


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	湖南科技大学
参赛队号	20105340008
队员姓名	**1.** 肖勇
	2. 侯俊杰
	3. 肖浩

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：

汽油精制研究更趋数据化和智能化，这对国家发展，科技创新，环境保护，企业效益等诸多方面有着巨大的影响，如何降低汽油精制过程中的辛烷值损失已经成为了汽油精制领域的一大挑战。本文主要采用皮尔森相关分析、遗传算法、LightGBM 算法和粒子群算法等，对数据进行了深入的挖掘，建立了主要变量对汽油精制过程中辛烷值损失的数量关系，具体做法如下：

针对问题 1，首先，以拉伊达准则为依据，剔除异常测量值，运用最大最小限幅法，剔除异常操作值；其次，我们设置一个数据缺失比例阈值，直接剔除有效数据量低于数据缺失比例的位点，其余数据缺失的位点以平均值法补全数据；最后得到 285 号和 312 号的修正数据（见表 5.2），填入附件一，以供后续研究。

针对问题 2，由于样本数据变量过多（多达 365 个），直接进行主要变量筛选存在较大困难。为此，我们采用二级逐次降维。首先，根据皮尔森相关系数过滤掉无关变量实现第一级变量降维，得到相关性较强的 57 个变量（见表 6.3）；接着，采用遗传算法过滤强耦连变量（冗余变量）实现第二级变量降维；然后，为了平衡数据之间的差异性，我们将数据进行数据清洗、归一化处理和完整度分析；此外，在算法中我们使用独热编码方案和能区分样本差异的余弦相似度作为适应度函数；最后，得到 30 个建立降低辛烷值损失模型的主要变量（见表 6.7）。

针对问题 3，结合样本数据中主要变量之间高度非线性和数量较多的特征，我们选择了非线性预测模型来建模主要变量和辛烷值损失之间的关系，再考虑到训练数据少，深度学习算法容易造成过拟合和计算资源多、运行时间长的实际，我们选取了普通机器学习中的轻型梯度提升机（LightGBM）算法来学习非线性模型中各变量的数量关系，得到了辛烷值损失预测模型（见 7.4 小节），同时，我们筛选出了线性回归、深度神经网络、决策树、支持向量机回归和梯度提升决策树等五种预测方法来对 LightGBM 算法进行对比，并利用 Python 和 Matlab 等工具，将对比结果可视化，直观地验证了 LightGBM 算法的有效性。

针对问题 4，基于问题 2 的主要变量筛选过程，我们首先计算各个变量对于硫含量的皮尔森系数剪裁无关变量，然后使用遗传算法筛选出了 30 个影响产品硫含量的主要变量，接着采用问题 3 中已验证的最优 LightGBM 算法，用于估算对应操作条件的硫含量。最后我们建立了辛烷值损失优化模型，对所有样本进行优化，利用全局寻优能力较强的粒子群算法求解，计算辛烷值损失降幅并给出降幅超过 30%的优化后操作条件，见附件 5。

针对问题 5，我们将操作变量调整过程的变化值梯度化，共 10 个梯度，这样有利于观察逐步调整的过程，也利于工厂的实际操作和自动化。最终，我们利用 python 画出从原始操作调整到辛烷值损失最优化后的操作过程中对应的汽油辛烷值和硫含量的变化轨迹，如图 9.2 和图 9.3 所示。

目录

1. 问题重述

1.1 问题背景

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。

依据从催化裂化汽油精制装置采集的 325 个数据样本，通过数据挖掘技术来建立汽油辛烷值（RON）损失的预测模型，并给出每个样本的优化操作条件，在保证汽油产品脱硫效果（为了给企业装置操作留有空间，本次建模要求产品硫含量不大于 5μg/g）的前提下，尽量降低汽油辛烷值损失在 30%以上。

1.2 需要解决问题

围绕降低汽油精制过程中辛烷值损失问题的相关信息，我们需要解决的问题依次如下：

问题 1. 数据处理：依据“样本确定方法”（附件二）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，供下面研究使用。

问题 2. 寻找建模主要变量：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用于建模的 30 个以下主要变量，使之尽可能具有代表性、独立性，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，以图形展示 133 号样本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. 问题假设

（1）原始样本数据不存在数据属性不完整、数据不一致和噪声数据等问题；
（2）问题 1 预处理完以后的数据是合乎后续要求的；
（3）问题 2 中的适应性函数符合后续处理的要求，筛选出的变量具有代表性和独立性；
（4）问题 3 中训练好的辛烷值损失预测模型，能够满足后续问题的要求。

3. 符号说明

符号	意义
v_i	被测变量第 i 次采样的剩余误差
$Cosine_{Sim(s_1,s_2)}$	s_1与s_2样本的余弦相似度
fit	适应度得分
p_l表示	种群中第l个基因型（个体）的适应度
X	训练集数据
$\tilde{V}_j(d)$	第j个变量的信息增益
g_i	第i个数据样本的损失函数的负梯度方向
$n_l^j(d)$	小于等于分割点的样本数
$n_r^j(d)$	大于分割点的样本数
ε_{gen}^{GOSS}	$GOSS$的泛化误差
RMSE	均方根误差
MAE	平均绝对误差
$h(x)$	辛烷值损失降幅
c	学习因子

4. 问题分析

4.1 问题 1 分析

问题 1 要求我们将 285 号和 313 号数据进行预处理，并将处理后的数据加入到附件一中，以便于提高主要变量数量关系模型的精度。

此问题的难点在于因设备与操作问题，原始数据可能存在一定的误差，不同数据的采集时间不同步，而这些将直接影响到后续模型的精度，因此，我们需要对原始数据进行清洗操作。具体步骤为：

1）以拉伊达准则为依据，采用最大最小限幅方法，剔除出其中的异常数据；

2）删除残缺数据过多的变量；

3）将清洗后的数据的平均值作为样本数据。

4.2 问题 2 分析

问题 2 要求根据附件一所提供的 325 个样本数据，筛选出影响辛烷值损失的代表性变量，并尽量确保变量之间相互独立，来提高模型精度。

该问题的难点在于：原始样本数据变量太多（367 个变量），且存在 1）有些变量与目标（降低辛烷值损失）之间的关联性低，2）有些变量冗余且具有相互强耦连性；这些问题影响模型的收敛与训练时间，为了解决这些难题，我们采筛选出关联性较强的主要变量，具体步骤如下：

1）使用皮尔森相关系数法，分析各个变量与目标降低辛烷值损失的相关性；

2）设定相关性阈值，剔除与降低辛烷值损失目标相关性较低的变量（即：无关变量），实现对样本变量的第一次降维；

3）使用遗传算法，从剩余变量中，筛选出 30 个以内的主要变量，实现对样本变量的第二次降维。

4.3 问题 3 分析

问题 3 要求我们采用问题 2 的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。也就是说需要我们建立辛烷值（RON）损失与主要变量之间的数量关系，然后利用合适的度量指标对建立好的数学模型进行验证。该问题的主要难点在于：选择何种机器学习算法，如何验证该算法对解决本问题的优越性。为了突破上面问题，我们分如下两个方面来进行分析：

一、预测模型的选取

1）线性预测模型与非线性预测模型

由于炼油设备的多样化，工艺过程的复杂性，使得操作变量（控制变量）之间具有高度非线性，同时主要操作变量相对较多，且与辛烷值（RON）损失之间的关系比较复杂，若采用线性预测模型（例如线性回归（Linear Regression）等）可能无法较好地满足建模的需求，也难以达到理想的效果。因此，我们优先考虑利用非线性预测模型来建立辛烷值（RON）损失预测模型。

2）普通机器学习预测模型与深度学习预测模型

基于机器学习算法的高效的组织和拟合参数能力，我们采用机器学习方法来解决非线性建模问题。考虑到本题中用于建模的样本数只有 325 个，经过问题二的筛选降维后，特征变量由 367 个变为了 30 个，数据量相对较少，使用深度学习的方法很容易使模型无法完成充分的训练，容易造成过拟合，且相对来说需要更多的计算资源和运行时间，因此我

们放弃自动提取特征的深度学习算法，采用普通机器学习算法来构建辛烷值损失预测模型。

结合本题的样本数据特征，通过综合分析，本题拟选择用轻型梯度提升机（Light Gradient Boosting Machine）算法来构建辛烷值损失预测模型。

二、对比实验分析验证法

同时，我们也注意到其他普通的机器学习预测模型各具千秋，在本题中同样具备一定的优势，因此我们很难从理论的角度进一步论证 LightGBM 算法的有效性。

基于以上分析，我们将从实际效果的角度出发，采用对比实验分析法对本题拟采用的辛烷值（RON）损失预测模型进行实验验证。为此，我们一共选取了五种方法与 LightGBM 进行对比：一种线性预测方法：线性回归（Linear Regression）、一种深度学习预测方法：深度神经网络（Deep Neural Networks）、三种普通的机器学习方法：决策树（Decision Tree）、支持向量机回归（Support Vector Machine Regression）、梯度提升决策树（Gradient Boosting Decision Tree）。

前两种对比方法主要是为了验证上述分析所提出的线性预测模型和深度神经网络预测模型不适用于本题的观点，而后三种方法主要是为了根据实际的预测效果，进一步验证轻型梯度提升机（LightGBM）的有效性。此外，我们利用 Python 和 Matlab 等工具，将对比结果可视化，直观地验证了 LightGBM 算法的优越性。

4.4 问题 4 分析

问题四要求我们在保持优化过程中原料、待生吸附剂、再生吸附剂的性质不变和产品硫含量不大于 5ug/g 的条件下，给出辛烷值损失降幅大于 30%的主要变量优化操作方案。该问题有两个难点，其一为附件数据中硫含量与其他变量的关系不明确，其二为这是一个带复杂约束条件的最优化问题，用一般算法求解时，很容易陷入局部最优，为此，我们采用如下步骤来解决这些问题。

1）使用准确性高，训练速度较快的 Light GBM 算法对硫含量进行估计，筛选出影响产品硫含量的主要变量，以明确表达在优化过程中硫含量的限制条件。

2）使用粒子群优化算法智能避免局部最优。以产品辛烷值为目标函数，同时设定各类约束条件，得到辛烷值损失优化模型。由于解空间并非线性结构，存在很多峰谷型的局部最优值，采用粒子群优化算法，通过模拟群体智能避免局部最优，对参数进行简单的调整，就可以快速地得到全局最优解。

3）计算样本主要变量优化后的辛烷值降幅，给出辛烷值（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。

4.5 问题 5 分析

问题 5 要求我们在主要操作变量逐步优化调整过程中，对 133 号样本的汽油辛烷值和硫含量的变化轨迹给出可视化展示。

此问难点在于展示多个变量调整与辛烷值和硫含量的变化轨迹，操作变量较多；从外，由于工业装置为了平稳生产，优化后的操作主要变量只能逐步调整到位。这些因素造成模型可视化难度较大。

为此，我们将操作变量取值的变化梯度化，将梯度作为横坐标画出最优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹，从原始操作条件描绘到优化后的条件为止。以便将无法表达的 30 个变量维度通过梯度变化清晰地展示出来，从而观察逐步调整的过程。

5. 问题一（数据处理）

5.1 问题分析

问题 1 要求我们将 285 号和 313 号数据进行预处理，并将处理后的数据加入到附件一中，以便于提高主要变量数量关系模型的精度。

此问题的难点在于因设备与操作问题，原始数据可能存在一定的误差，不同数据的采集时间不同步，而这些将直接影响到后续模型的精度，因此，我们需要对原始数据进行清洗操作。具体步骤为：

1）以拉伊达准则为依据，采用最大最小限幅方法，剔除出其中的异常数据；

2）残缺数据处理；

3）将清洗后的数据的平均值作为样本数据。

5.2 数据处理

5.2.1 数据统计

附件三提供了 285 号和 313 号两个样本的原始数据，这些原始数据统计结果如表 5.1 和表 5.2 所示。

表 5.1 展示了 285 号和 313 号两个样本在不同采样点（非操作变量）的采样情况；在原料、待生吸附剂、再生吸附剂、脱硫汽油这四个采样点的采样时间都是 8 点这一时刻；此外，这些原始数据的完整度为 100%。

表 5.1 非操作变量数据统计表

样本编号	采样时间				完整度
	原料	待生吸附剂	再生吸附剂	产品	
285 号	2017-07-17 08:00:00				100%
313 号	2017-05-15 08:00:00				100%

表 5.2 展示 285 号和 313 号两个样本在操作变量这一采样点下的采样情况；由表可知，操作变量的采样时间区间为 2 个小时，采样频次为 3 分钟/次。由操作变量的采样时间区间和采样频次，我们可以计算出每个样本进行过 40 次采样。此外，操作变量的采样数据的完整度为 100%。

