数据较多，无法补充，应将此类位点删除，所以将 313 号样本的这 3 个变量设为 NaN，留到章节4.1.4中处理。



(a) D-106 热氮气流量



(b) S-ZORB.FT_2431.DACA



(c) 3.0 步骤 FIC2432.SP

图 2　313 号样本处理 2 中的 3 个变量

4.1.2 剔除不在操作范围内的样本

考虑到“325 个样本数据”中的很多样本已经超出了“354 个操作变量信息”所规定的范围，于是应该进行简单的处理，用“325 个样本数据”中每个操作变量的最大与最小值来扩充“354 个操作变量信息”中操作变量的范围，并用扩充后的操作变量范围代替原操作变量范围。

经过计算，285 样本数据中所有的采样数据都在操作变量范围内，而 313 样本数据仅有 1 个采样数据的所有变量在操作变量范围内。经过综合考虑，决定用以下公式来计算一个采样样本操出操作变量范围的程度：

$$InvalidDegree = \sum_{j}^{M} \frac{Exceed_j}{Upper_j - Lower_j} \tag{1}$$

其中，j 表示某采样样本的第 j 个操作变量，M 表示共有 M 个操作变量，$Upper_j$ 表示操作变量 j 的上界，$Lower_j$ 表示操作变量 j 的下界，$Exceed_j$ 表示操作变量 j 超出范围的大小。

使用上述公式，对 313 号样本的采样数据进行计算，结果如表2：

经过综查阅相关文献和讨论，在此决定以 $InvalidDegree >= 1$ 为阈值，删除满足其

表 2　313 样本超出范围程度

time	invalid-degree	rank
2017-05-15 06:57:00	1.986283	0
2017-05-15 07:24:00	1.868216	1
2017-05-15 07:18:00	1.533825	2
2017-05-15 06:33:00	1.514671	3
2017-05-15 06:54:00	1.298304	4
2017-05-15 06:51:00	1.057390	5
2017-05-15 07:21:00	0.876193	6
2017-05-15 07:51:00	0.866200	7
2017-05-15 07:48:00	0.770276	8
2017-05-15 07:54:00	0.658065	9

条件的所有采样样本。

4.1.3 用 3σ 准则删除异常样本

3σ 准则 [**?**]：设对被测量变量进行等精度测量，得到 $x_1, x_2, \ldots, x_n$，算出其算术平均值 x 及剩余误差 $v_i = x_i - x(i = 1, 2, \ldots, n)$，并按贝塞尔公式算出标准误差 σ，若某个测量值 x_b 的剩余误差 $v_b(1 <= b <= n)$，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。贝塞尔公式如下：

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

根据上述公式，285 号样本的所有采样都满足 3σ 准则，而 313 号样本有 27 个采样不满足其条件，故删除这些不满足条件的变量。

4.1.4 对工业数据中的空值变量进行处理

通过前述操作，将 313 号样本和 285 号样本经过处理后加入到 325 个样本的工业数据中，并将章节4.1.1处理 2 中删除的变量设置为 NaN ，待进一步的处理。

对于不同类型的情况，制定 3 种处理空值的策略：

$$\begin{cases} delete\ column, if\ len(EmptyElements) > 50\ and\ mean(vector) > 5 \\ delete\ element, if\ len(EmptyElements) < 50\ and\ mean(vector) > 5 \\ do\ not\ process, if\ mean(vector) <= 5 \end{cases} \tag{3}$$

根据公式(3)，对于工业数据中的空值有 3 种处理策略：

策略 1 如果某列空值元素的长度 $len(EmptyElements)$ 大于 50，并且这列元素的均值 $mean(vector)$ 大于 5，说明空值较多，做空值填充的意义不大，应该将此列删除。

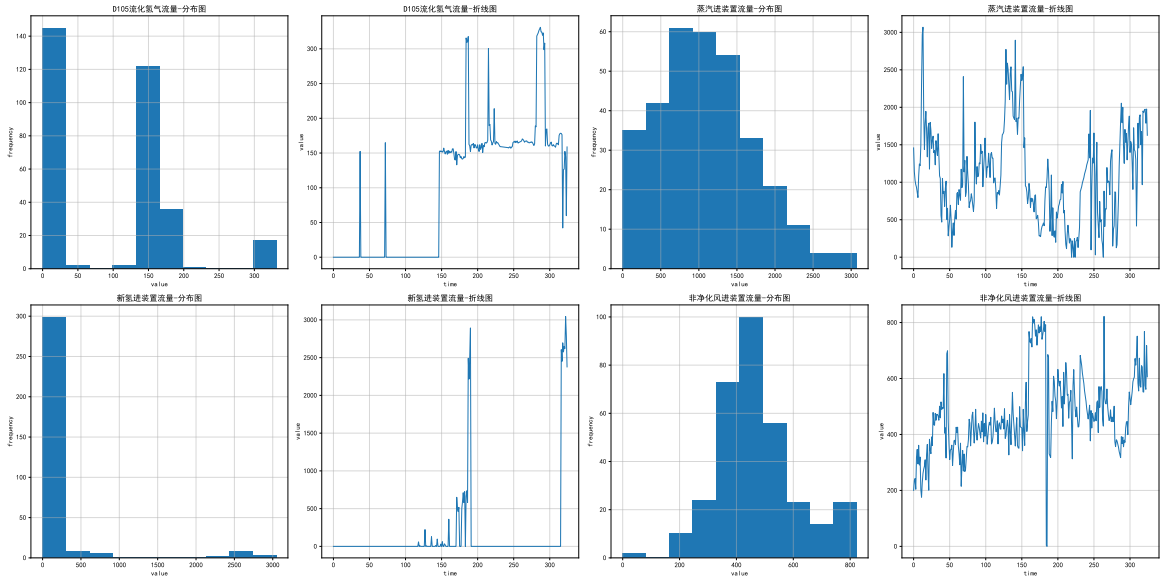
策略 2 如果某列空值元素的长度小于 50，并且这列元素的均值大于 5，说明空值相对较少，可以通过做空值填充保留下来。

策略 3 如果这列元素的均值小于等于 5，说明这列元素基本为 0，0 可能不是这列元素的空值，所以不做处理。

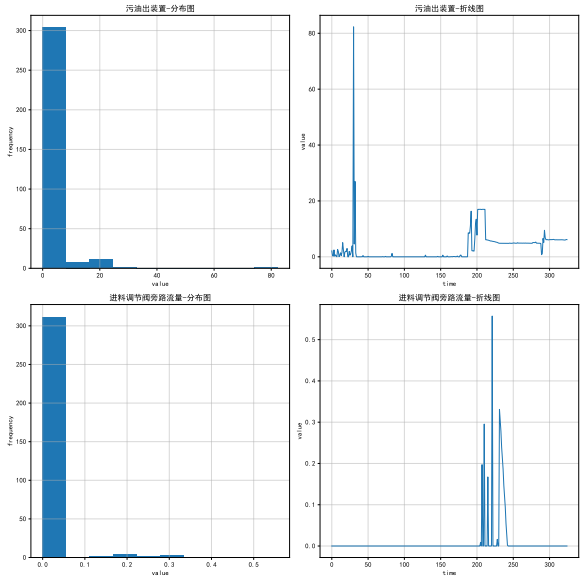
三种策略的处理结果数据如下：

对于策略 2 与4.1.1中填充的 NaN，用每列其余变量的均值代替，具体方法用的是 `scikit-learn` 的 `impute.SimpleImpute` 。

经过上述处理后，删除了 20 列特征，得到了一个 325 行 347 列的数据预处理结果，供后续问题处理。



(a) 策略 1　　**(b) 策略 2**



(c) 策略 3

图 3　3 种处理空值的策略

4.2 问题二：寻找建模主要变量

4.2.1 问题分析

建立降低辛烷值损失模型涉及包括 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 个产品性质等变量以及另外 354 个操作变量（共计 367 个变量），工程技术应用中经常使用先降维后建模的方法，这有利于忽略次要因素，发现并分析影响模型的主要变量与因素。

寻找主要变量的方法有两类，一类方法是主成分分析（Principal Component Analysis,

PCA）[**?**]，其所要做的就是设法将原来众多具有一定相关性的变量，重新组合为一组新的相互无关的综合变量来代替原来的变量。通常，数学上的处理方法就是将原来的变量做线性组合，作为新的综合变量。另一类方式是特征筛选 [**?**]，通过剔除无关变量与混淆变量，保留主要变量与关键变量，从而提升后续机器学习建模过程的表现。

在本题中，虽然 PCA 方法也能起到降维的作用，但考虑到问题 3 与问题 4 需要对主要变量进行优化，而不是 PCA 计算得到的主成分变量，所以决定在此使用特征筛选方法寻找建模主要变量。

在本题中，虽然有大量的特征可使用，有的特征携带的信息丰富，有的特征携带的信息有重叠，有的特征则属于无关特征，如果所有特征不经筛选地全部作为训练特征，经常会出现维度灾难问题，甚至会降低模型的准确性。因此，必须需要进行特征筛选，排除无效/冗余的特征，把有用的特征挑选出来作为模型的训练数据。

4.2.2 通过特征筛选获取主要变量

特征筛选的方法又分为 3 类，分别是 Filter 方法（过滤式），Wrapper 方法（封装式）和 Embedded 方法（嵌入式）：

1. 过滤式：主要思想是对每一维特征“打分”，即给每一维的特征赋予权重，这样的权重就代表着该特征的重要性，然后依据权重排序。主要方法有卡方检验（Chi–squared Test），信息增益（Information Gain），相关系数（Correlation Coefficient Scores）等方法。
2. 封装式：主要思想是将子集的选择看作是一个搜索寻优问题，生成不同的组合，对组合进行评价，再与其他的组合进行比较。这样就将子集的选择看作是一个优化问题，这里有很多的优化算法可以解决，尤其是一些启发式的优化算法，如 GA[**?**]、PSO[**?**] 等。最为经典的代表是 `scikit-learn` 实现的递归特征消除法（Recursion Feature Elimination, RFE）
3. 嵌入式：主要思想是在模型既定的情况下学习出对提高模型准确性帮助最大的特征，也就是在确定模型的过程中，挑选出那些对模型的训练有重要意义的特征。主要方法有用带有 L1 正则化的项完成特征选择（也可以结合 L2 惩罚项来优化）、随机森林平均不纯度减少法/平均精确度减少法。

经过实践，发现虽然过滤式虽然计算速度快，但是并不能准确地筛选出关键特征；封装式虽然筛选效果好，但是计算速度很慢；嵌入式的筛选效果与封装式基本持平，但计算速度相对快很多，故选择嵌入式特征筛选方法。