表 5.2 操作变量数据统计表

样本编号	采样时间区间	采样频次	完整度
285 号	2017-07-17 06:00:00- 08:00:00	3 分钟/次	100%
313 号	2017-05-15 06:00:00- 08:00:00	3 分钟/次	100%

材料提示设备部分位点以及一些操作存在问题，这导致原始数据可能存在一定的误差；此外，不同变量数据采集时间不同步。这些误差数据可能会对后续模型精度造成较大的影响。

为此，需要对原始数据进行清洗处理。具体地，我们做了如下的数据清洗工作。

5.2.2 数据清洗

在这一部分，我们经过三个操作步骤确定了最终的样本数据并将最终数据保存至附件一中。具体流程如图 5.1 所示。

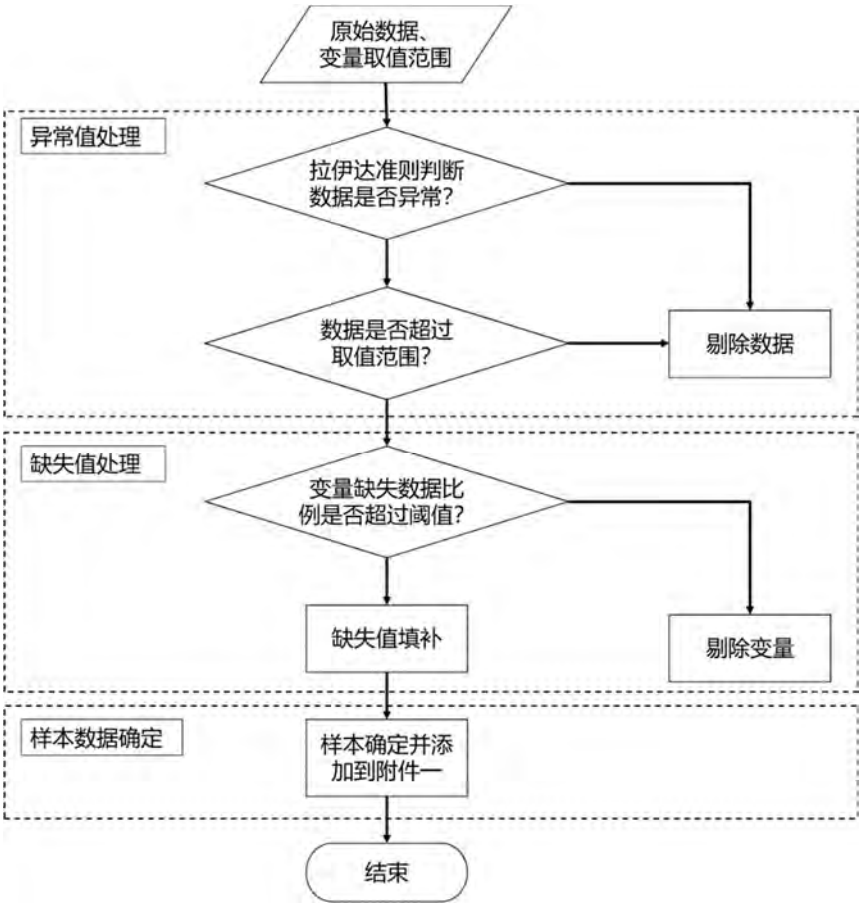


图 5.1 原始数据的清洗流程

步骤一：异常值处理。首先，根据拉伊达准则检测每个变量的异常值。具体地：

1）计算被测变量x的算术平均值$\bar{x}$以及被测变量各个属性值x_i的剩余误差v_i。$\bar{x}$和v_i的计算公式如下：

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{5.1}$$

$$|v_i| = |x_i - \bar{x}| \tag{5.2}$$

其中x_i表示被测变量的第 i 次采样的数据；n表示采样数据的个数，当采样点为非操作变量时，$n = 1$，；当采样点为操作变量时，$n = 40$。$\bar{x}$表示被测属性的算术平均值。

2）根据贝塞尔公式计算出被测变量的标准误差，贝塞尔公式如下所示：

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n} v_i^2\right]^{1/2} = \left\{\frac{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}{n-1}\right\} \tag{5.3}$$

其中x_i表示被测变量的第i次采样对应的数据值，σ表示为这一被测变量的标准误差。

3）计算出被测变量的标准误差后，我们利用拉伊达准则识别异常值数据，拉伊达准则如下：

$$|v_i| > 3\sigma \tag{5.4}$$

其中$|v_i|$表示x_i的剩余误差。如果x_i的剩余误差大于3σ，该采样数据将被判定为异常值。

因为附件三中非操作变量只进行一次采样，所以采样值的剩余误差一定满足$|v_i| < 3\sigma$。因此，只需对操作变量的采样数据进行拉伊达准则检测。最终，我们发现所有操作变量中

不存在剩余误差过大的异常采样数据。

接着，根据最大最小限幅方法检测每个变量的异常值。

1）根据[1-4]给出的非操作变量取值范围以及附件四提供的操作变量取值范围，我们得到了原始数据中各个变量的取值范围。

2）当变量采样数值超出了这些操作范围，将其标识为异常数据并剔除。

经过最大最小限幅方法检测，我们发现并没有变量的取值超过取值范围。

步骤二：缺失值处理。经过异常值处理之后，采样数据可能存在空值数据，需要进行缺失值处理。受[4-6]这些工作的启发，设置一个缺失比例阈值thd_{mis}。本文数据残缺比例阈值设置为$thd_{mis}=0.95$。如果被测变量的数据缺失比例超过thd_{mis}，从样本数据中剔除该变量所有数据。如果被测变量的数据缺失比例小于thd_{mis}，执行缺失值填补操作。填补操作为采用最近两个采样值的平均值进行填补。公式如下所示：

$$x_i=\frac{x_{i-1}+x_{i+1}}{2} \tag{5.5}$$

步骤三：样本数据确定。由表 5.1 可知，因为非操作变量在 2 个小时内只采样一次，所以非操作变量的采样数据直接作为最终数据。由于操作变量在 2 个小时内采样 40 次数据，所以需要将 2 个小时内操作变量采样数据的平均值作为最终操作变量数据。表 5.3 展示了 285 号和 313 号样本的最终数据。

表 5.2 样本各变量最终数据

变量名称	单位	样本取值	
		285 号样本	313 号样本
硫含量	μg/g	199	392
辛烷值	–	89.3	90.3
饱和烃	v%	60.06	55.05
⋮	⋮	⋮	⋮
8.0MPa 氢气至反吹氢压缩机出口	–	5984749.325	2154163.85
D101 原料缓冲罐压力	–	-97.2106975	-113.3759175

经过步骤三后，我们将处理后的 285 号和 313 号样本数据添加到附件一中。

6. 问题二（建模主要变量筛选）

6.1 问题分析

问题 2 要求根据附件一所提供的 325 个样本数据，筛选出影响辛烷值损失的代表性变量，并尽量确保变量之间相互独立，来提高模型精度。

该问题的难点在于：原始样本数据变量太多（367 个变量），且存在 1）有些变量与目标（降低辛烷值损失）之间的关联性低，2）有些变量冗余且具有相互强耦连性；这些问题影响模型的收敛与训练时间，为了解决这些难题，我们采筛选出关联性较强的主要变量，具体步骤如下：

1. 使用皮尔森相关系数法，分析各个变量与目标降低辛烷值损失的相关性；

2. 设定相关性阈值，剔除与降低辛烷值损失目标相关性较低的变量（即：无关变量），实现对样本变量的第一次降维；

3. 使用遗传算法，从剩余变量中，筛选出 30 个以内的主要变量，实现对样本变量的第二次降维。

6.2 数据说明与处理

6.2.1 数据说明

经过部分 5 的操作，得到 325 个样本的数据，该数据的统计结果如表 6.1 所示。表 6.1 中表明样本个数为 325，每个样本具有 7 个原料性质、2 个待生吸附剂性质、2 个再生吸附剂性质以及 354 个操作变量。此外，样本数据完整度达到 100%。

表 6.1 数据统计结果

样本数	原料性质个数	待生吸附剂性质个数	再生吸附剂性质个数	操作变量个数	完整度（%）
325	7	2	2	354	100

6.2.2 数据清洗

附件一中提供的样本数据不存在数据属性不完整、数据不一致和噪声数据等问题。因此，数据不用进行数据清洗。

6.2.3 归一化处理

为了消除不同类别差距在数据上的影响，本文将使用的每个输入进行归一化处理，数据归一化后可以方便处理后面数据，保证程序运行加快。归一化方法有两种形式，一种是把数变为（0，1）之间的小数，一种是把有量纲表达式变为无量纲表达式。主要是为了数据处理方便提出来的，把数据映射到 0～1 范围之内处理，更加便捷快速[18]。在本题中，我们采用较为简单的（0，1）标准化方法，其计算公式如下：

$$x_{normalization} = \frac{x - Min}{Max - Min} \tag{6.1}$$

其中x表示变量的值，Min表示该类变量中的最小值，Max表示该类变量中的最大值，$x_{normalization}$表示归一化的值。

6.3 模型建立与求解

样本数据中影响辛烷值损失的变量较多（多达 367 变量），这些变量可以分类为以下几类变量：

1）无关变量：与目标变量辛烷值损失无关的变量；

2）相关变量：与目标变量辛烷值损失相关的变量；特别地是决定辛烷值损失的工艺条件和化学性质等操作变量。例如：[1,4]中提出的变量（再生吸附剂持硫率、氢油比、反应器质量空速、反应器底部压力、反应器底部温度）。

3）冗余变量：根据文献[4-6]，有些变量取值完全取决于前面其他操作条件，它只是一个测量值，不能直接改变。例如：原料中的辛烷值、产品中的辛烷值两个变量就是相互冗余的，通过它们中任意一个变量并结合目标变量辛烷值损失就可以推断出另一个变量。

无关变量会造成降低辛烷值损失模型不易收敛，训练时间过长；而冗余变量会导致筛选的主要变量不具代表性或独立性。这些问题都会影响到后续降低辛烷值损失模型的准确度。为了有效地选取主要变量，我们采用“两阶段降维”。为了方便描述这一过程，我们将此过程定义为主要变量筛选模型。主要变量筛选模型的构建过程分为两个部分：1）无关变量剔除、2）主要变量筛选。

6.3.1 无关变量剔除

为了剔除样本变量中的无关变量，实现样本变量的第一次降维，我们采用皮尔森相关系数。皮尔森相关系数可以用来形容两变量之间是否存在线性关系（即：两个变量之间的变化趋势是否同一）。具体来说，1.利用皮尔森相关系数计算各个变量与辛烷值的降低之间的相关性。2.设定相关性阈值，用于剔除相关性较低的变量。

皮尔森相关系数计算公式如下：

$$P = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N\sum x_i^2 - (\sum x_i)^2}\sqrt{N\sum y_i^2 - (\sum y_i)^2}} \tag{6.2}$$

其中i表示样本标号，x_i表示第i个样本中某个变量的值，y_i表示第i个样本的辛烷值损失。注意：x可以是 367 个变量中任意一个变量。$0 \le P \le 1$，其中不同大小的皮尔森相关系数对应两个变量之间的相关性；皮尔森相关系数越大，说明两个变量之间的相关性越高。表 6.2 给出了皮尔森相关系数对应相关性的强弱。

表 6.2 皮尔森相关系数判断范围

系数范围	相关强度
0.0~0.2	极弱相关或无相关
0.2-0.4	弱相关
0.4-0.6	中等程度相关
0.6-0.8	强相关
0.8-1.0	极强相关

根据公式 6.2，我们利用皮尔森相关系数计算了 367 个变量与目标变量辛烷值损失之间的相关性，如图 6.1 所示。

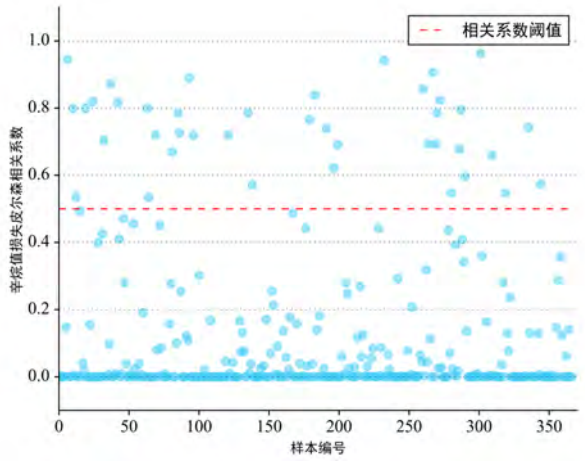


图 6.1　各变量与辛烷值损失之间的相关性强弱

如图 6.1 所示，散点主要分布在红线部分以下，因而图的下部分颜色较深。这表明，大部分变量与辛烷值损失的相关性较低，这也证实了我们之前的猜想（即：原始样本中可能很多变量属于无关变量）。为了消除这些无关变量对我们主要变量筛选模型的影响，我们将相关性低于相关性阈值的变量从样本中剔除。基于[7]，选取 0.5 作为相关性阈值。经过计算得到与辛烷值损失相关性低于 0.5 的变量有 310 个，不小于 0.5 的变量为 57 个。然后，进行第一次变量降维（即：将与辛烷值相关性低于 0.5 的变量剔除），我们留下 57 个候选变量。同理地，在样本数据执行列删除操作，将 310 个变量对应的列从原始数据中删除；清理后的数据保存为附件 1。表 6.3 展示了我们第一次降维后的变量。