GBDT 模型能够在建模过程中计算各个特征的特征重要度。在 GBDT 模型中，特征 j 的全局重要度通过特征 j 在单颗树中的重要度的平均值来衡量：

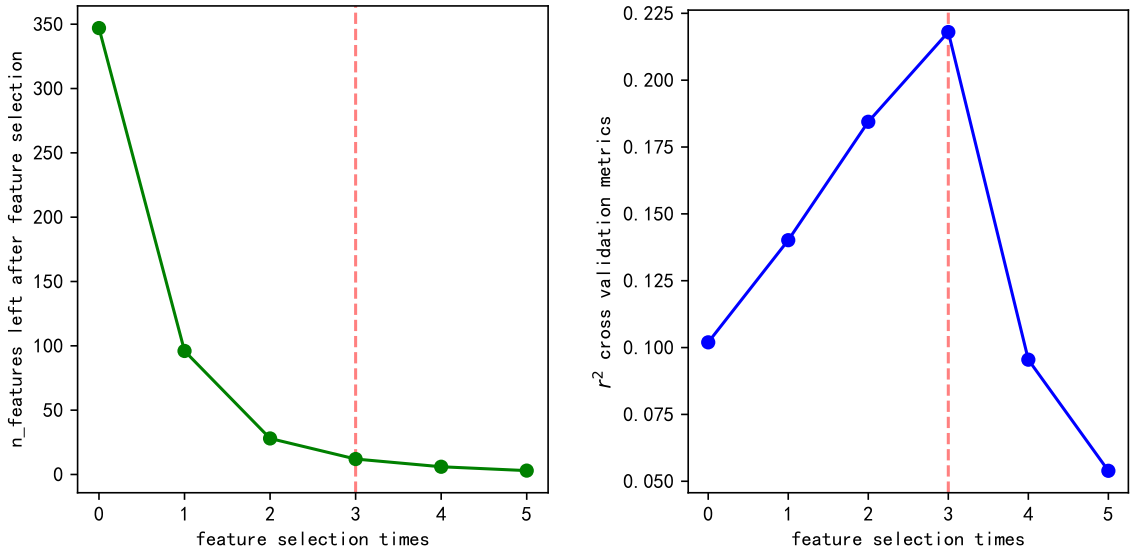
$$\hat{J}_j^2 = \frac{1}{M}\sum_{m=1}^{M}\hat{J}_j^2(T_m) \tag{4}$$

其中，M 是树的数量。特征 j 在单颗树中的重要度的如下：

$$\hat{J}_j^2 = \frac{1}{M}\sum_{t=1}^{L-1} \hat{i}_t^2 I(v_t = j) \tag{5}$$

其中，L 为树的叶子节点数量，L–1 即为树的非叶子节点数量（构建的树都是具有左右孩子的二叉树），$_t$ 是和节点 t 相关联的特征，$\hat{i}_t^2$ 是节点 t 分裂之后平方损失的减少值。

考虑到 GBDT 模型的准确性与鲁棒性优势，选用 GBDT 模型作为嵌入式特征筛选法的基模型。GBDT 采用的是 `sklearn.ensemble.GradientBoostingRegressor`，嵌入式特征筛选采用的是 `sklearn.feature_selection.SelectFromModel`。



(a) 特征筛选 n 次后剩余特征数　　**(b) 特征筛选次数与 r^2 评价指标的关系**

图 4　特征筛选 n 次后的剩余特征数与 r^2 评价指标

考虑到一次特征筛选可能并不能得到最好的筛选效果（无关变量还混杂在剩余的变量中），于是在本次建模过程中队员们通过多次试验实验探索了特征筛选次数与剩余特征数、下游模型 r^2 的关系。下游模型与章节4.3一致，为 LightGBM 模型。我们将特征筛选器与回归器组合成一个 pipeline：`Pipeline(SelectFromModel, LGBMRegressor)`，作为一个整体来验证特征筛选的效果。

该实验分别进行了 0 次特征筛选到 5 次特征筛选（0 次特征筛选表示不进行特征筛选），得到的实验结果如图 4 和表3所示。

实验反映了剩余特征数的下降程度随着特征筛选次数的增加逐渐减小，而模型整体表现在特征筛选数为 3 时到达峰值，之后迅速下降，说明 3 次特征筛选后保留的均为重要特征，如果再删除就会极大地影响模型表现。

表 3　特征筛选 n 次后的各指标

feature selection times	r^2 metrics	n_features left
0	0.101955	347
1	0.140157	96
2	0.184434	28
3	0.217998	12
4	0.095470	6
5	0.053913	3

通过实验，确定了做 3 次特征筛选的方案。根据这一方案，筛选出了 12 个主要变量，变量的名称与特征重要度见图 5 与表4。

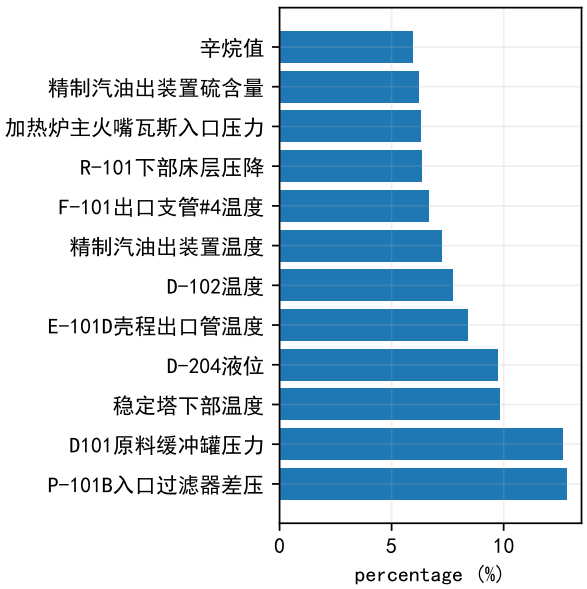


图 5　主要变量的特征重要度

表 4　主要变量的特征重要度

rank	feature importances	percentage (%)	name
0	0.0785	12.83	P-101B 入口过滤器差压
1	0.0774	12.66	D101 原料缓冲罐压力
2	0.0602	9.84	稳定塔下部温度
3	0.0596	9.74	D-204 液位
4	0.0515	8.42	E-101D 壳程出口管温度
5	0.0474	7.75	D-102 温度
6	0.0443	7.25	精制汽油出装置温度
7	0.0408	6.67	F-101 出口支管 #4 温度
8	0.0390	6.37	R-101 下部床层压降
9	0.0385	6.29	加热炉主火嘴瓦斯入口压力
10	0.0379	6.20	精制汽油出装置硫含量
11	0.0364	5.95	辛烷值

4.2.3 主要变量数据分析

对于筛选得到的 12 个主要变量，依据特征重要度取前 4 个变量加上预测目标 RON 损失一起做一个相关性分析，得到了图 6 所示的相关性矩阵图。根据 pearson 相关系数公式(6)计算了 12 个主要变量加上 RON 损失两两之间的相关系数，并绘制了图 7 所示的特征相关系数热力图。

$$\rho_{X,Y} = \frac{X,Y}{\sigma_X \sigma_Y} \tag{6}$$

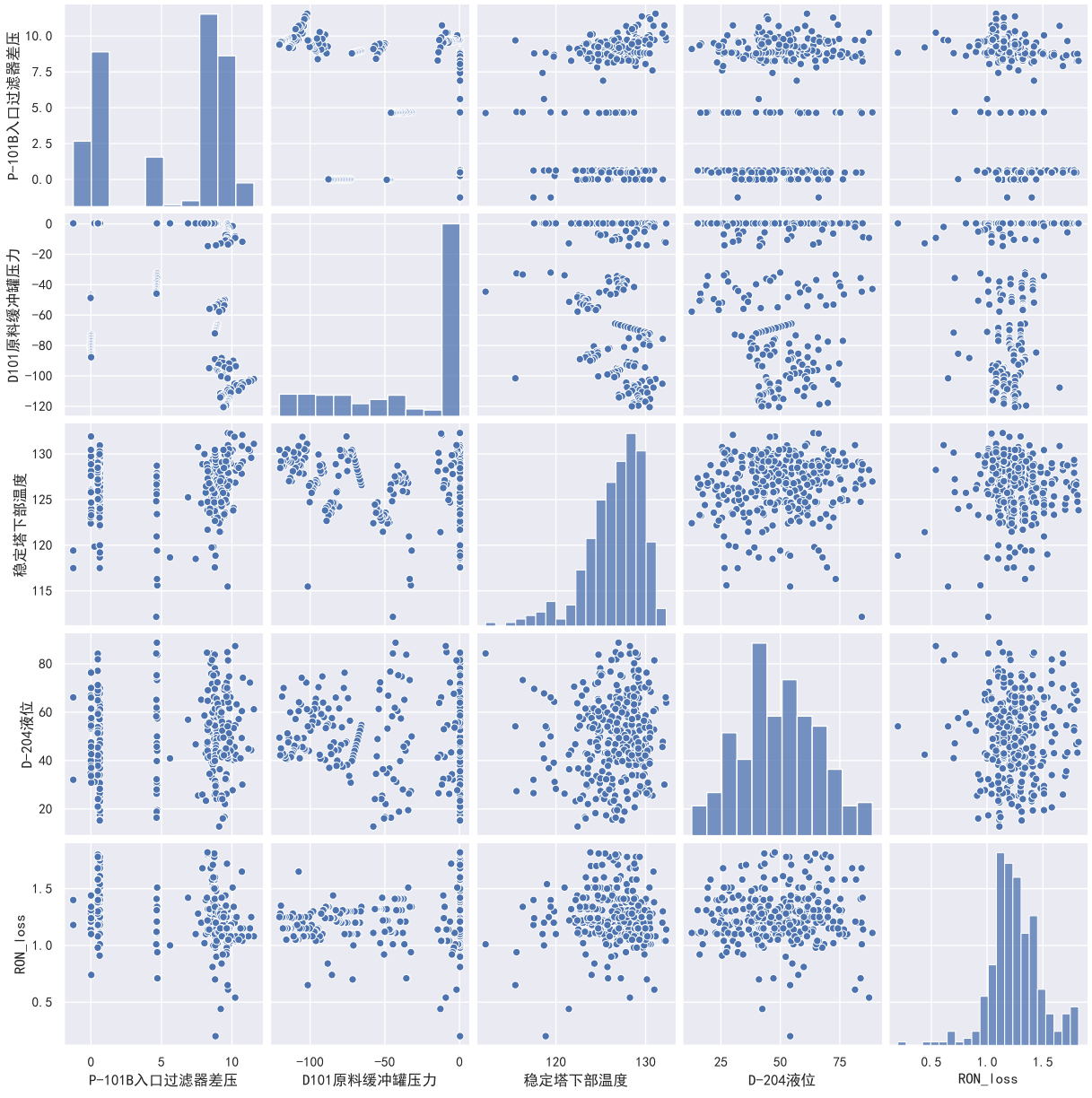

P-101B入口过滤器差压
D101原料缓冲罐压力
稳定塔下部温度
D-204液位
RON_loss
10.0
7.5
5.0
2.5
0.0
0
−20
−40
−60
−80
−100
−120
130
125
120
115
80
60
40
20
1.5
1.0
0.5
0
5
10
−100
−50
0
120
130
25
50
75
0.5
1.0
1.5