表 6.3　57 个相关变量

位号	变量中文名称	位号	变量中文名称
原料性质	辛烷值 RON	TE_5004.DACA	稳定塔顶出口温度
待生吸附剂性质	S, wt%	TE_5006.DACA	稳定塔底出口温度
再生吸附剂性质	S, wt%	TE_5003.DACA	C-201#37 层塔盘温度
CAL_H2.PV	氢油比	SIS_PT_2602.PV	再生器顶部/再生器接收器差压
PT_2801.PV	还原器压力	SIS_TE_2605.PV	再生器下部温度
TE_2005.PV	反应器底部温度	PT_6009.DACA	预热器空气出口压力
TC_5005.PV	稳定塔下部温度	TE_6002.DACA	烟气出辐射室温度
TE_5102.PV	干气出装置温度	PT_1101.DACA	循环氢压缩机去混氢点压力
PT_9301.PV	蒸汽进装置压力	FT_3501.DACA	循环氢至闭锁料斗料腿流量
FT_9101.PV	污油出装置	SIS_TE_6009.PV	预热器入口烟气温度
PT_9001.PV	燃料气进装置压力	TE_6008.DACA	空气预热器空气出口温度
TE_9301.PV	1.0MPa 蒸汽进温度	FC_1104.DACA	进料调节阀旁路流量
PT_9401.PV	净化风进装置压力	BS_AT_2401.PV	闭锁料斗烃含量
SISTE6010.PV	加热炉排烟出口温度	LC_2601.DACA	R-102 床层吸附剂料位密度
AC_6001.PV	加热炉氧含量	TE_2604.DACA	R-102 #1 通风挡板温度
TE_1608.PV	加热炉循环氢出口温度	TE_2004.DACA	R-101 床层下部温度
PC_1301.PV	K101 机出口压力	PDT_3502.DACA	ME-109 过滤器差压
LC_1201.PV	D104 液面	PDT_1004.DACA	ME-104 出入口
FC_1202.PV	D121 顶去放火炬流量	PT_7510B.DACA	K-103B 排气压力

AI_2903.PV	再生烟气氧含量	PT_7508B.DACA	K-103B 进气压力
FT_5104.TOTAL		PT_1604.DACA	F-101 长明灯线压力
FT_1501.TOTAL	新氢进装置流量	PT_1601.DACA	F-101 出口总管压力
FT_1204.PV		TE_1602.DACA	F-101 出口支管#1 温度
PT_1102.DACA	E-101 管程入口压力	TE_5007.DACA	E-203 壳程出口管温度
TE_1104.DACA	E-101F 管程出口温度	AT-0012.DACA.PV	S_ZORB AT-0012
LI_9102.DACA	D-204 液位	FT_5204.DACA.PV	汽油产品去气分流量
LT_3101.DACA	D-124 液位	FT_1503.DACA.PV	氢气至循环氢压缩机入口
LT_1501.DACA	D-122 液位液位	FT_1504.DACA.PV	氢气至反吹氢压缩机出口
PDT2503.DACA	D-107 排放滑阀压差		

6.3.2 主要变量筛选

题设给出操作变量之间具有高度非线性和相互强耦连的关系，因此仅仅使用 Pearson 相关系数筛选主要变量是不合适的，因为 1）皮尔森相关系数只考虑了单个变量与辛烷值损失之间的关系，而变量之间的影响、变量的组合对辛烷值的影响都没有考虑；2）皮尔森相关系数在一定程度上会保护冗余变量。

遗传算法 1）从问题解的串集开始搜索；2）同时处理群体中的多个个体，减少了陷入局部最优解的风险以及 3）可以考虑变量之间的相互影响以及不同变量组合对目标的影响。为了有效地选出 30 个以内的主要变量，本研究采用遗传算法。

1. 编码

遗传算法不能直接处理问题空间的参数，需要通过编码将待求解问题表示成遗传空间中的染色体或个体。这一转换过程称为遗传算法的编码，或称作问题表征[8]。此外，编码策略的是否可行决定了遗传算法能否有效工作。为了评估编码策略的可行性，常采用以下 3 个准则：

完备性:问题空间中的所有候选解都能转换成遗传空间中的点(个体或染色体)。

健全性:遗传空间中的染色体必须都能对应到问题空间中的候选解。

非冗余性:染色体（或个体）和候选解一一对应。

当自定义遗传算法编码策略同时满足这三个准则，则说明拟定的编码策略具备可行性。

本研究中，问题空间为从 57 个变量中找出 30 个主要变量，可行解为不同的 30 个变量的组合。为了将问题空间转换为遗传空间，我们采用独热码对不同可行解进行编码。

因为可行解中的 30 个变量是从 57 个变量中选择出来的，所以用一个维度为 57 的向量（即：一个个体或基因型）表示一个可行解，当一个变量被选择，则基因型对应的分量标记为 1，否则变为 0。例如：一个可行解的变量组合为$(1,2,4,\cdots,57)$，那么它的基因型为$X=(1,1,0,1,\cdots,1)$。因为该可行解选择了第 1 个变量，所以基因型对应的第一个分量是 1，然而该可行解没有选择第三个变量，所以基因型对应的第三个分量为 0。此外，当我们得到一个基因型，我们可以反向推导（即：解码）可以得到唯一的可行解，这说明我们采用的独热编码策略满足可行性。

表 6.3 遗传算法所用术语

遗传术语	算法术语	本文对象
群体	可行解集	变量的不同组合
个体	可行解	一个变量组合
染色体(基因型)	可行解编码	变量组合的独热编码
基因	可行解编码的分量	独热码中的一个分量

2. 适应度函数

遗传算法的适应度函数是一个用于评估个体优劣的指标函数，它根据问题的目标函数来进行评估的。遗传算法在搜索进化过程中仅通过适应度函数来判断个体优劣，并将此作为后续遗传操作的依据。在此基础上，适应度函数排序并选择概率，所以适应度函数为非负函数。此外，适应度函数的设计需要满足这些条件：单值、连续、合理、一致性、计算量小、通用性强。

本研究中，我们的目标函数为辛烷值损失，不同样本表现出不同的辛烷值损失，我们需要找出与辛烷值相关性最强的 30 个主要变量。当同一变量下不同样本之间的值相差很大，而辛烷值却相差无几；这说明这一变量并不能有效区分样本，也不会影响到辛烷值的变化。当同一变量下不同样本之间的值相差很大，同时辛烷值也相差较大；此时，我们可以该变量值高的样本划分一类（这一类样本中辛烷值损失很高或者很低），低的分为另一类。这种情况下，辛烷值变化受该变量值影响较大，同时样本也被有效区分。因此，为了找到这些可以有效地影响辛烷值的变量可以转换为找到有效区分样本的变量这一问题。

我们适应度函数的目标是选择出来的变量能够有效地区分样本之间的细微差异。为了实现这一目的，我们基于归一化余弦相似度构建适应度函数。具体来说，给定一个基因型，例如：$X = (1,1,0,1,\cdots,1)$，将基因型分量为 0 的变量从样本中删除，得到新样本。此时一个样本的变量个数为 30，然后将任意两个样本$s_1, s_2 \in S, |S| = 365$，进行余弦相似度计算。余弦相似度计算公式如下：

$$Cosine_{Sim(s_1,s_2)} = \frac{\sum_{i=1}^{30} s_{1_i} \times s_{2_i}}{\sqrt{\sum_{i=1}^{30}(s_{1_i})^2} \times \sqrt{\sum_{i=1}^{30}(s_{2_i})^2}} \tag{6.3}$$

其中s_{1_i}表示样本s_1在第i个变量下的值，$i \in [1,30]$，$Cosine_Sim(s_1, s_2)$表示两个样本之间的余弦相似度且$Cosine_Sim(s_1, s_2) \in (-1,1)$。

$$sim(s_1, s_2) = \frac{1 + Cosine\_Sim(s_1, s_2)}{2} \tag{6.4}$$

$$fit = 1 - \frac{\sum_{j=1}^{365}\sum_{k=1}^{365} sim(s_j, s_k)}{365 \times 365} \tag{6.5}$$

其中$sim(s_1, s_2)$表示两个样本之间的归一化余弦相似度且$sim(s_1, s_2) \in (0,1)$。fit表示该个体（变量组合）的适应度且$fit \in (0,1)$。因为存在多组样本，如果只选用其中一项的归一化余弦相似度作为评估值，这样无法准确评估该个体是否优劣。因此，我们采用 365 个样本两两之间的归一化余弦相似度的平均值作为评估值。适应度fit越低表示该变量组合下样本之间越相似。因此，高适应度将会是个体（变量组合）的进化趋势。

我们的适应性函数值介于 0 到 1 之间属于非负函数，此外，单值（具有唯一的值）、连续、合理、一致性、计算量小、通用性强等特性也都满足。因此，我们的适应性函数设计合理。

3. 遗传算法运算流程

在确定合理的编码方案以及适应度函数之后，我们便可以使用遗传算法筛选主要变量。具体步骤如图所示。

图 6.2 遗传算法流程图

步骤一：设定遗传算法模型参数：种群大小、染色体长度、交叉概率、变异概率以及终止条件，本研究中参数设定如表 6.4 所示。

1）种群大小是指一代群体中的个体总数，种群大小一般设置为 10-200，本研究中设置一代个体数为 60。

2）染色体长度为变量个数（即：57）。

3）适应度阈值（选择阈值）设置为 0.5，高于 0.5 说明个体适应度较高，该个体较大概率会被选择。选择概率定义如下：

$$p_l = \frac{fit_l}{\sum_{l=1}^{60}(fit_l)} \tag{6.6}$$

其中p_l表示种群中第l个基因型（个体）的适应度且$p_l \in (0,1)$，fit_l表示第l个基因型（个体）的适应度得分。

4）交叉操作中的概率是用于确定两个个体之间是否进行基因交叉操作，一般都会大于 0.9；因此本研究中采用 0.92 作为交叉概率。

5）变异操作的概率是允许少数个体存在变异情况，以避免限入局部最优解，其值一般都在 0.1 以下，因此本文选择 0.05 作为变异概率。

6）当个体的最优适应度达到给定的阈值，或者迭代次数达到设定的代数时，遗传算法终止。一般代数设置为 100-1000 代。本文选择 200 代作为终止条件。

表 6.4 遗传算法参数设定

种群大小	染色体长度	选择阈值	交叉概率	变异概率	终止规则
60	57	0.5	0.92	0.05	200

步骤二：初始化种群。遗传算法中初始群体中的个体是随机产生的。本研究中，初始群体的设定采取策略为：先随机生成 60 个体，然后从中挑出最好的个体加到初始群体中。这个过程不断迭代，直到初始群体中个体数达到了预先确定的规模，设置进化代数$t = 0$。

步骤三：个体适应度计算和适应度排序。根据公式 6.5 计算群体中每个个体的适应度。然后，根据排序算法将适应度从大到小进行排序。

步骤四：选择操作。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。本研究中，会找到p_l位于 0.5 邻域位置的个体，当$p_l > 0.5$，则前l个个体会遗传到下一代；否则前$l-1$个个体遗传到下一代。

步骤五：交叉操作。对选中的成对个体，以 0.92 的概率交换部分基因。交换基因个数可以随机确定，但是交换后的有效的基因（基因型中为 1 的分量）数不变。例如：$X_1 = (1,1,0,1,\cdots,1)$，$X_2 = (0,1,1,1,\cdots,0)$。如图 6.3 所示，两个个体之间发生基因交叉操作必须是成对基因发生交换，例如：第一对和第三对基因发生交叉操作，得到$X_1' = (0,1,1,1,\cdots,1)$和$X_1' = (1,1,1,0,\cdots,1)$。而仅是第一对基因和第二对基因发生基因交叉是不被允许的，因为第一个个体的有效基因变为 29，而第二个基因变为 31。

$$X_1 = (1,1,0,1,\cdots,1)$$

1 2 3 4 5

$$X_2 = (0,1,1,1,\cdots,0)$$

图 6.3　基因交换操作

步骤六：变异操作。变异是变动群体中的某些个体中某些基因位置上的值。变异在遗传算法中属于辅助性的操作，它的目的为保持群体的多样性。注意：变异操作是一个个体的成对基因变异，这是为了保证变异后的有效基因个数仍是 30。例如：$X_1 = (1,1,0,1,\cdots,1)$中第三位和最后一位基因发生变异得到$X_1' = (1,1,1,1,\cdots,0)$。设置较低的概率是保护群体中重要基因的丢失。例如：群体中存在多个个体都具有第一位基因，这时有个体发生第一位基因变异，如果变异基因导致适应度变差，则说明此次变异是无益变异，这种变异会被摒弃；而有益变异会因为适应度变好从而会继续遗传下去。如果群体大量个体发生无益变异时，这可能导致群体消失（不能找到可行解）。

群体经过选择、交叉、变异操作之后得到下一代群体。

步骤七：终止条件判断。如果迭代次数达到我们设定的值，即$t = 200$，则将进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

表 6.3 展示了我们第二次降维后的变量，我们得到的最终子代群体以及它们的相关信息如表 6.4 所示。

表 6.3　30 个主要变量

位号	变量中文名称	位号	变量中文名称
原料性质	辛烷值 RON	PDT_2503.DACA	D-107 底排放滑阀压差
再生吸附剂性质	S, wt%	TE_5004.DACA	稳定塔顶出口温度
CAL_H2.PV	氢油比	TE_5003.DACA	C-201#37 层塔盘温度
TE_2005.PV	反应器底部温度	SIS_PT_2602.PV	再生器顶部/再生器接收器差压
TC_5005.PV	稳定塔下部温度	SIS_TE_2605.PV	再生器下部温度
TE_5102.PV	干气出装置温度	TE_6002.DACA	烟气出辐射室温度
FT_9101.PV	污油出装置	SIS_TE_6009.PV	预热器入口烟气温度
TE_9301.PV	1.0MPa 蒸汽进装置温度	TE_6008.DACA	空气预热器空气出口温度
SIS_TE_6010.PV	加热炉排烟出口温度	LC_2601.DACA	R-102 床层吸附剂料位密度
TE_1608.PV	加热炉循环氢出口温度	TE_2604.DACA	R-102 #1 通风挡板温度
FC_1202.PV	D121 顶去放火炬流量	TE_2004.DACA	R-101 床层下部温度
AI_2903.PV	再生烟气氧含量	PDT_1004.DACA	ME-104 出入口
FT_1501.TOTAL	新氢进装置流量	FT_5204.DACA.PV	汽油产品去气分流量
TE_1104.DACA	E-101F 管程出口管温度	FT_1503.DACA.PV	氢气至循环氢压缩机入口
LI_9102.DACA	D-204 液位	FT_1504.DACA.PV	氢气至反吹氢压缩机出口

表 6.4 最终群体相关数据

最终子代群体	适应度	占总数的比例
(1,1,1,1,0,⋯,1,1)	0.1783	0.693
(1,1,1,1,0,⋯,0,1)	0.1277	0.115
⋮	⋮	⋮
(1,1,1,1,1,⋯,0,0)	0.0365	0.008

7. 问题三（辛烷值（RON）损失预测模型的建立）

7.1 问题分析

问题 3 要求我们采用问题 2 的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。也就是说需要我们建立辛烷值（RON）损失与主要变量之间的数量关系，然后利用合适的度量指标对建立好的数学模型进行验证。该问题的主要难点在于：选择何种机器学习算法，如何验证该算法对解决本问题的优越性。为了突破上面问题，我们分如下两个方面来进行分析：

一、预测模型的选取

1）线性预测模型与非线性预测模型

由于炼油设备的多样化，工艺过程的复杂性，使得操作变量（控制变量）之间具有高度非线性，同时主要操作变量相对较多，且与辛烷值（RON）损失之间的关系比较复杂，若采用线性预测模型（例如线性回归（Linear Regression）[9]等）可能无法较好地满足建模的需求，也难以达到理想的效果。因此，我们优先考虑利用非线性预测模型来建立辛烷值（RON）损失预测模型。

2）普通机器学习预测模型与深度学习预测模型

基于机器学习算法的高效的组织和拟合参数能力，我们采用机器学习方法来解决非线性建模问题。考虑到本题中用于建模的样本数只有 325 个，经过问题 2 的筛选降维后，特征变量由 367 个变为了 30 个，数据量相对较少，使用深度学习的方法很容易使模型无法完成充分的训练，造成过拟合，影响预测效果，且相对来说需要更多的计算资源和运行时间[10]，因此我们放弃自动提取特征的深度学习算法，采用普通机器学习算法来构建辛烷值损失预测模型。

轻型梯度提升机（Light Gradient Boosting Machine）[11]是普通机器学习预测模型的典型代表之一，由微软公司于 2017 年提出，大量的研究[12,13]已经表明其相对于传统的机器学习预测模型具有更快的训练效率和更高的准确率，不仅可以在低内存的情况下使用，而且支持并行化学习。结合本题的样本数据特征，通过综合分析，本题拟选择用轻型梯度提升机（LightGBM）算法来构建辛烷值损失预测模型。

二、对比实验分析验证法

同时，我们也注意到其他普通的机器学习预测模型各具千秋，在本题中同样具备一定的优势，因此我们很难从理论的角度进一步论证 LightGBM 算法的有效性。

基于以上分析，我们将从实际效果的角度出发，采用对比实验分析法对本题拟采用的辛烷值（RON）损失预测模型进行实验验证，并进一步说明 LightGBM 模型相对于其他模型的优越性。为此，我们一共选取了五种方法与 LightGBM 进行对比：

一种线性预测方法：线性回归（Linear Regression）；

一种深度学习预测方法：深度神经网络（Deep Neural Networks）[14]；

三种普通的机器学习方法：决策树（Decision Tree）[15]、支持向量机回归（Support Vector Machine Regression）[16]、梯度提升决策树（Gradient Boosting Decision Tree）[17]。

前两种对比方法主要是为了验证上述分析所提出的线性预测模型和深度神经网络预测模型不适用于本题的观点，而后三种方法主要是为了根据实际的预测效果，进一步验证 LightGBM 的有效性。具体的实验步骤如下所示：

1）根据模型的数据要求对本题的样本数据进行预处理，划分好对应的训练集和测试集，并设置合适的模型参数；

2）将训练集数据放入六个选取的模型中分别进行训练，得到对应的训练好的辛烷值（RON）损失预测模型；

3）将测试集放入模型中进行测试，选择合适的评价指标对模型进行评估与验证，并将对比结果进行可视化。

7.2 模型框架

在上述分析的基础上，我们分成三步来完成我们整个模型的辛烷值（RON）损失预测模型的建立、求解和验证，具体的模型框架如图 7.1 所示。

1）对问题二筛选降维以后的基础数据进行预处理，一共包括三个步骤：完整度分析、归一化处理和数据集划分。

2）利用训练集数据对基于轻型梯度提升机（LightGBM）的辛烷值（RON）损失预测模型进行训练，然后对测试集数据进行预测，得到对应的预测结果。

3）对五种对比方法进行训练和测试，得到所有预测结果以后，利用均方根误差 RMSE（Root Mean Squared Error）、平均绝对误差 MAE（Mean Absolute Error）和运行时间（Running Time）对结果进行分析论证。

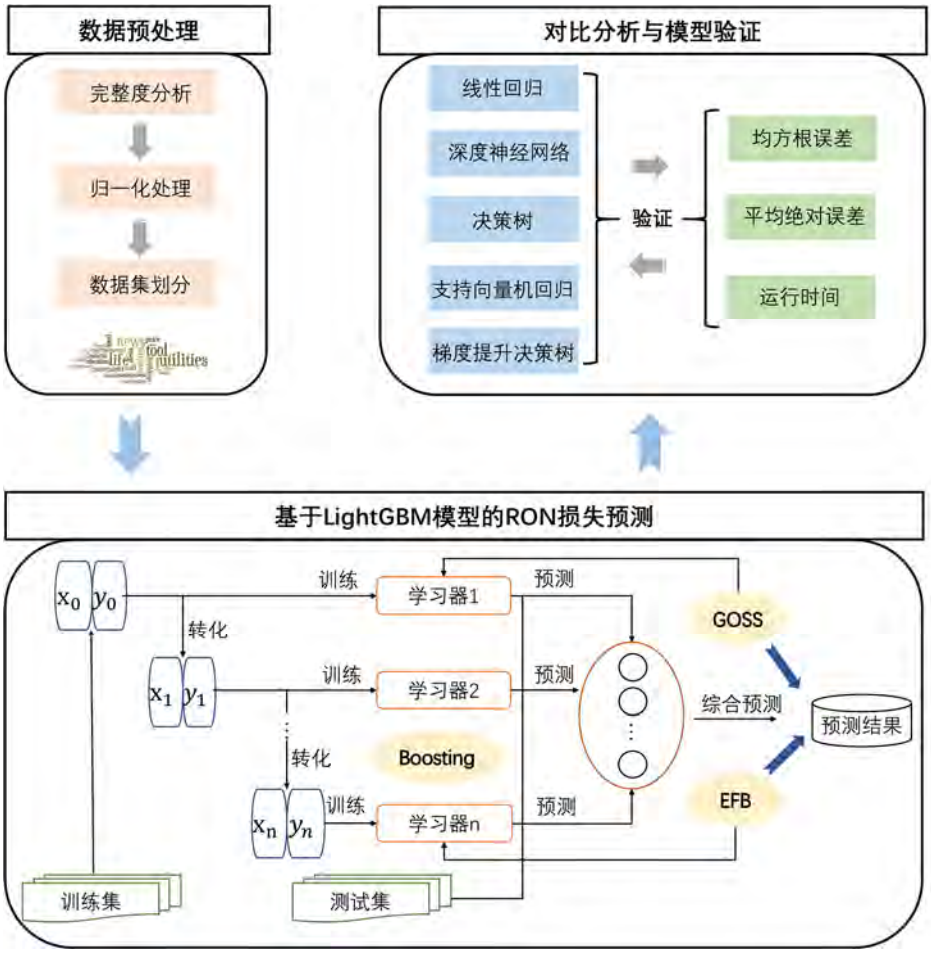


图 7.1 模型框架

7.3 数据分析与预处理

在经过问题 1 的数据处理和问题 2 的变量筛选以后，我们得到了包含 325 个样本数据，每个样本数据包括 30 个变量和 1 个辛烷值（RON）损失值，详细的数据信息请见附

件 3。由于这些数据都是从裂化汽油精致装置采集的真实数据样本，且已经进行了一些基本的处理，所以无需进行数据清洗。我们只需根据数据特征和后续各个模型的数据输入格式进行适当的分析和预处理即可，因此我们主要做了以下三个方面的工作：

a) 数据完整度分析

考虑到数据中存在一些空值的数据，我们需要计算每个变量的数据完整度，其计算公式如下所示：

$$I_i = \frac{N_i}{Sum_i} \tag{7.1}$$

其中，I_i 为第i个变量的完整度，N_i为第i个变量中非空数据的个数，Sum_i为第i个变量的样本个数。根据变量完整度计算公式，我们可以计算出各个变量的数据完整度如表 6.1 所示。

b) 归一化处理

为了消除不同类别差距在数据上的影响，本文将使用的每个输入进行归一化处理，数据归一化后可以方便处理后面数据，保证程序运行加快。归一化方法有两种形式，一种是把数变为（0，1）之间的小数，一种是把有量纲表达式变为无量纲表达式。主要是为了数据处理方便提出来的，把数据映射到 0～1 范围之内处理，更加便捷快速[18]。在本题中，我们采用较为简单的（0，1）标准化方法，其计算公式如下：

$$x_{normalization} = \frac{x - Min}{Max - Min} \tag{7.2}$$

其中x表示变量的值，Min表示该类变量中的最小值，Max表示该类变量中的最大值，$x_{normalization}$表示归一化的值。

c) 数据集划分

对于本题中的所有模型，我们将所有数据的 70%作为训练集，30%作为测试集。此外，为了后续预测结果的可视化，我们对每个样本按照其序号进行了编码。

7.4 基于 LightGBM 模型的 RON 损失预测

轻型梯度提升机（Light Gradient Boosting Machine）是一个快速的，分布式的，高性能的基于决策树算法的梯度提升框架。可用于排序，分类，回归以及很多其他的机器学习任务中。LigthGBM 是 Boosting 集合模型中的新进成员，由微软提供，它是对 GBDT 的高效实现，主要由单边梯度采样 GOSS（Gradient-based One-Side Sampling）算法和互斥特征绑定 EFB（Exclusive Feature Bundling）算法组成[11]。其优势主要表现在：更快的训练效率、低内存使用、更高的准确率、支持并行化学习、可处理大规模数据、支持直接使用类别特征[12]。本题将使用 LigthGBM 进行辛烷值损失预测建模，具体的建模步骤如下：