图 6　相关性矩阵图

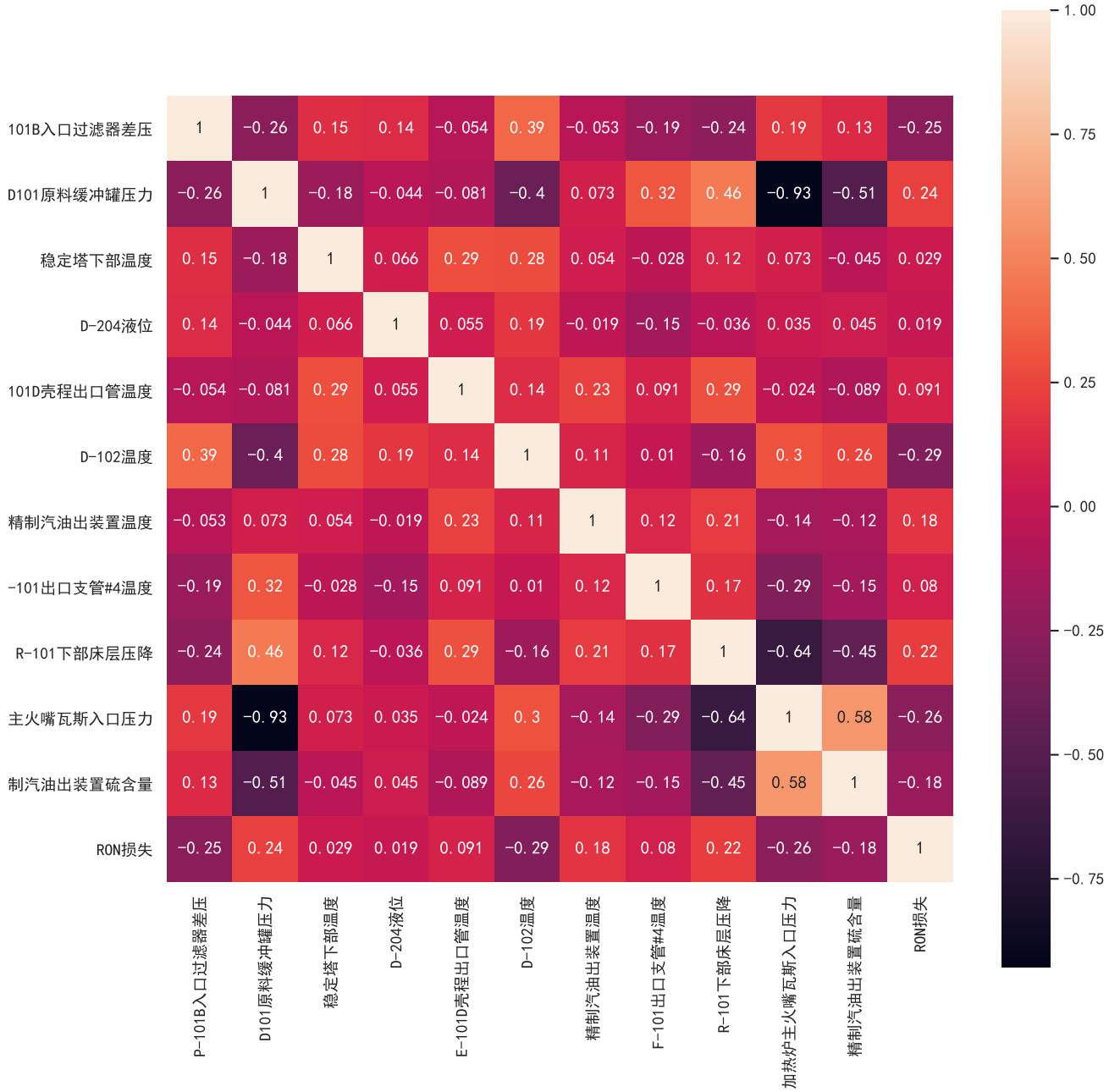


图 7　特征相关系数热力图

4.3 问题三：建立辛烷值（RON）损失预测模型

4.3.1 问题分析

本问题需要基于上述问题 2 找到的主要变量，通过数据挖掘技术建立辛烷值损失预测模型，并进行模型验证。

预测问题的本质是构造或学习获得问题输入与输出间的映射函数 $f:\mathbb{R}^n \to \mathbb{R}$，常见解决思路有回归分析、灰色预测模型、神经网络等多种方法。而辛烷值损失预测问题由于操作变量复杂、采样误差较大等多种因素的影响，使得上述映射函数变为十分复杂的函数，实际上难以用化工理论和经验模型等传统方法或者简单的线性模型准确拟合。综上考虑，需要选用能处理特征与特征，特征与类别之间高度非线性和相互强耦联的关系的机器学习模型。通过对各种机器学习模型的反复试验，选择了 LightGBM 作为回归器。

实验设计

本实验的实验数据采用章节4.1处理得到的 325×347 的数据。实验目标是结合章节4.2.2得到的特征筛选器，设计一个机器学习管线（详见**模型设计**），并通过 5 折交叉验证（5-folds Cross Validation, 5CV）的方式，在 r^2 评价指标下表现最好。

$$r^2 = 1 - \frac{\sum_i^N (\hat{y^{(i)}} - y^{(i)})^2}{\sum_i^N (\bar{y} - y^{(i)})^2} \tag{7}$$

r^2 的计算方法见公式(7)，其中 $\sum_i^N (\bar{y} - y^{(i)})^2$ 表示使用预测产生的错误，是 baseline，其中 $\sum_i^N (\hat{y^{(i)}} - y^{(i)})^2$ 表示使用待验证模型才生的错误。r^2 的取值范围为 $(-\inf, 1]$，应该越大越好。

模型设计

考虑到所选特征的取值范围差异较大，如“加热炉主火嘴瓦斯入口压力”的取值范围是 $[0.05, 0]$，“R-101 下部床层压降”的取值范围是 $[0, 120]$，容易造成不同尺度特征间难以比较的问题。缩放可以有效地解决上述问题，让变量之间可比性增强。除了特征之间尺度不同，特征的异常值与非正常分布也会对模型的鲁棒性与泛化能力造成影响。

QuantileTransformer 可以把每个特征转换到 0-1 均匀分布或者 $\mathbb{N}(0, 1)$ 的正态分布中。它执行的转换可以平滑掉非正常的分布，所以受到奇异点的影响比传统的 StandardScaler 要少。

根据章节4.2.2得到的特征筛选器，与上述的 QuantileTransformer，组建了如图8的机器学习管线。

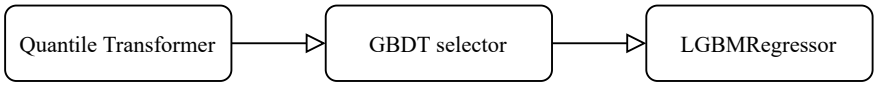


图 8　机器学习管线图

该机器学习管线的处理流程为：先用 QuantileTransformer 将数据的分布进行转换，转换为者 $\mathbb{N}(0,1)$ 的正态分布。然后用 GBDT selector 筛选出章节4.2.2得到的 12 个关键特征，最后用 LGBMRegressor 对筛选后的数据进行训练。

4.3.2 模型训练与验证

通过上文初步建立的模型，对实验数据进行训练与验证得到的 r^2=0.2179，其效果不能满足预期。经过分析，认为实验数据含有部分异常样本，需要将这些异常样本从实验数据中甄别并剔除。

检测异常样本的方法是用实验数据对机器学习管线进行训练，并在原数据上进行预测，然后用的公式(8)的方法计算误差，如果误差大于阈值 E_t，则认为是异常样本。

$$E = |y - \hat{y}| \tag{8}$$

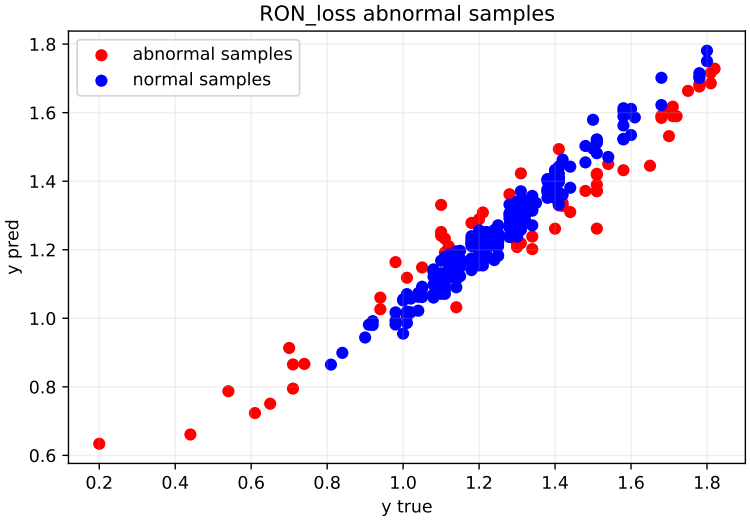


图 9　删除异常样本

经过多次实验，确定阈值 E_t=0.08，在该阈值下删除了 60 个样本。实验结果如图9所示。

删除异常样本后，在 5CV 下 r^2=0.3727，说明原实验数据受异常样本影响较大，删除异常样本后模型能达到不错的效果。

最后用 5CV 方法划分了 5 个不同的训练集与验证集，进行了综合地评估，在 5 次验证中验证集的 r^2 评价指标分别为 0.477, 0.409, 0.289, 0.362, 0.324，均值为 0.3727，模型表现稳定，具有鲁棒性与泛化能力。在训练集上 r^2 评价指标为 0.911，pearson 相关系数为 0.9648，见图10。

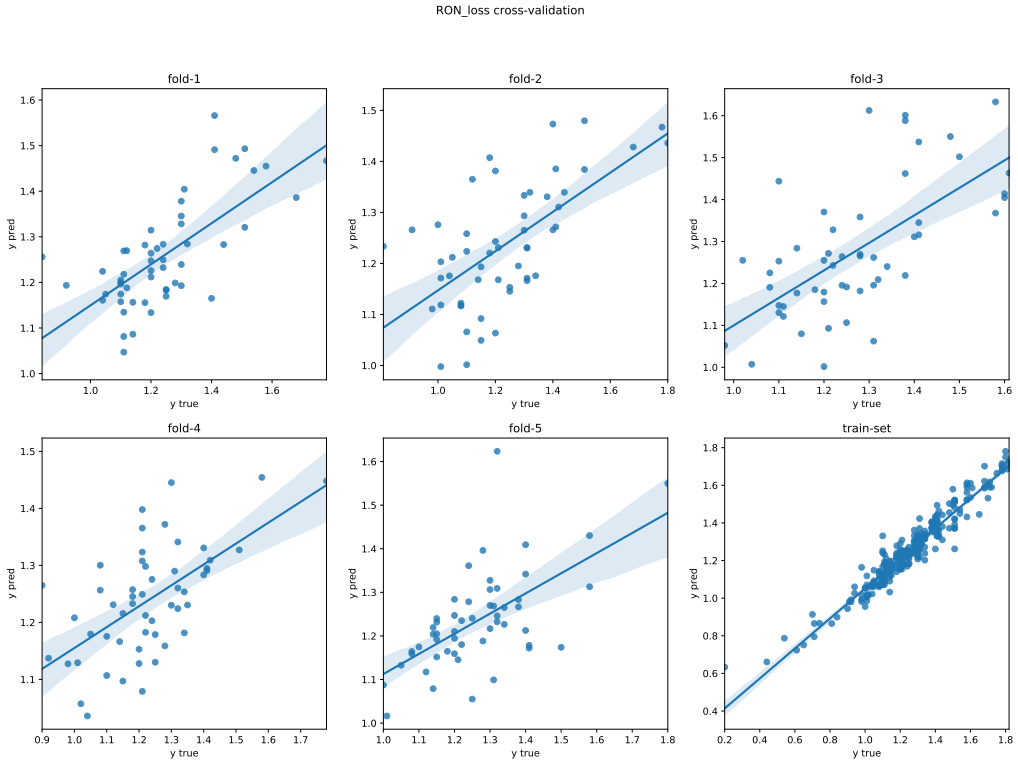


图 10　对 RON 损失进行交叉验证

4.4 问题四：主要变量操作方案的优化

4.4.1 问题分析

问题四的目标是在保证产品硫含量不大于 5μ/g 的前提下，将章节4.3得到的 RON 损失模型作为评价函数，对章节4.2.2得到的 12 个主要变量（其中 11 个变量是操作条件，原料的 RON 值是常数）进行优化，尽量使 RON 损失降幅超过 30%。

考虑到有硫含量的约束，所以先对操作变量与硫含量的关系进行建模。建模使用的回归器与机器学习管线与章节4.3中的相一致，其中特征筛选得到了 9 个关键变量：’精制汽油出装置硫含量’，’混氢点氢气流量’，’还原器温度’，’D203 出口燃料气流量’，’D-123 压力’，’烟气出对流室温度’，’冷氮气过滤器 ME-114 差压’，’R-102 底喷头压差’。其中产品硫含量为常数，其余 8 个为可调整的操作变量。

分析**与 RON 损失相关的 11 个操作变量**和**与硫相关的 8 个操作变量**，两者交集为**“精炼汽油出装置硫含量”**。分析原工业数据的产品硫含量，其中有 59 个样本的硫含量超过了 5μ/g 。故应该先对硫含量进行优化，将 325 个样本的产品硫含量都优化到 5μ/g 一下；然后对产品的 RON 损失值进行优化。因为操作变量“精炼汽油出装置硫含量”与 RON 损失相关，所以在对 RON 损失优化的过程中需要增加对产品硫含量的约束，使其满足题设条件；最后，如果有样本优化后仍不满足硫含量的约束，视其为异常样本，将其删除。算法

流程图见图11所示。

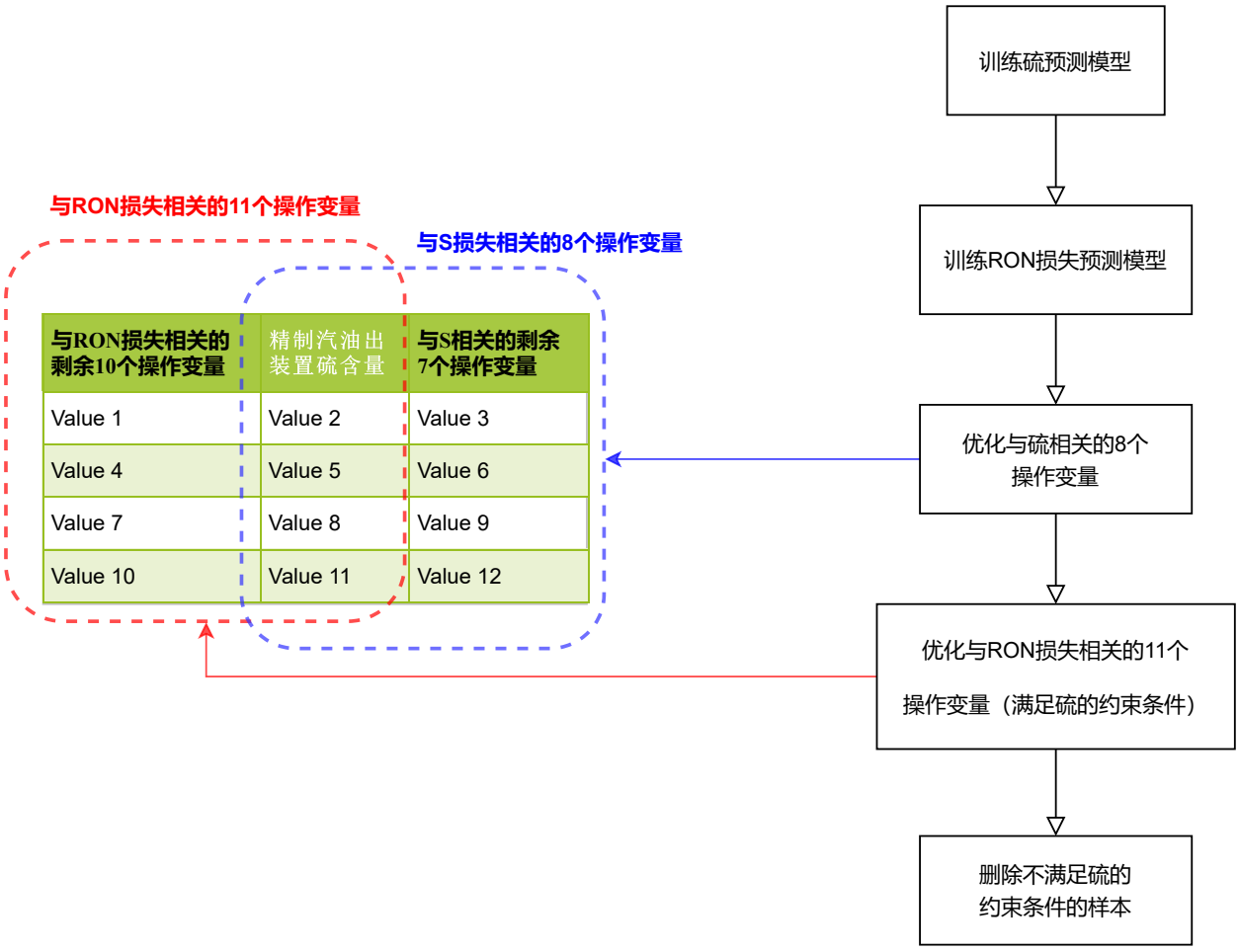


图 11　对 RON 损失进行交叉验证

4.4.2 贝叶斯优化与 TPE 算法简介

假设有一个函数 $f : x \to \mathbb{R}$ 需要在 $x \in X$ 内找到

$$x^* = \arg\min_{x \in X} f(x) \tag{9}$$

当 f 是凸函数且定义域 X 也是凸的时候，可以用已经广泛研究的凸优化来处理，但 f 不一定是凸的。在这种情况下就可以用启发式算法来做全局优化，常见的算法有进化算法、粒子群算法等。但贝叶斯算法具有鲁棒性高、搜索效率高等特点与优点，更适合本题的场景，故在此选择贝叶斯优化来对 RON 损失和硫含量进行优化。

Sequential model-based optimization (SMBO)[**?**] 是贝叶斯优化的最简形式，其算法思路如图12所示。

- f: 评价函数，在本题中是章节4.3得到的预测模型
- X: 搜索空间，在本题中是与 RON 损失或硫相关的主要操作变量。

Algorithm 1 Sequential Model-Based Optimization

Input: $f, \mathcal{X}, S, \mathcal{M}$
$\mathcal{D} \leftarrow \text{INITSAMPLES}(f, \mathcal{X})$
for $i \leftarrow |\mathcal{D}|$ **to** T **do**
  $p(y \mid \mathbf{x}, \mathcal{D}) \leftarrow \text{FITMODEL}(\mathcal{M}, \mathcal{D})$
  $\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x},\ p(y \mid \mathbf{x}, \mathcal{D}))$
  $y_i \leftarrow f(\mathbf{x}_i)$ ▷ Expensive step
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$
end for

图 12　SMBO 算法

- D: 表示数据组成的数据集，在本题中是原工业数据。
- S: 是 **Acquisition Function**（采集函数），用来评价采样样本的好坏，一般采用 EI 函数，见公式(10)。
- M: 是对数据集 D 进行拟合得到的模型，如随机森林、高斯过程等。本题使用 TPE[**?**] 模型。

$$EI(X) = \mathbb{E}_{max}(f(x) - f(x^+), 0) \tag{10}$$

TPE 全称 Tree-structured Parzen Estimator，是用 GMM（Gaussian Mixture Model）来学习超参模型的一种方法。

首先根据 Bayes 定义，$p(x|y)$ 即模型 $loss$ 为 y 的时候超参为 x 的条件概率。第一步，根据已有的数据选取一个 $loss$ 的阈值 y^*，一般为 15% 的分位数。对大于阈值和小于阈值的数据，分别学习两个概率密度 $l(x)$ 和 $g(x)$ 。

$$p(x|y) = \begin{cases} l(x) & if\ y < y^* \\ g(x) & if\ y \geq y^* \end{cases} \tag{11}$$