一、利用 Boosting 模型组合一个较好的辛烷值损失学习器

在这一步骤中，首先给定处理好的训练集数据$X = \{x_i, y_i\}_{i=1}^{n}$，我们的目的是通过 Boosting 算法找到一个大致的辛烷值损失值$\hat{f}(x)$，将特定的损失函数$L(y, f(x))$的期望值最小化，从而组合出一个较好的辛烷值损失学习器：

$$\hat{f}(x) = \arg minE_{y,X} L(y, f(x)) \tag{7.3}$$

我们通过集成 T 个回归树$\sum_{t=1}^{T} f_t(X)$来逼近最终的模型：

$$f_t(x) = \sum_{t=1}^{T} f_t(X) \tag{7.4}$$

然后我们可以应用不同分配下的基础的（机器）学习算法，每个算法都会生成一个弱规则，这是一个迭代的过程，多次迭代后，Boosting 算法可以将它们组合成一个强大的决策规则。为了选择正确的分配方式，可以遵循下面几个步骤：

步骤 1：给所有分布下的基础学习器对于每个观测值都分配相同的权重；

步骤 2：如果第一个基础的学习算法预测错误，则该点在下一次的基础学习算法中有更高的权重；

步骤 3：迭代第 2 步，直到到达预定的学习器数量或预定的预测精度。

最后，将输出的多个弱学习器组合成一个强的学习器，提高模型的整体预测精度。

二、利用单边梯度采样（GOSS）估计每个变量的信息增益

在 GOSS 中，首先根据数据的梯度将训练降序排序，然后保留 Top a 个数据实例，作为数据子集 A；接着对于剩下的数据的实例，随机采样获得大小为 b 的数据子集 B；最后通过以下方程估计信息增益:

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A: x_{ij} \le d} g_i + \frac{1-a}{b}\sum_{x_i \in B: x_{ij} \le d} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A: x_{ij} > d} g_i + \frac{1-a}{b}\sum_{x_i \in B: x_{ij} > d} g_i\right)^2}{n_r^j(d)}\right) \tag{7.5}$$

其中，d表示分割点，n表示总的训练集样本数，g_i表示第i个数据样本的损失函数的负梯度方向，$n_l^j(d)$表示小于等于分割点的样本数，$n_r^j(d)$表示大于分割点的样本数。此处 GOSS 通过较小的数据集估计信息增益$\tilde{V}_j(d)$，将大大地减小计算量。

因此，我们定义 GOSS 近似误差为：

$$\varepsilon(d) = \left|\tilde{V}_j(d) - V_j(d)\right| \tag{7.6}$$

$$\bar{g}_l^j(d) = \frac{\sum_{x_i \in (A \cup A^c)_l} |g_i|}{n_l^j(d)} \tag{7.7}$$

概率至少是$1-\delta_1$，有：

$$\varepsilon(d) \le C_{a,b}^2 \ln(1/\delta) * max\left\{\frac{1}{n_l^j(d)}, \frac{1}{n_r^j(d)}\right\} + 2 * D * C_{a,b}\sqrt{\frac{\ln(1/\delta)}{n}} \tag{7.8}$$

其中$C_{a,b} = \frac{1-a}{\sqrt{b}} max_{x \in A^c}|g_i|, D = \max\left(\bar{g}_l^j(d), \bar{g}_r^j(d)\right)$

考虑 GOSS 泛化误差$\varepsilon_{gen}^{GOSS}(d) = \left|\tilde{V}_j(d) - V_*(d)\right|$，这是 GOSS 抽样的实例计算出的方差增益与实际样本方差增益之间的差距，可变换为：

$$\varepsilon_{gen}^{GOSS}(d) = \left|\tilde{V}_j(d) - V_j(d)\right| + \left|V_j(d) - V_*(d)\right| = \varepsilon_{GOSS}(d) + \varepsilon_{gen}(d) \tag{7.9}$$

因此，在 GOSS 准确的情况下，GOSS 泛化误差近似于全量的真实数据。另一方面，采样将增加辛烷值损失学习器的多样性（因为每次采样获得的数据可能会不同），这将提高泛化性。

三、利用互斥特征绑定（EFB）降低冲突比率，提升计算效率

EFB 是通过特征捆绑的方式减少特征维度（其实是降维技术）的方式，来提升计算效率。通常被捆绑的特征都是互斥的（一个特征值为零,一个特征值不为零），这样两个特征捆绑起来才不会丢失信息。如果两个特征并不是完全互斥（部分情况下两个特征都是非零值），可以用一个指标对特征不互斥程度进行衡量，称之为冲突比率，当这个值较小时，我们可以选择把不完全互斥的两个特征捆绑，而不影响最后的精度[13]。

具体的算法步骤如下：

1、将特征按照非零值的个数进行排序；

2、计算不同特征之间的冲突比率；

3、遍历每个特征并尝试合并特征，使冲突比率最小化；

四、利用训练集数据进行模型训练，利用测试集数据进行模型测试

搭建好模型框架以后，我们将训练集数据放入 LigthGBM 模型中进行训练，并得到对应的模型。接下来，我们将测试集放入模型中进行测试和验证，为了结合其余的对比方法进行分析，具体的验证过程我们将在各个模型的对比分析部分给出。

7.5 对比分析与模型验证

7.5.1 对比方法

进一步验证轻型梯度提升机（LightGBM）的有效性，我们用 GWSC 比较了五种预测模型算法：一种线性预测模型：线性回归，一种深度学习预测模型：深度神经网络，三种普通的机器学习模型：决策树、支持向量机回归、梯度提升决策树。

- 线性回归（LR）[9]：线性回归是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析，这种函数是一个或多个称为回归系数的模型参数的线性组合。只有一个自变量的情况称为简单回归,大于一个自变量情况的叫做多元回归，本题将采用多元线性回归。
- 深度神经网络（DNN）[14]：深度神经网络是机器学习领域中一种技术，由多层神经元构成。通过前馈神经前向传播算法和反向传播算法来训练参数，最终得到训练好的模型。
- 决策树（DT）[15]：决策树是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。
- 支持向量机回归（SVR）[16]：支持向量机回归，顾名思义就是用支持向量机来进行回归预测，而支持向量机是一种按监督学习方式对数据进行二元分类的广义线性分类器，其决策边界是对学习样本求解的最大边距超平面。
- 梯度提升决策树（GBDT）[17]：梯度提升算法是一种通用的学习算法，除了决策树，还可以使用其它模型作为基的学习器。梯度提升算法的思想是通过调整模型，让损失函数的值不断减小，然后将各个模型加起来作为最终的预测模型。而梯度提升决策树则是以决策树为基的学习器。

7.5.2 主要参数说明

设置合适的参数对于机器学习方法的模型训练至关重要[19]，在本题中，为了切实保证实验结果的客观性和可对比性，本题中所用到的所有模型均采用其默认参数。各个模型的主要参数设置如表 7.1 所示。

表 7.1 主要参数统计表

方法参数名		参数值
线性回归（LR）	fit_intercept	'True'
深度神经网络（DNN）	hidden_layers	3
	hidden_layer_sizes	64
	activation	'Relu'
	learning_rate	0.001
	momentum	0.9
决策树（DT）	criterion	'mse'
	min_samples_split	2
	min_samples_leaf	1
支持向量机回归（SVR）	kernel	'rbf'
	degree	3
	epsilon	0.1
	cache_size	200
梯度提升决策树（GBDT）	learning_rate	0.05
	n_estimators	100
	subsample	1.0
	min_samples_split	2
	min_samples_leaf	1
	max_depth	3
轻型梯度提升机（LigthGBM）	num_leaves	31
	num_iterations	100
	learning_rate	0.05
	n_estimators	20

7.5.3 评价指标

在本小节中，我们采用三个评价指标对各个模型的预测结果进行评估:

a) 均方根误差（RMSE）

数理统计中均方根误差（RMSE）是指参数估计值与参数真值之差平方根的期望值。RMSE 是衡量"平均误差"的一种较方便的方法，RMSE 可以评价数据的变化程度，RMSE 的值越小，说明预测模型描述实验数据具有更好的精确度。RMSE 对一组测量中的特大或特小误差反映非常敏感，所以，它能够很好地反映出测量的精密度，因此在工程测量中广泛被采用。其计算公式如下：

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(R_t - P_t)^2} \tag{7.10}$$

其中，N表示样本数，R_t表示t样本对应的真实辛烷值损失值，P_t表示t样本对应的预测辛烷值损失值。

b) 平均绝对误差（MAE）

平均绝对误差（MAE）是所有单个观测值与算术平均值的偏差的绝对值的平均，平均绝对误差可以避免误差相互抵消的问题，因而可以准确反映实际预测误差的大小。此外，由于离差被绝对值化，不会出现正负相抵消的情况，因而，平均绝对误差能更好地反映预

测值误差的实际情况。其计算公式如下：

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|R_t - P_t| \tag{7.11}$$

其中，N表示样本数，R_t表示t样本对应的真实辛烷值损失值，P_t表示t样本对应的预测辛烷值损失值。

c) 运行时间（RT）

模型运行时间，即模型从开始训练到完成测试的时间。

7.5.4 实验结果展示及分析

本小节将对各个方法的预测结果进行分析和可视化展示。具体来说，我们将训练集放入各个模型进行训练，待模型训练好以后，我们将测试集输入模型，将最终得到的结果与真实值比较，计算出他们的均方根误差、平均绝对误差和运行时间，三种评价指标的值越小代表预测结果越好，所有的实验结果如表 7.2 和图 7.2 所示。同时，为了更直观地对比各个模型的预测结果，我们将各个样本的编号作为横坐标，对应的辛烷值损失作为纵坐标，画出了各个模型对应的散点图（如图 7.3(a-f)所示），通过展示训练集、测试集和原始数据的辛烷值损失对比，分析出各个模型的训练及测试的效果。具体来说，可总结为以下几点：

1）线性回归（LR）方法的预测效果远不如其他非线性预测模型，但是由于其本身的模型并不复杂，需要训练的参数较少，所以运行时间相对来说是最少的。该实验结果进一步验证了本题的应用场景不适合用线性预测模型的观点。

2）深度神经网络（DNN）的预测结果要低于其他普通机器学习预测模型，且由于其需要训练大量参数，其运行时间是最多的。该实验结果进一步验证了本题的应用场景不适合用深度学习预测模型的观点。

3）LigthGBM 方法无论是预测效果还是运行时间都要明显优于其余模型。该实验结果这进一步验证了 LigthGBM 模型在本题应用场景中的有效性。

表 7.2 实验结果统计表

方法	均方根误差	平均绝对误差	运行时间
线性回归（LR）	161.3609	46.7845	0.00058
深度神经网络（DNN）	0.3550	0.2671	0.02107
决策树（DT）	0.2416	0.1952	0.00194
支持向量机回归（SVR）	0.2760	0.2165	0.00106
梯度提升决策树（GBDT）	0.1907	0.1461	0.01493
轻型梯度提升机（LigthGBM）	0.1784	0.1295	0.00099