根据 EI 公式(10)，将其带入公式(11)，发现只要能 $\arg\min_x g(x)/l(x)$ 就能 $\arg\min_x EI(x)$。

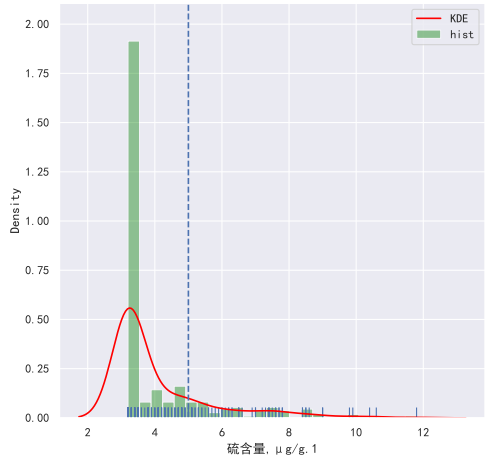
$$EI_{y^*}(x) \propto (\gamma + \frac{g(x)}{l(x)}(1-\gamma))^{-1} \tag{12}$$

综上，TPE 的原理就是建立两个分布：优势样本的分布于劣势样本的分布。TPE 在每次迭代中按照这样的规则去采样：采样的样本在优势样本分布 $l(x)$ 中的概率密度尽量大，在劣势样本分布 $g(x)$ 中的概率密度尽量小，本质上是逐渐增大在优势样本附近采样率的一个启发式优化过程。

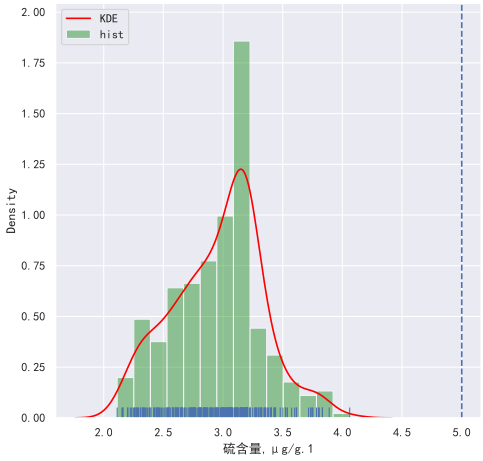
4.4.3 对产品硫含量进行优化

原工业数据的产品硫含量的分布图见图13a，其中大于 $5\mu g/g$ 的样本有 59 个，占 18.15%。

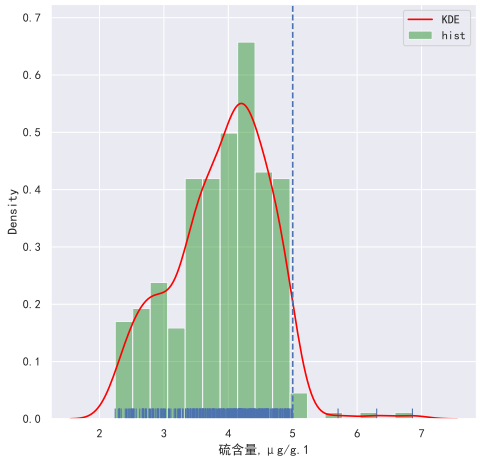
可以看到原工业数据的产品硫含量为长尾分布，充斥着大量硫含量偏高的样本。



(a) 原工业数据硫含量的分布图



(b) 单独对与硫优化后的分布图



(c) 对 RON 损失优化后硫的分布图

图 13　313 号样本处理 2 中的 3 个变量

以与硫含量相关的 8 个主要变量（见章节4.4.1）作为待优化变量，并根据操作变量的取值范围得到一个参数空间 $\mathbb{H}_S \in \mathbb{R}^8$。用之前得到的硫预测模型作为优化过程的评价函数 $g(s), s \in \mathbb{H}_S$，用贝叶斯优化库 `HyperOpt` 的 TPE 算法对工业数据的 325 个样本逐一进行优化，每次优化迭代 50 次。

单独对参数空间 $\mathbb{H}_S$ 优化后，产品硫含量的分布图见图13b。可以看到产品的硫分布被压缩到了区间 $[0,5)$ 中了。

4.4.4 对 RON 损失进行优化

以与 RON 损失含量相关的 11 个主要变量（见表4）作为待优化变量，并根据操作变量的取值范围得到一个参数空间 $\mathbb{H}_R \in \mathbb{R}^{11}$。用之前得到的 RON 损失预测模型作为评价函数 $f(r), r \in \mathbb{H}_R$。

但实际使用时并不能直接用 $f(r)$ 作为优化过程的评价函数。如流程图11所示，硫与 RON 损失有一个操作变量重合，虽然在上一步中通过调整与硫相关的 8 个操作变量是的目前产品硫含量大幅下降，但是在优化 RON 损失的过程中可能在调整这一重合的操作变量时会导致产品硫含量超出题设范围。故对 RON 损失的评价函数做出调整，如公式(13)所示，用调整后的 $\hat{f}(r)$ 代替 $f(r)$：

$$\hat{f} = f(r) + \infty \times \max(0, g(s) - 5) \quad r \in \mathbb{H}_R \;\; s \in \mathbb{H}_S \tag{13}$$

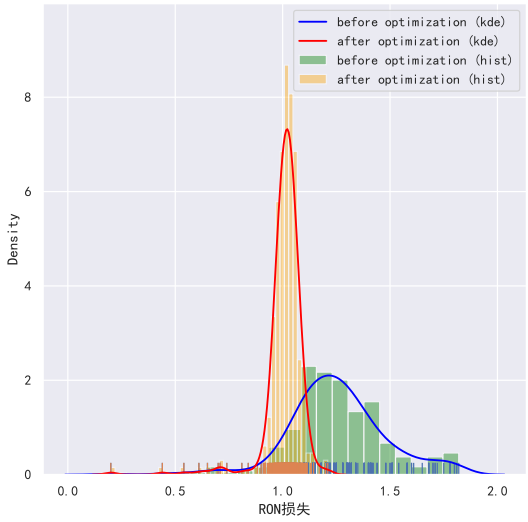


图 14　优化前后的 RON 损失分布图

更新评价函数后，如之前的操作，对工业数据的 325 个样本逐一进行优化，每次优化迭代 50 次。优化前后 RON 损失的分布图如图14所示，可见 RON 损失在优化后被压缩到了 $[0.8, 1.2]$ 附近的区域。

优化后产品硫含量分布如图13c所示，可见分布在 $S = 5$ 处呈截断的痕迹，但仍有 3 个样本超出了 $S \leq 5$ 的限制，这 3 个样本在表5中列出。

优化后，有 55 个样本的 RON 损失降幅大于 30%，占比为 16.92%。RON 损失降幅情

表 5　3 个优化后不符合约束条件的异常样本

time	原硫含量 ($\mu g/g$)	优化后硫含 ($\mu g/g$)	原 RON 损失	优化后 RON 损失
2017/11/10 8:00:00	4.2	6.30	1.10	0.99
2017/11/6 8:00:00	6.2	5.70	1.10	0.91
2017/9/18 8:00:00	3.8	6.86	1.21	0.89