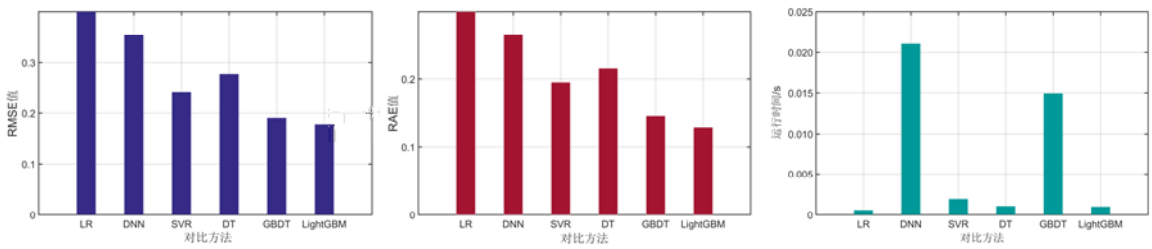
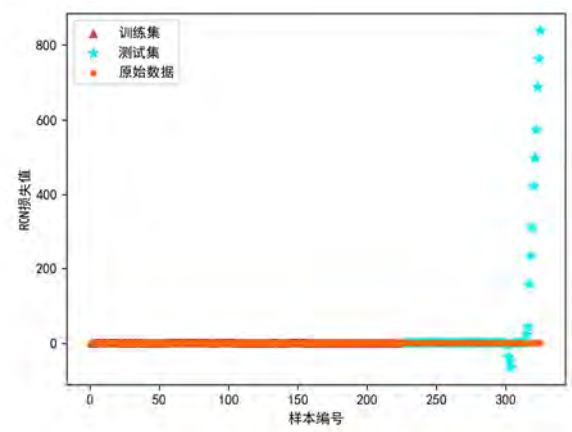
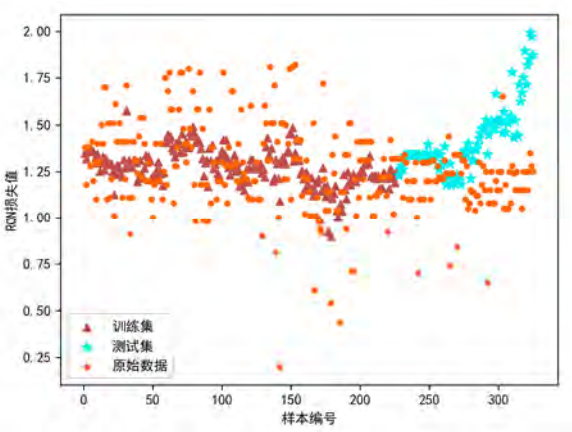


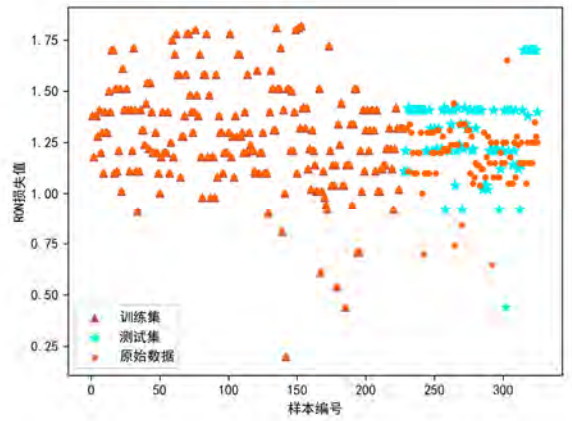
图 7.2 实验结果对比图



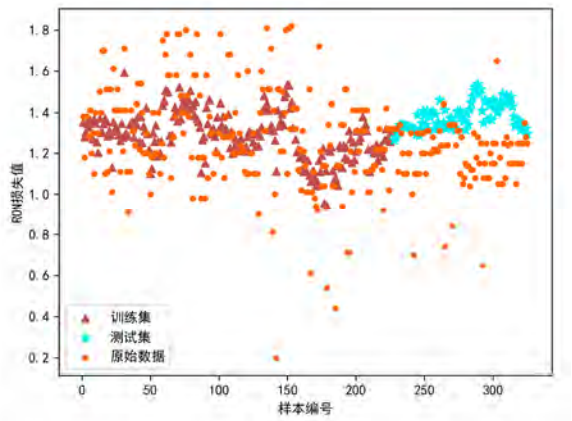
(a) 线性回归（LR）



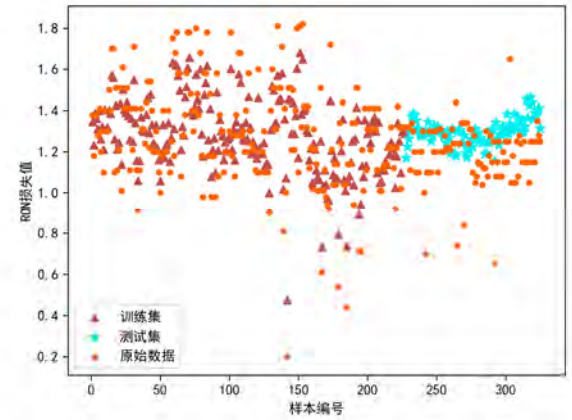
(b) 深度神经网络（DNN）



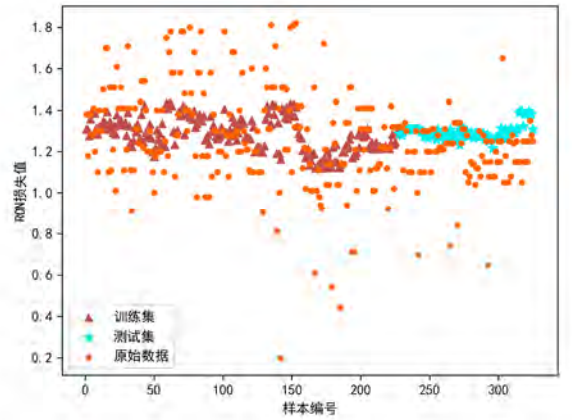
(c) 决策树（DT）



(d) 支持向量机回归（SVR）



(e) 梯度提升决策树（GBDT）



(f) 轻型梯度提升机（LigthGBM）

图 7.3 预测结果对比图

8. 问题四（主要变量操作方案的优化）

8.1 问题分析

问题 4 要求我们在保持优化过程中原料、待生吸附剂、再生吸附剂的性质不变和产品硫含量不大于 5ug/g 的条件下，给出辛烷值损失降幅大于 30%的主要变量优化操作方案。该问题有两个难点，其一为附件数据中硫含量与其他变量的关系不明确，其二为这是一个带复杂约束条件的最优化问题，用一般算法求解时，很容易陷入局部最优，为此，我们采用如下步骤来解决这些问题。

1）使用准确性高，训练速度较快的 LightGBM 算法对硫含量进行估计，筛选出影响产品硫含量的主要变量，以明确表达在优化过程中硫含量的限制条件。

2）使用粒子群优化算法智能避免局部最优。以产品辛烷值为目标函数，同时设定各类约束条件，得到辛烷值损失优化模型。由于解空间并非线性结构，存在很多峰谷型的局部最优值，采用粒子群优化算法，通过模拟群体智能避免局部最优，对参数进行简单的调整，就可以快速地得到全局最优解。

3）计算样本主要变量优化后的辛烷值降幅，给出辛烷值（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。

8.2 产品硫含量估计

a）参数筛选

根据问题二的分析与求解，同样地对估算产品硫含量的主要变量筛选后再建立数量关系，首先计算产品硫含量与其他各个变量的皮尔森系数，结果如下图

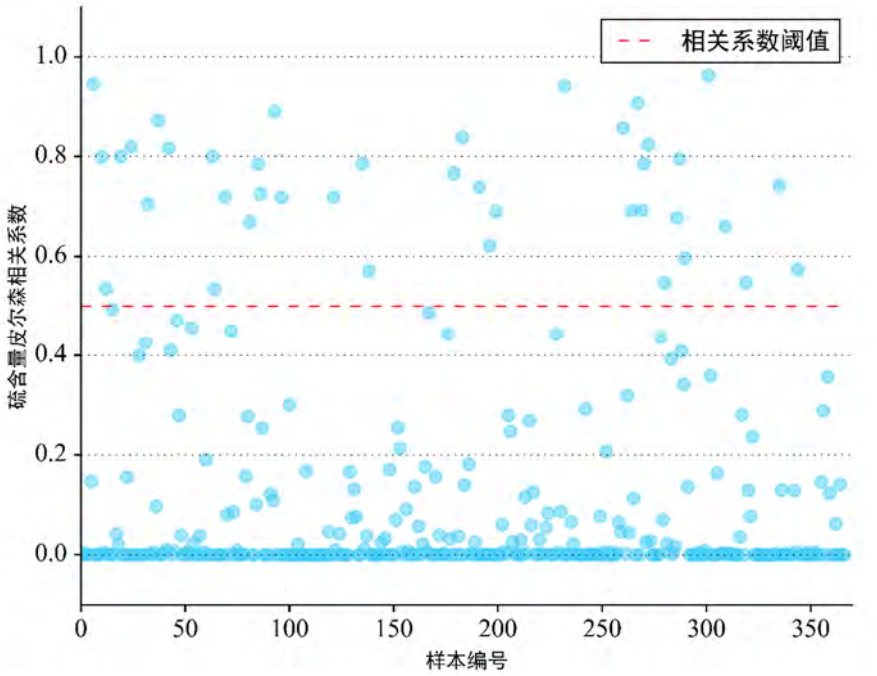


图 8.1 硫含量与其他各个变量的皮尔森系数

综合上图和文献[6]，来看脱附空速、吸附空速、吸附温度、脱附温度这些工艺条件极大影响着汽油精制的脱硫过程，并且原料油的硫含量和烯烃含量限定硫含量的范围。然后使用遗传算法进一步筛选的主要变量如表 8.1，本题将使用以下主要变量估算产品硫含量：

表 8.1 主要相关变量

位号	变量中文名称	位号	变量中文名称
原料性质	辛烷值 RON	PDT_2503.DACA	D-107 底排放滑阀压差
再生吸附剂性质	S, wt%	TE_5004.DACA	稳定塔顶出口温度
CAL_H2.PV	氢油比	TE_5003.DACA	C-201#37 层塔盘温度
TE_2005.PV	反应器底部温度	SIS_PT_2602.PV	再生器顶部/再生器接收器差压
TC_5005.PV	稳定塔下部温度	SIS_TE_2605.PV	再生器下部温度
TE_5102.PV	干气出装置温度	TE_6002.DACA	烟气出辐射室温度
FT_9101.PV	污油出装置	SIS_TE_6009.PV	预热器入口烟气温度
TE_9301.PV	1.0MPa 蒸汽进装置温度	TE_6008.DACA	空气预热器空气出口温度
SIS_TE_6010.PV	加热炉排烟出口温度	LC_2601.DACA	R-102 床层吸附剂料位密度
TE_1608.PV	加热炉循环氢出口温度	TE_2604.DACA	R-102 #1 通风挡板温度
FC_1202.PV	D121 顶去放火炬流量	TE_2004.DACA	R-101 床层下部温度
AI_2903.PV	再生烟气氧含量	PDT_1004.DACA	ME-104 出入口
FT_1501.TOTAL	新氢进装置流量	FT_5204.DACA.PV	汽油产品去气分流量
TE_1104.DACA	E-101F 管程出口管温度	FT_1503.DACA.PV	氢气至循环氢压缩机入口
LI_9102.DACA	D-204 液位	FT_1504.DACA.PV	氢气至反吹氢压缩机出口

b）Light GBM 估算硫含量

在问题 3 的求解分析中，我们已经验证了 Light GBM 算法准确性高，更快训练效率的优点，而硫含量估计同样也是回归问题。因此，我们使用 Light GBM 算法建立硫含量估计模型。本文将处理后的数据集按 7：3 的比例划分为训练集和验证集。皮尔森系数和遗传算法的参数筛选加速模型训练的速度，而根据制备工艺和化学性质遴选的参数保证了模型的正确和精准。最终，模型在训练集和验证集上得到的结果如图所示，训练集上的 MAE 误差为 0.63，测试集上的误差为 0.85。误差较小，平均 0.74，说明估计模型有较好的精度。

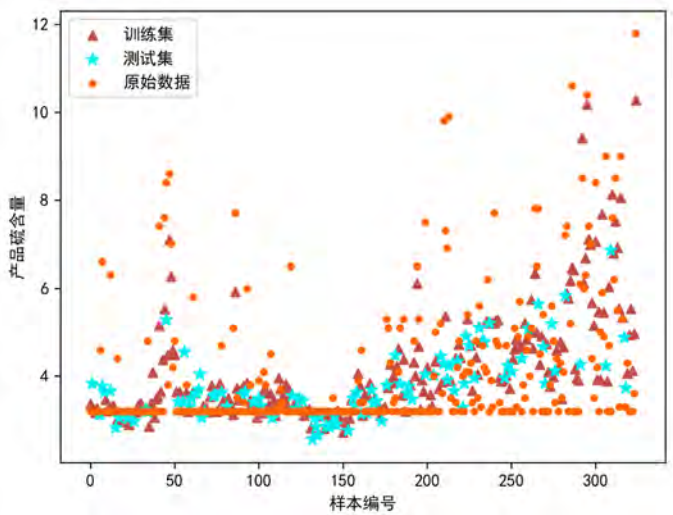


图 8.2 LightGBM 估算硫含量结果

8.3 模型建立与模型求解

8.3.1 辛烷值损失优化模型建立

要优化主要变量以降低辛烷值损失，首先需要公式化各个限制条件和目标函数，问题

4 的数学规划形式为：

$$\begin{cases} \min f(x) \\ x = [c_1, c_2, \dots c_n] \\ s.t.\ \ g(x) < 5 \\ h(x) > 0.3 \end{cases} \tag{8.1}$$

式中x是操作变量的可行域，函数$g(x)$表示 x对应的产品硫含量，$h(x)$表示辛烷值损失降幅，该式表示在$g(x)$小于 5，$h(x)$大于 0.3 的限制条件下，尽可能使得辛烷值损失函数$f(x)$的取值最小化。由于自变量多达 30 维，且限制条件复杂，为能更快速准确地寻优。本部分根据粒子优化算法建立 RON 损失的优化模型，用于复杂空间的最优解的搜索，其模型流程图如下。

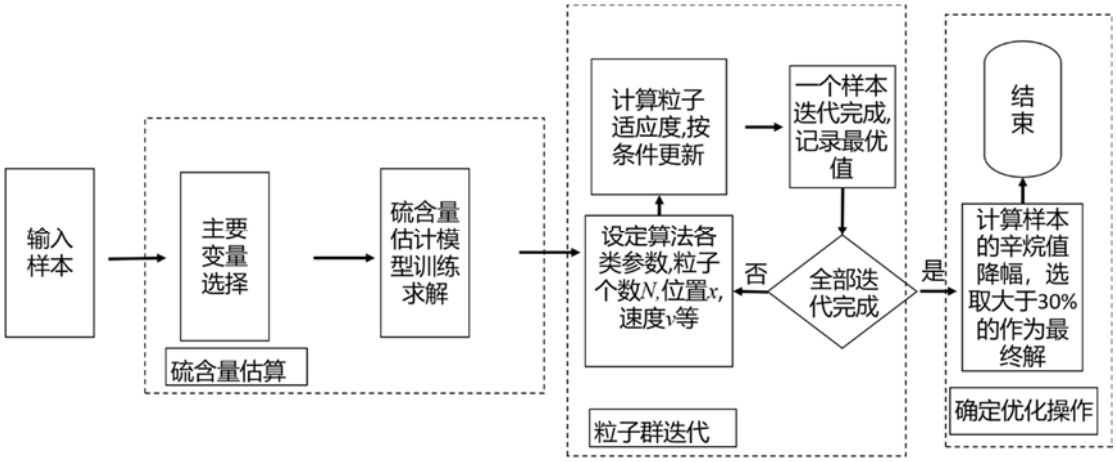


图 8.3　辛烷值损失优化模型

其中 x 和速度向量 v 共同表征了问题四的原料限制条件，硫含量的限制设定在更新粒子个体条件中，粒子群算法迭代具体过程如下：

1）初始化粒子群数 N

$$x_i = (c_1, c_2, \dots, c_{30}) \tag{8.2}$$

= 100，粒子维度就是优化的操作变量个数，D = 30，最大迭代次数为 T = 200，学习因子 $l_1 = l_2$ =1.2，惯性权重为 w = 0.8。

本题设定 100 个粒子组成一个群落，其中第 i 个粒子的位置向量 x 为一个 30 维的向量：式中 c_1 到 c_3 为挑选出来的主要变量中保持不变的操作条件，c_4 到 c_{30} 为可以改变的优化操作条件，则最大位置和最小位置分别为：

$$\begin{cases} x_{max} = (c_1, c_2, c_3, 1, \dots 1) \\ x_{min} = (c_1, c_2, c_3, 0, \dots 0) \end{cases} \tag{8.3}$$

第 i 个粒子的飞行速度也是一个 30 维的向量：

$$vi = (v_1, v_2, v_3, \dots v_{30}) \tag{8.4}$$

相应的，v_1 到 v_3 为保持不变的原料待生吸附剂、再生吸附剂的性质的飞行速度，由于保持不变其取值为零，v_4 到 v_{30} 为优化操作条件的变化速度，考虑到各变量的调整限度，则最大速度和最小的速度为：

$$\begin{cases} Vmax = (0, 0, 0, 1, \dots, 1) \\ Vmin = (0, 0, 0, -1, \dots, -1) \end{cases} \tag{8.5}$$

2）初始化种群粒子位置 x 和速度 v，计算每个粒子的适应度：

$$fit = f(x) \tag{8.6}$$

式中 $f(x)$即为问题三中辛烷值损失预测模型的取值，优化目标就是最小化辛烷值损失。计

算适应度后得到粒子个体最优位置 p 和最优值 b_{best}，以及粒子群全局最优位置 g 和最优值 g_{best}。

3）更新位置 x 和速度值 v，并进行边界处理，使其不超出最大值和最小值，避免粒子落在解空间之外，然后对每个粒子,用它的适应度值fit和个体极值 Pbest(i)比较。如果fit <Pbest(i),以及硫含量 $s(i)$<5,则替换粒子个体最优位置 p 和最优值 Pbest、粒子群全局最优位置 g 和最优值 g_{best}。

4）判断是否满足终止条件：若满足，则结束算法并输出优化结果;否则继续迭代

最后计算各个样本优化后的辛烷值降幅：

$$h(x) = (y - f(x))/y \tag{8.7}$$

8.3.2 辛烷值损失优化模型求解

本部分根据给出的样本数据，共采用 325 个样本，由于变量取值相差较大，部分变量达到了 10000，因此我们使用公式（归一化）对数据进行归一化处理，则各变量的最大值和最小值均为 0 和 1，最后通过 Python 编程求解，粒子群算法各项具体参数如下表：

表 8.2 粒子群算法参数

种群规模	惯性权重	加速常数 c_1	加速常数 c_2	粒子最大速度	粒子最大速度	位置最大值	位置最小值	最大迭代
100	0.8	1.5	1.5	1	-1	1	0	200

在优化过程中，粒子刚开始无规律地分布在解空间中，然后逐渐找到附近的局部最优点，接着大多数粒子朝着全局最优点演化，最后大部分粒子都驻守在全局最优点，如下图。

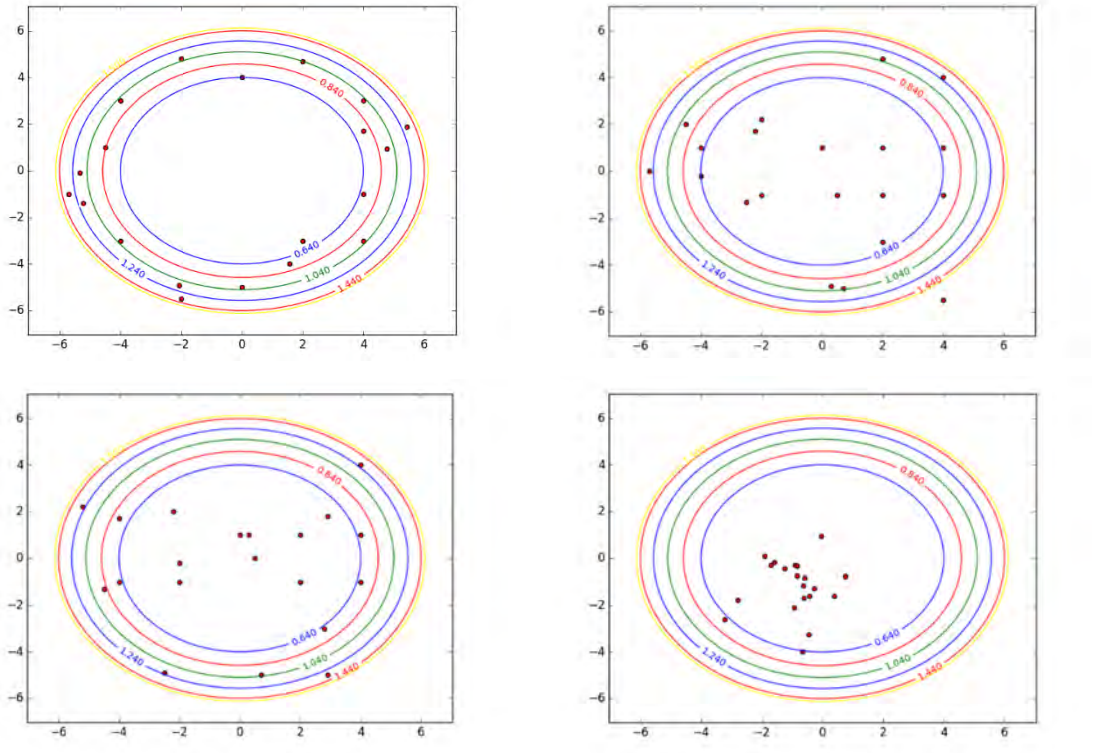


图 8.3 粒子群演化过程

在计算过程中，样本 1 与 57 的适应度进化曲线如下图，可以看出模型收敛的速度较快，避免了陷入局部最优，效率较高。

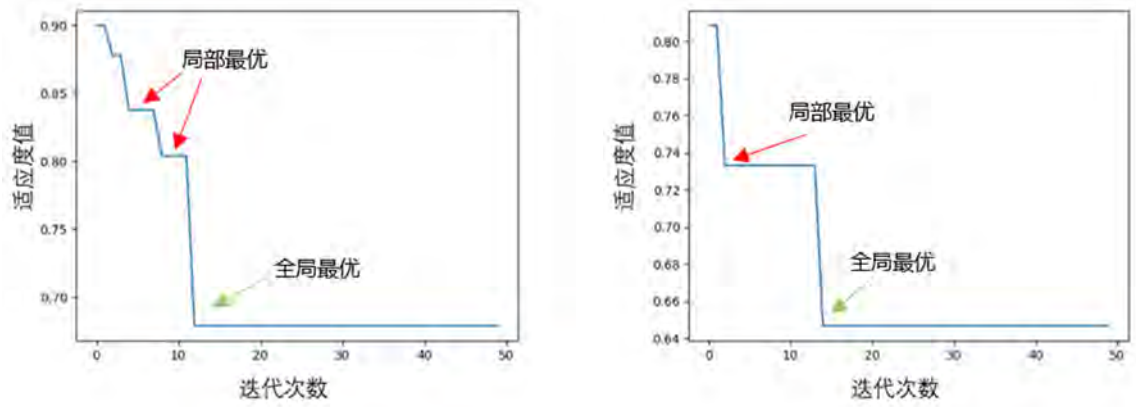


图 8.4 适应度进化曲线

在演化过程中，所有样本收敛时的迭代次数见表 8.3，平均收敛次数为 14 次，并未超过预设参数，说明模型的参数设置是合理的，模型是比较准确快速的。

表 8.3 各个样本收敛时迭代次数

样本编号	收敛时迭代次数
1	12
2	14
…	…
325	13

最终所有样本优化结束后得到的结果如图 8.5，产品硫含量都控制在 5ug/g 以下，辛烷值的损失降幅平均达到了 41%，说明模型的求解性能较好，优化辛烷值损失的能力较强。