况见图15。

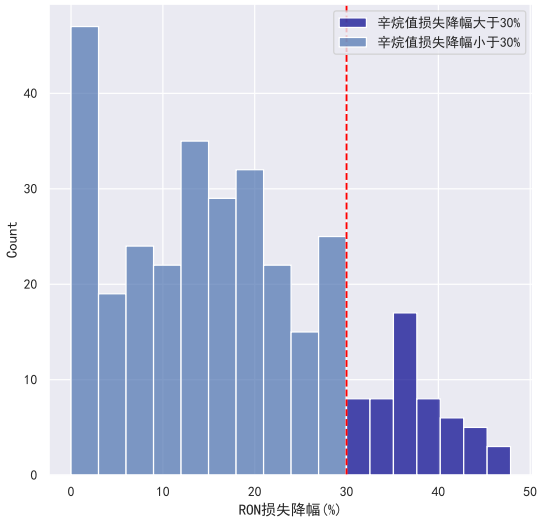


图 15　RON 损失降幅直方图

为了更好地对比优化 RON 损失前后主要操作变量的变化情况，绘制了 11 个主要操作变量和产品辛烷值（常量）的分布图，如图16和图17所示。

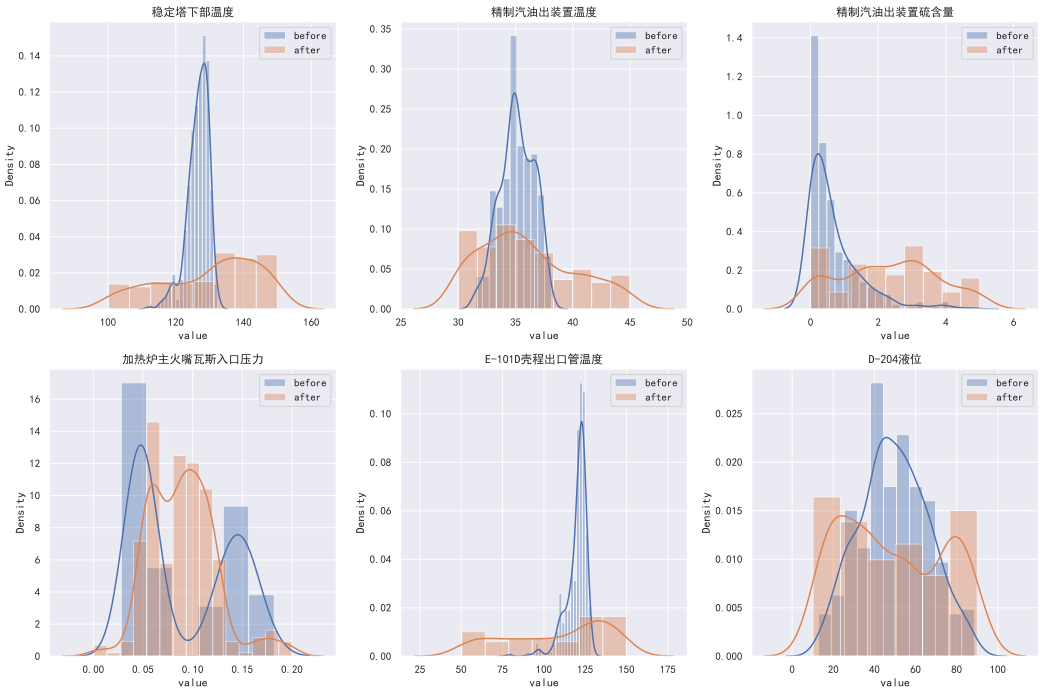


图 16　优化前后 6 个操作变量的分布

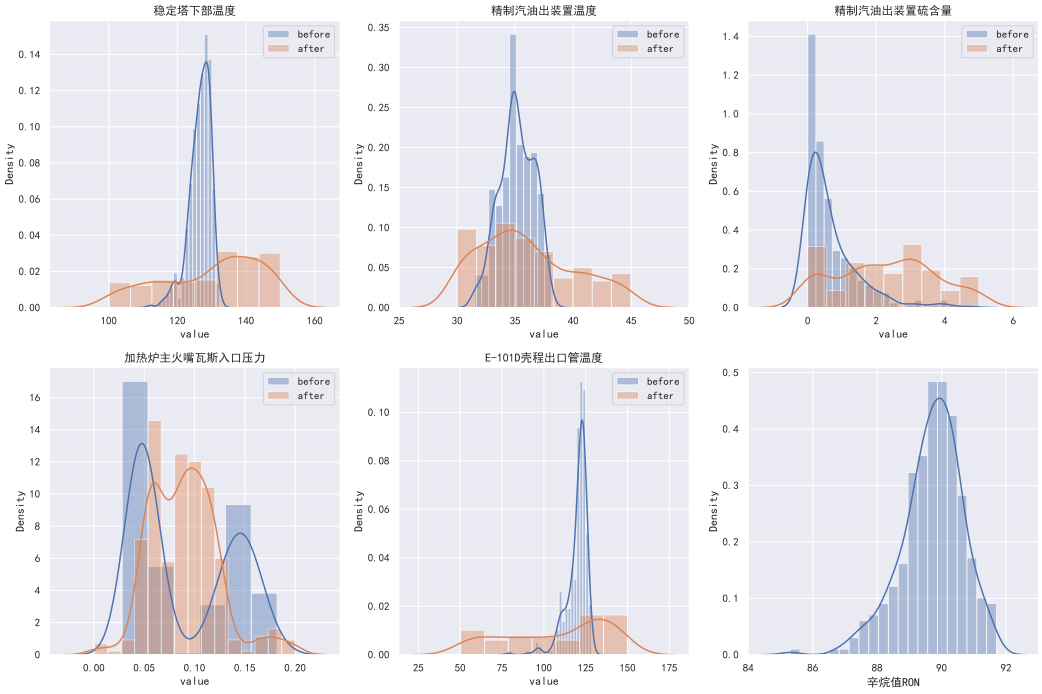


图 17　RON 与优化前后的 5 个操作变量的分布

4.5 问题五：模型的可视化展示

4.5.1 问题分析

本文已经在问题三的解答中选出了 12 个主要变量，在问题四中对包括 133 样本在内的许多样本进行了优化，其中 133 样本可以通过改变一些操作变量实现在硫含量不高于 5（5μg/g）的情况下辛烷值损失降低至 0.8828。而在这一问我们需要找出由原先的操作条件，逐步调整为优化后的操作条件的调整路径，使得每一次调整幅度不超过操作变量每次允许调整幅度值 Δ，同时，每一步操作变量对应的状态产生的硫不应该高于 5μg/g。而且，操作变量调整过程中不应超出变量的操作范围（操作变量的范围由经验值以及附件中给出的操作变量信息共同划定）。

4.5.2 问题建模

考虑到工程实际情况，在此使用改进的启发式搜素 A*[**?**] 算法求解该问题。本问题的处理流程图及 A* 算法的流程图如下：

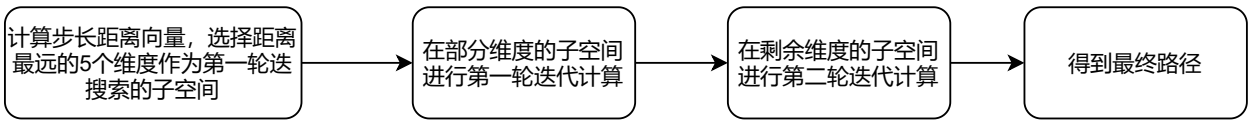


图 18　模型的可视化展示总体流程图

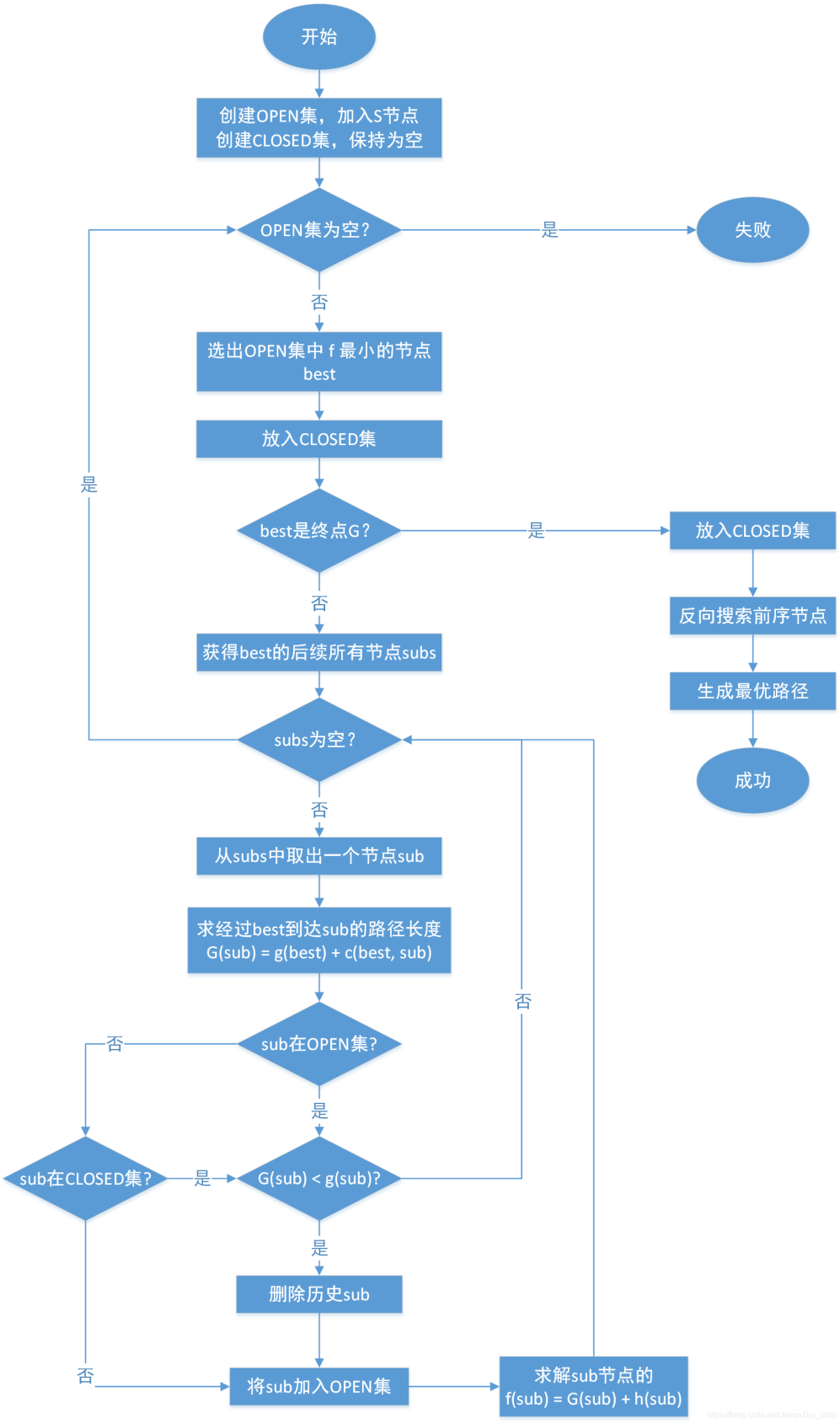

开始
创建OPEN集，加入S节点
创建CLOSED集，保持为空
OPEN集为空？
是
失败
否
选出OPEN集中 f 最小的节点
best
放入CLOSED集
是
best是终点G？
是
放入CLOSED集
否
反向搜索前序节点
获得best的后续所有节点subs
生成最优路径
subs为空？
成功
否
从subs中取出一个节点sub
求经过best到达sub的路径长度
G(sub) = g(best) + c(best, sub)
否
sub在OPEN集？
否
是
sub在CLOSED集？
是
G(sub) < g(sub)?
是
删除历史sub
否
将sub加入OPEN集
求解sub节点的
f(sub) = G(sub) + h(sub)

图 19　A* 算法的流程图

对该问题建模如下：首先 12 个主要变量中，只有 11 个是操作变量，这 11 维操作变量的数值可以看成 11 维空间中的一个位置的坐标。记原始的坐标为 x，第四问优化后的变量为 y，所以实际就是求解在 11 维空间中由 x 到 y 的一条路径，其中路径的途径点不能超过所有变量的取值范围组成的 11 维立方体，且不能经过 11 维立方体中使硫产量大于 5 的区域，路径应尽量短，而且每一步在每一个维度上步幅都不应超过对应操作变量允许调整的幅度值 Δ。为简化问题以及减少计算量，假设每一个操作变量的每一次调整都刚好等于 Δ（最后一步除外，最后一次是小于等于 Δ）。第三问得到的预测硫和辛烷值损失的模型记为硫产量 $S = S(x)$，辛烷值损失 $RON\_LOSS = N(x)$。

4.5.3 模型优化

在建模过程中发现，在 11 维空间中运行 A* 算法需要非常巨大的计算资源，于是进一步优化计算模型，得到的计算公式如下：

$$\vec{d} = |(\vec{y} - \vec{x})| \tag{14}$$

$$\overrightarrow{step} = \vec{d}./\overrightarrow{\Delta} \tag{15}$$

式（15）中./代表向量运算中的点除。用起始点 x 与终点 y 坐标的距离向量，元素除以对应的 Δ，然后向上取整，得到一个 11 维向量，每个维度的值代表每个维度上起始点 x 与终点 y 的步长距离，经过计算得到 11 维向量 $[4, 1, 6, 2, 3, 2, 59, 4, 3, 3, 6]$, 经过观察发现只有少数维度步长距离较远，大部分维度距离并不远，于是我们决定将 A* 算法修改为两轮计算，每一轮都只在部分维度组成的子空间中找通往目的地的通路（其他维度保持不变），只要每一轮都可以找到这样符合上文所述约束条件的通路，那么两个通路链接起来，就是完整的从起点到终点的一条道路。从中选取步长距离较远的 $index = [0, 2, 6, 7, 10]$ 维度进行第一轮优化，其他维度保持不变，当挑选的这些维度都优化到 y 点对应维度的值时，第一轮优化结束，此时的坐标为 x1=[121.3461, 34.4524, 4.9924, 0.0386, 120.7679, 52.7805, 321.5952, 32.7367, 8.8462,410.3512,-2.2292] 对应的 RON 损失, S 为 1.0422, 4.3748。接着，保持 $index = [0, 2, 6, 7, 10]$ 对应的坐标值不变，对剩余的 6 个维度进行第二轮优化，优化结束后最终坐标等于 [121.3461,34.2690,4.9924,0.1060,123.3863,59.7742,321.5952,32.7367,11.2289,412.8519,-2.2292]，即为第四问中求出的优化后操作变量值。x1=[121.3461, 34.4524, 4.9924, 0.0386, 120.7679, 52.7805, 321.5952, 32.7367, 8.8462,410.3512,-2.2292] 对应的 RON 损失, S 为 1.0422, 4.3748。接着，保持 $index = [0, 2, 6, 7, 10]$ 对应的坐标值不变，对剩余的 6 个维度进行第二轮优化，优化结束后最终坐标等于 [121.3461,34.2690,4.9924,0.1060,123.3863,59.7742,321.59, 32.7367,11.2289,412.8519,-2.2292]，即为第四问中求出的优化后操作变量值。将每一轮迭代中的硫产量, 辛烷值损失，及相关操作变量画成曲线如图所示：

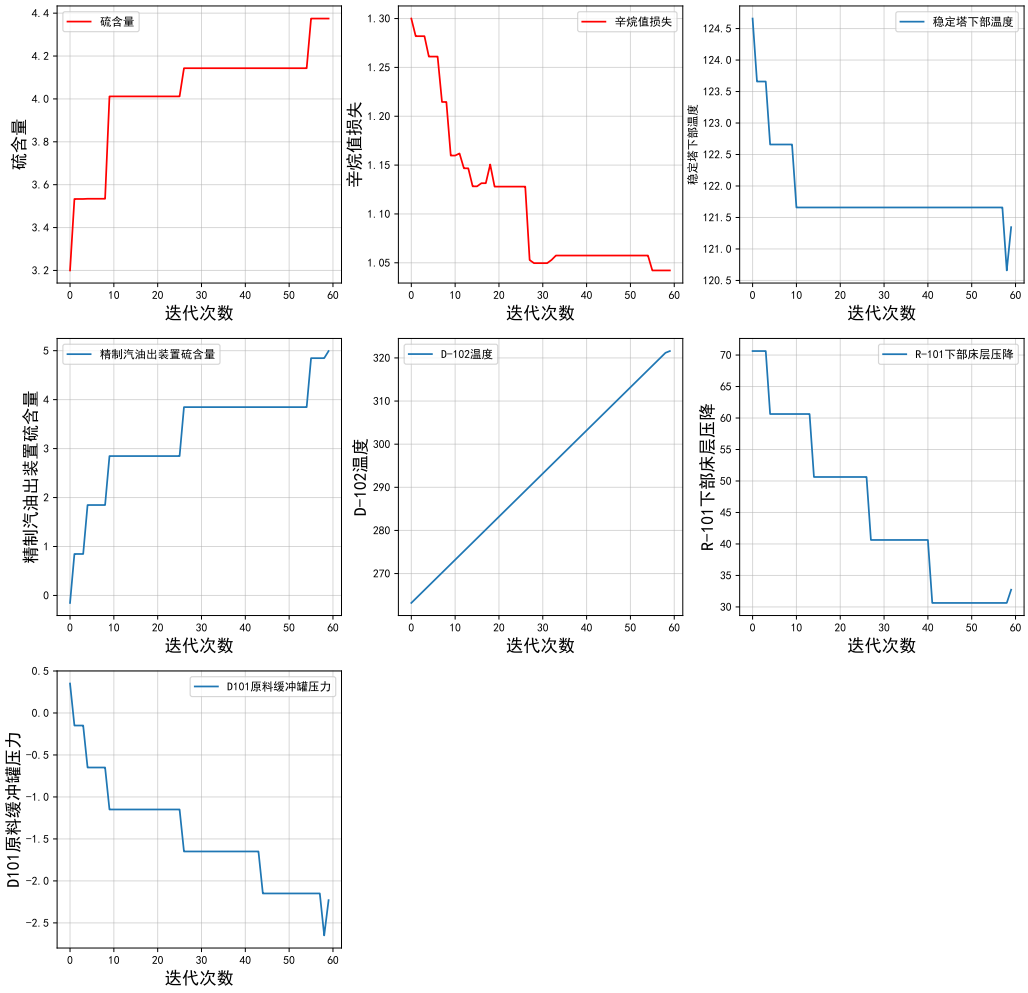


图 20　第一轮迭代中硫产量、辛烷值损失及相应的五个操作变量随迭代次数变化的曲线

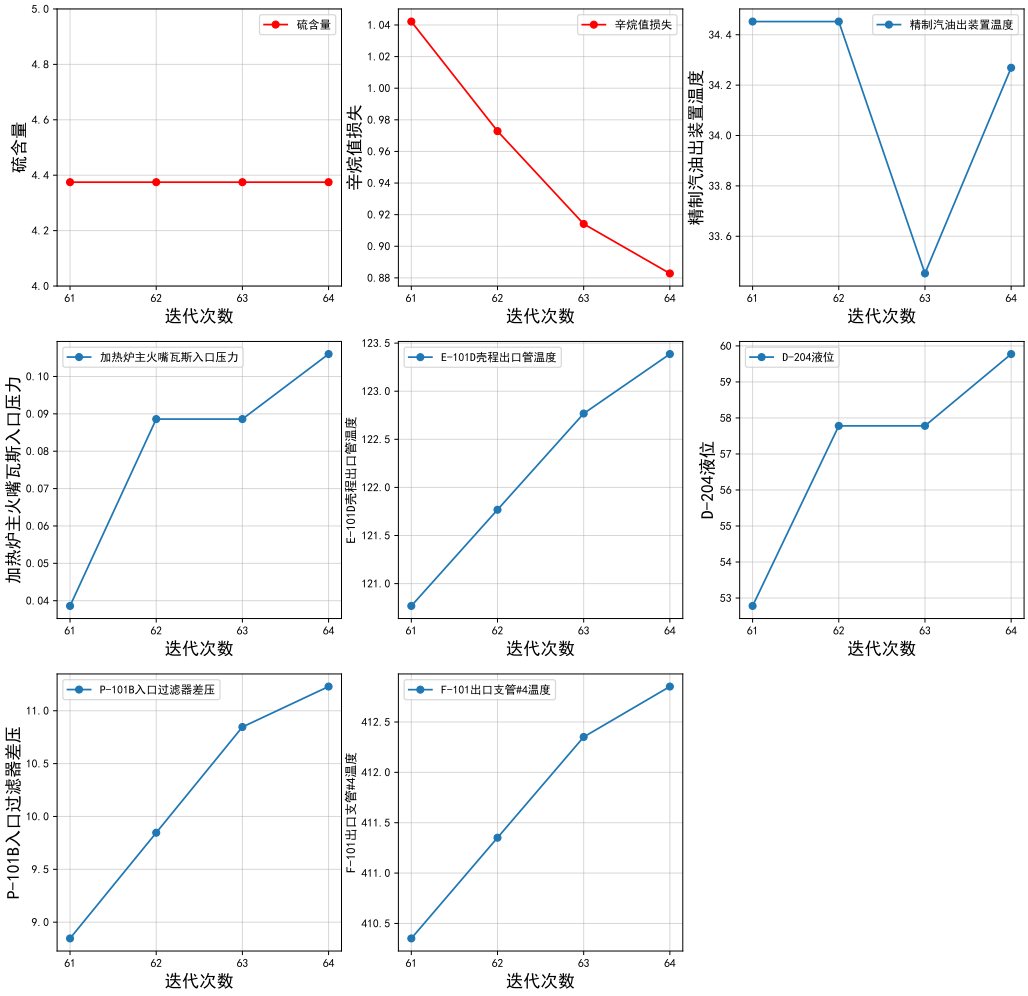


图 21　第二轮迭代中硫产量、辛烷值损失及相应的五个操作变量随迭代次数变化的曲线

5. 模型评价

5.1 模型的优点

5.1.1 问题一

1. 根据时序上是否平稳来决定 313 与 285 号样本的空值填充策略，灵活且具有自适应性。

2. 通过公式定义操作变量的超出程度并以此为阈值删除采样样本，用 3σ 准则删除异常的采样样本，为下游操作提供鲁棒的模型。

3. 根据不同场景制定对应的空值处理策略，灵活且高效。

5.1.2 问题二

1. 用嵌入式的特征打分模型梯度提升树对特征进行评价，计算高效，结果可靠。

2. 对比不同的特征筛选次数进一步限定主要变量的范围，科学且合理。

5.1.3 问题三

1. 基于分布转换与 LightGBM 模型在 12 个主要变量的基础上建立 RON 损失模型，鲁棒且可泛化。

2. 甄别出离群的异常样本并将其删除，提升了模型的综合表现。

5.1.4 问题四

1. 根据产品硫含量不大于 5μg/g 的约束条件重新设计了评价函数，评价函数科学合理。

2. 通过贝叶斯优化的 TPE 算法对 325 个样本的 RON 损失值进行优化，优化算法稳定高效。

5.1.5 问题五

1. 对问题进行了合理的建模，求解出完全满足题目要求的修改路径，保证每次调整不超过 Δ 允许的范围，每次调整不会超出操作变量的可行域，每次调整不会让硫产量大于 5μg/g，最终实现辛烷值损失降低 32.61%。

2. 对问题进行了合理简化，将连续变化的操作变量离散化处理，有利于使用启发式搜索算法，同时减少了计算量。

3. 模型创新性的使用两步迭代算法，进一步简化模型，极大的减少了运算量缩短了运算时间，同时保证结果的正确性。

5.2 模型的缺点

5.2.1 问题一

1. 数据的空值等异常情况错综复杂，当前的处理策略可能难以覆盖所有的情况。

5.2.2 问题二

1. 特征筛选的粒度还不够细，当前得到的 12 个主要变量可能不是最优的。

5.2.3 问题三

1. 没有做超参优化与集成学习，模型的表现可能不是最优的。

5.2.4 问题四

1. 贝叶斯优化可能不能在限定的优化次数中找到全局最优解。

5.2.5 问题五

1. 为了减少搜索空间每次改变操作变量都是按照固定的值 Δ 调整，而实际操作中可以按照连续值调整自由度更高。
2. 两步迭代的算法可能无法找到最短修改操作变量的路径。

附录 A 数据预处理关键部分 Python 源程序

```
import numpy as np
import parse

import pylab as plt

df = pd.read_excel("附件一：325个样本数据.xlsx", skiprows=[0, 1])
columns = ops.columns[eq_0>=H]
print(f"many 0 n_columns = {len(columns)}")
for column in columns:
  plt.subplot(1,2,1)
  (ops[column].hist())
  plt.subplot(1,2,2)
  plt.plot(ops[column].index, ops[column].tolist())
  plt.show()
  print("column =", column)
  print("n_zeros = ",np.count_nonzero(ops[column]==0))
  print("mean = ",np.mean(ops[column]))
  print("var = ",np.var(ops[column]))
  print("="*20)
  def plot_empty(cols, fname):
  N = len(cols)
  plt.rcParams['figure.figsize'] =(6 * N, 12)

for i, col in enumerate(cols):
# 获取名字
name = op_df.loc[col, "位号"]
print(name)
title = f'{name}'
cn = op_df.loc[col, "中文名称"]
print(cn)
if not pd.isna(cn):
  title += f"({cn})"
  # 直方图（分布）
  plt.subplot(N, 2, i * 2 + 1)
  plt.grid(alpha=0.5)
  plt.hist(ops[col])
  plt.xlabel("value")
```

```
  plt.ylabel("frequency")
  plt.title(f"{cn}-分布图")
  # 折线图
  plt.subplot(N, 2, i * 2 + 2)
  plt.title(f"{cn}-折线图")
  #    x_major_locator=MultipleLocator(50)
  #    ax.xaxis.set_major_locator(x_major_locator)
  plt.grid(alpha=0.5)
  plt.xlabel("time")
  plt.ylabel("value")

s=ops[col];
plt.plot(df.index, s.tolist())
plt.tight_layout()
plt.savefig(f"{fname}.pdf")
plt.close()
for i in range(354):
  logic = ~np.logical_and(d285.iloc[:,i+1]>=op_df.loc[i,"lower"] ,
     d285.iloc[:,i+1]<=op_df.loc[i,"upper"])
  n_invalids = (logic.sum())
  if n_invalids>0:
  print(i, n_invalids)
  x_major_locator=MultipleLocator(5)
  ax=plt.gca()
  #ax为两条坐标轴的实例
  ax.xaxis.set_major_locator(x_major_locator)
  plt.xticks(rotation=30)
  plt.rcParams['figure.figsize'] =(7, 6)
  plt.grid()
  plt.title(op_df.loc[108, "位号"])
  s=d313_["Unnamed: 109"]; plt.plot(d313.time, s.tolist()) # 2
     zeros [24, 25]
  plt.tight_layout()
  plt.savefig("313_fill_1.pdf")
```

附录 B LightGBM 建模与特征筛选 Python 源程序

```
import numpy as np
```

```
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectFromModel
from lightgbm import LGBMRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import KFold, cross_val_score
import pylab as plt
from scipy.stats import pearsonr
from sklearn.metrics import r2_score
from sklearn.preprocessing import QuantileTransformer
from joblib import dump
import warnings
import seaborn as sns

warnings.filterwarnings("ignore")
X = pd.read_csv("../processed.csv")
y_s = X.pop("S")
RON = X.pop("RON")
y = X.pop("RON_loss")


def func(X: pd.DataFrame, y: pd.Series, selection_times=3,
   title="RON_loss", del_abnormal=False,
  abnormal_threshold=0.08):
  y = np.array(y)
  selector = SelectFromModel(
  estimator=GradientBoostingRegressor(random_state=0))
  X_ = QuantileTransformer(n_quantiles=1000).fit_transform(X)
  X_ = pd.DataFrame(X_, columns=X.columns)
  for i in range(selection_times):
  X_d = selector.fit_transform(X_, y)
  X_ = pd.DataFrame(X_d, columns=X_.columns[selector.get_support()])
  X_d =
     QuantileTransformer(n_quantiles=1000).fit_transform(X[X_.columns])
  # X_d=X[X_.columns].values
  X_ = pd.DataFrame(X_d, columns=X_.columns)
  print(f"{title} | {selection_times}次特征筛选后的X_.shape =
     {X_.shape}")
  print(f"{title} | 特征筛选后保留的列: {X_.columns.tolist()}")
  cv = KFold(n_splits=5, shuffle=True, random_state=0)
```

```
pipeline = LGBMRegressor(random_state=0, n_estimators=100,
    learning_rate=0.1)
pipeline.fit(X_, y)
y_pred = pipeline.predict(X_)
train_score = r2_score(y, y_pred)
pearson_correlation = pearsonr(y, y_pred)[0]
print(f"{title} | 在训练集上, r2 = {train_score}, pearson 相关系数 =
    {pearson_correlation}")
y_ = y.copy()
y_pred_ = y_pred.copy()
if del_abnormal:
y_pred = pipeline.predict(X_)
err = np.abs(y - y_pred)
mask = err > abnormal_threshold
print(f"{title} | 异常样本数 = {np.count_nonzero(mask)}")
plt.rcParams['figure.figsize'] = (7, 4.5)
plt.grid(alpha=0.2)
plt.scatter(y[mask], y_pred[mask], label="abnormal samples",
    c="r")
plt.scatter(y[~mask], y_pred[~mask], label="normal samples",
    c="b")
plt.legend(loc="best")
print(f"{title} | 删除异常样本前的表现 = {cross_val_score(pipeline,
    X_, y, cv=cv).mean()}")
X_ = X_.loc[~mask, :]
y = y[~mask]
print(f"{title} | 删除异常样本后的表现 = {cross_val_score(pipeline,
    X_, y, cv=cv).mean()}")
plt.title(f"{title} abnormal samples")
plt.xlabel("y true")
plt.ylabel("y pred")
plt.savefig(f"{title}_abnormal.pdf")
plt.close()
valid_scores = []
plt.rcParams['figure.figsize'] = (18, 12)
for i, (train_ix, valid_ix) in enumerate(cv.split(X_, y)):
  X_train = X_.iloc[train_ix, :].copy()
  X_valid = X_.iloc[valid_ix, :].copy()
  y_train = y[train_ix].copy()
```

```
    y_valid = y[valid_ix].copy()
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_valid)
    plt.subplot(2, 3, i + 1)
    sns.regplot(x="y true", y="y pred",
    data=pd.DataFrame({"y true": y_valid, "y pred": y_pred}))
    plt.title(f"fold-{i + 1}")
    valid_scores.append(r2_score(y_valid, y_pred))
    plt.subplot(2, 3, 6)
    sns.regplot(x="y true", y="y pred",
    data=pd.DataFrame({"y true": y_, "y pred": y_pred_}))
    plt.title(f"train-set")
    plt.suptitle(f"{title} cross-validation")
    print(f"{title} | 5折交叉验证后，在验证集上的平均r2 =
       {np.mean(valid_scores)}\n"
    f"{title} | 每折的r2 = {valid_scores}")
    plt.savefig(f"{title}_cross-validation.pdf")
    plt.close()
    X_["label"] = y
    X_.to_csv(f"{title}_data.csv", index=False)
    dump(pipeline, f"{title}_model.bz2")

func(X, y, selection_times=3, title="RON_loss", del_abnormal=True,
   abnormal_threshold=0.08)
func(X, y_s, selection_times=2, title="S", del_abnormal=False)
```

附录 C 硫与 RON 损失优化 Python 源程序

```
  space = []
  label = RON_loss
  X_cur = X_RON.drop("label", axis=1)
  for column in X_cur.columns:
  if column == "label":
    continue
    if not column.startswith("op"):
      space.append(X.loc[R, column])
    else:
      space.append(hp.uniform(column, op_df.loc[column, "lower"],
```

```
            op_df.loc[column, "upper"]))
        #       init_point.append(X.loc[R, column])

    def objective(arg):
        op21 = arg[3]
        s = X.loc[R, X_S.drop("label", axis=1).columns]
        #       print(s)
        s["op21"]=op21
        #       print(s)
        arr = []
        arr = np.array(arg).reshape(1, -1)
        RON_pred = RON_pipeline.predict(arr).flatten()[0]
        S_pred = S_pipeline.predict(s.values.reshape(1, -1)).flatten()[0]
        if S_pred >= 5:
            return 100000
        return RON_pred

    print(f"before optimization, loss = {label[R]}")
    best = fmin(objective, space, algo=tpe.suggest,
        max_evals=max_evals) # points_to_evaluate=[init_point]
    point = space_eval(space, best)
    loss_opted = objective(space_eval(space, best))

    #    before_losses.append(RON_loss[R])
    if loss_opted<label[R]:
        after_losses.append(loss_opted)
    else:
        after_losses.append(label[R])
    print(f"after optimization, loss = {after_losses[-1]}")
    X.loc[R, X_cur.columns] = point
    print("=" * 50)

def optimize_S(R):
    space = []
    X_cur = X_S.drop("label", axis=1)
    transformer =
        QuantileTransformer(n_quantiles=1000).fit(X[X_cur.columns])
    for column in X_cur.columns:
    if column == "label":
```

```
    continue
  if not column.startswith("op"):
    space.append(X.loc[R, column])
  else:
    space.append(hp.uniform(column, op_df.loc[column, "lower"],
       op_df.loc[column, "upper"]))
  pipeline = Pipeline([
  ("transformer", transformer),
  ("lgbm", S_model),
  ])
  def objective(arg):
  arr = []
  arr = np.array(arg).reshape(1, -1)
  return pipeline.predict(arr).flatten()[0]

  print(f"before optimization, loss = {S[R]}")
  best = fmin(objective, space, algo=tpe.suggest, max_evals=50)
  point = space_eval(space, best)
  loss_opted = objective(space_eval(space, best))
  print(f"after optimization, loss = {loss_opted}")
  if loss_opted >= 5:
    print(f"[WARNING] sample {R} is bad!!!")
  bad_samples.append(R)
  else:
    X.loc[R, X_cur.columns] = point
  if loss_opted<S[R]:
    S_after_losses.append(loss_opted)
  else:
    S_after_losses.append(S[R])
```