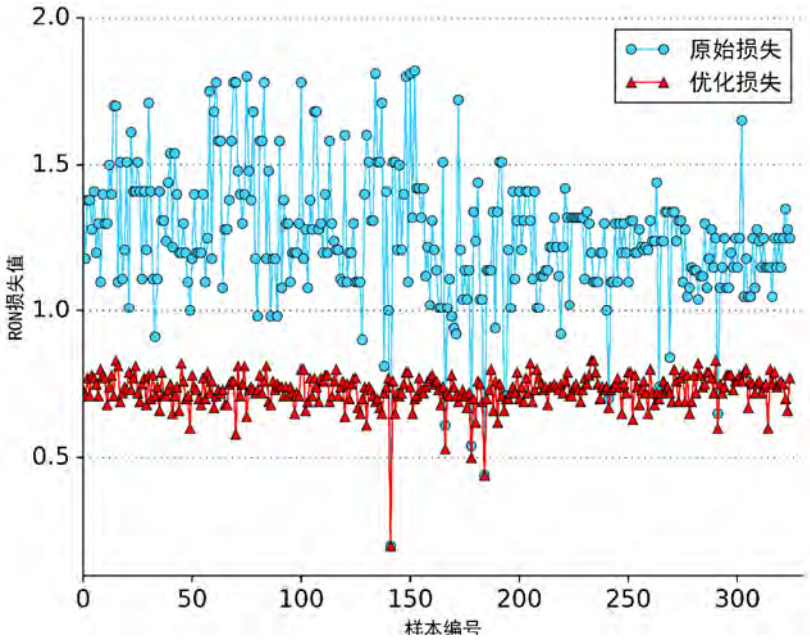


图 8.5 样本优化前后的辛烷值对比

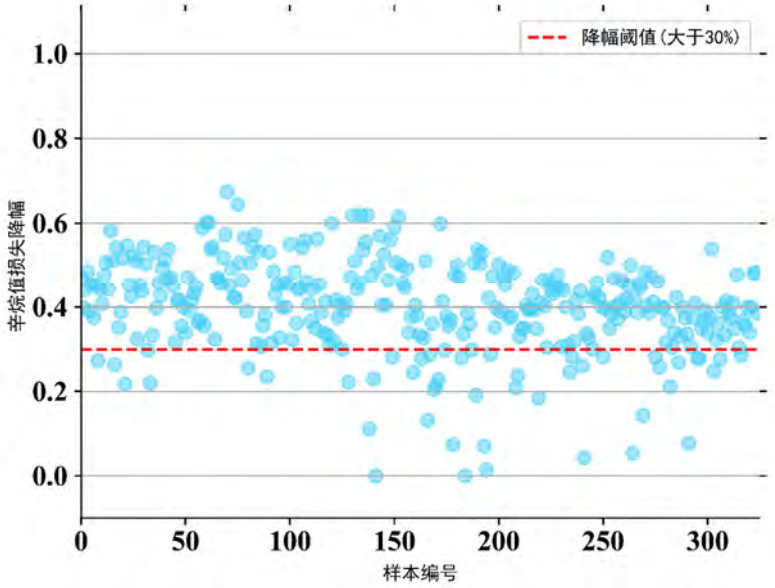


图 8.6　辛烷值损失降幅结果

部分样本优化后的操作条件具体见表 8.4，所有样本对应的主要变量优化后的操作条件见附件 5。

表 8.4　主要变量优化后的操作条件

样本编号	优化后的操作条件
1	0.39，0.83，0.86，0.47，0.12，0.28，0.88，0.00，…，0.74，0.06，0.71，0.54
2	0.33，0.81，0.81，0.45，0.12，0.28，0.94，0.05，…，0.60，0.06，0.08，0.76
…	… …
325	0.62，0.72，0.17，0.96，0.34，0.28，0.61，0.66，…，0.65，0.39，0.28，0.36

9. 问题五（模型可视化展示）

9.1 问题分析

问题 5 是模型的可视化展示，将 133 号样本以图形展示模型主要操作变量优化调整过程中汽油对应辛烷值和硫含量的变化轨迹。

此问难点在于展示多个变量调整与辛烷值和硫含量的变化轨迹，操作变量较多；从外，由于工业装置为了平稳生产，优化后的操作主要变量只能逐步调整到位。这些因素造成模型可视化难度较大。

为此，我们将操作变量取值的变化梯度化，将梯度作为横坐标画出最优化调整过程中对应的汽油辛烷值和硫含量的变化轨迹，从原始操作条件描绘到优化后的条件为止。以便将无法表达的 30 个变量维度通过梯度变化清晰地展示出来，从而观察逐步调整的过程。

9.2 变化轨迹可视化

梯度计算公式见 9.1，式中 x 为原始的操作条件，x_s 为优化后的操作条件，我们将其差值的十分之一作为梯度，逐步调整主要操作变量，以满足每次允许的调整幅度。观察 133 号样本在此过程中汽油辛烷值和硫含量的变化轨迹，首先其适应度的空间分布如图 9.1

$$d = (x - x_s)/10 \tag{9.1}$$

汽油辛烷值和硫含量的变化如图 9.2 与图 9.3，硫含量的值不断升高，限制着汽油辛烷值的进一步优化。

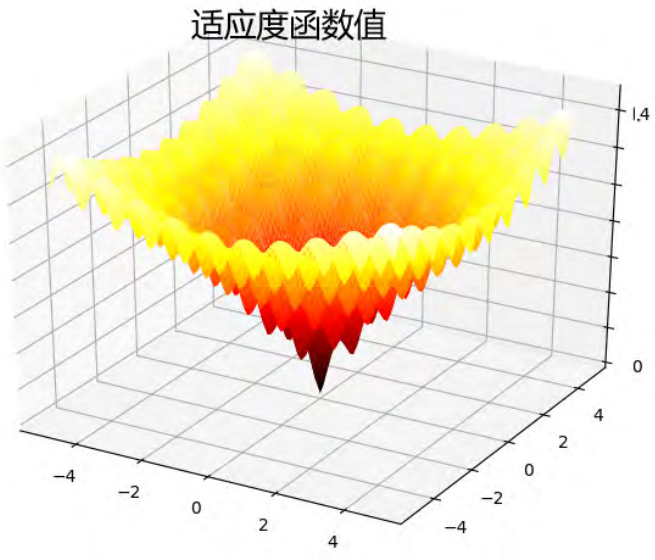


图 9.1 133 号样本优化过程中适应度函数的空间分布

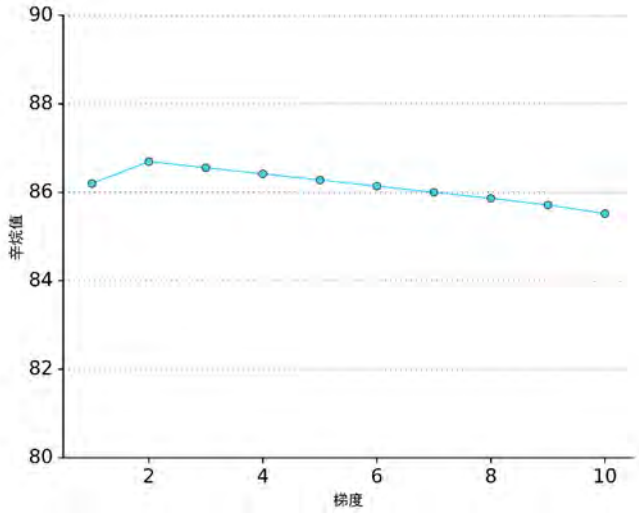


图 9.2　辛烷值随梯度变化

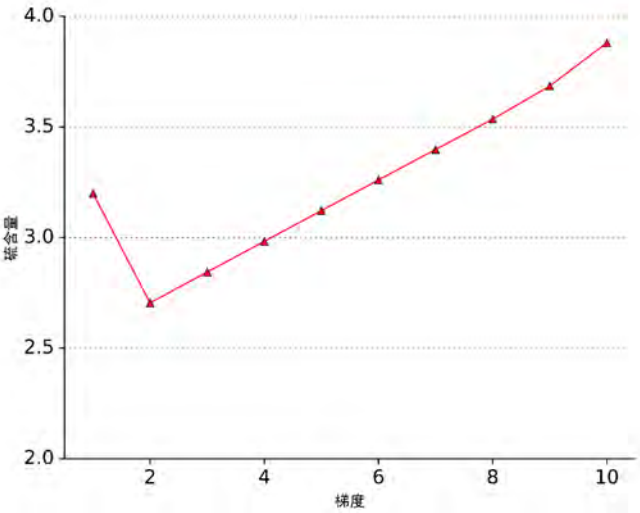


图 9.3　硫含量随梯度变化

10. 问题总结

10.1 问题 1 总结

问题 1 中，我们对 285 号和 312 号样本的原始数据进行了数据分析，科学合理地使用方法进行了数据处理。我们使用拉伊达准则和最大最小限幅剔除异常数据,设定数据缺失比例阈值，并使用平均值法补充缺失值，使用平均值进行样本数据确定。最终实现 285 号和 312 号样本数据的确定并添加到附件一中。

10.2 问题 2 总结

问题 2 中，通过相关资料，我们定义了无关变量、相关变量和冗余变量。由于样本数据中变量过多，直接进行主要变量筛选操作，会导致变量筛选过程时间长以及选择的变量不能有效满足要求。为了快速有效地筛选出主要变量，我们采用二级逐次降维的方法。首先，通过皮尔森相关系数筛选出 57 个候选变量，实现第一次变量降维；通过皮尔森相关系数，我们排除大量无关变量。接着，为了达到题设要求，我们采用适应度函数为余弦相似度的遗传算法排除强耦连的变量（冗余变量）实现第二次变量降维。最后，我们筛选出 30 个建模主要变量。

10.3 问题 3 总结

问题 3 需要建立辛烷值（RON）损失与主要变量之间的数量关系，然后利用合适的度量指标对建立好的数学模型进行验证。并解决该问题的两个主要难点在于：选择何种机器学习算法，如何验证该算法对解决本问题的优越性。

首先，我们结合本题的样本数据特征，通过综合分析各个模型的优劣势，初步选取了轻型梯度提升机（LightGBM）算法来构建辛烷值损失预测模型；

其次，我们建立并求解了辛烷值损失预测模型；

最后，我们通过对比实验分析验证法，对六种方法的预测结果进行了分析，并进一步验证了问题分析中所提出的线性预测模型和深度神经网络预测模型不适用于本题的观点，也验证了基于 LightGBM 模型的 RON 损失预测方法的有效性。

因此，我们的方法较好地对问题 3 进行了模型建立与求解。

10.4 问题 4 总结

问题 4 中，求解需要先表征硫含量与主要变量的数量关系，以公式化硫含量限制条件求解复杂约束优化问题，粒子群算法具有较强地全局寻优能力，而且通过简单的参数调整可以快速求解全局最优解，模型结果比较准确，模型表现良好。

10.5 问题 5 总结

问题 5 中，梯度化调整过程，有利于对样本辛烷值和硫含量指标的观察，并且在实际操作过程中易于实现或是自动化梯度调优过程，这是一个合理的操作形式。

参考文献

[1] 刘永才, 李佳. (2019). 影响 S Zorb 装置汽油辛烷值损失因素分析. 石油化工设计, (4), 6.

[2] 李辉, 姚智. (2013). S-Zorb 装置生产京 V 汽油实践. 炼油技术与工程, (2), 11-14.

[3] 王军强, 阚宝训, 蒋红斌. (2015). S Zorb 装置生产国 V 汽油的实践. 炼油技术与工程, (4), 1-4.

[4] 周欢, 齐万松, 李宏勋. (2019). S Zorb 装置辛烷值损失大原因的分析与措施. 云南化工, (9), 36.

[5] 史朋飞. (2015). 汽油辛烷值提高与成品汽油优化控制探讨 (Doctoral dissertation).

[6] 赵乐平, 周勇, 段为宇, 庞宏, 李扬, 刘继华. (2004). OCT-M FCC 汽油选择性加氢脱硫技术的开发和工业应用. 工业催化, 12(1), 16.

[7] Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In Noise reduction in speech processing (pp. 1-4). Springer, Berlin, Heidelberg.

[8] 王铁方, 刘晓洁, 李涛, 龚勋, 蒋亚平, 杨进, 胡晓勤. (2006). 基于家族基因的网格信任模型. 四川大學學報 (工程科學版), 38(6), 123-126.

[9] Galton, F. (1886). Regression towards mediocrity in hereditary stature. The Journal of the Anthropological Institute of Great Britain and Ireland, 15, 246-263.

[10] Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016, October). DNN-based prediction model for spatiotemporal data. In Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (pp. 1-4).

[11] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Advances in neural information processing systems (pp. 3146-3154).

[12] Sun, X., Liu, M., & Sima, Z. (2020). A novel cryptocurrency price trend forecasting model based on LightGBM. Finance Research Letters, 32, 101084.

[13] Guo, Q., Zhu, Z., Pei, H., Xu, F., Lu, Q., Zhang, D., & Wu, W. (2019, December). Mobile user credit prediction based on Lightgbm. In 2019 International Conference on Big Data, Electronics and Communication Engineering (BDECE 2019) (pp. 140-144). Atlantis Press.

[14] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.

[15] Helmbold, D. P., & Schapire, R. E. (1997). Predicting nearly as well as the best pruning of a decision tree. Machine Learning, 27(1), 51-68.

[16] Yang, H., Chan, L., & King, I. (2002, August). Support vector machine regression for volatile stock market prediction. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 391-396). Springer, Berlin, Heidelberg.

[17] Ye, J., Chow, J. H., Chen, J., & Zheng, Z. (2009, November). Stochastic gradient boosted distributed decision trees. In Proceedings of the 18th ACM conference on Information and knowledge management (pp. 2061-2064).

[18] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

[19] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26.

附录

问题 1 和问题 2 代码

```
import pandas as pd
import scipy
from scipy.stats import pearsonr
import csv
from matplotlib import pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
from matplotlib import rcParams
rcParams['axes.unicode_minus']=False
def plot_lines(y1, y2, figname):
    # 数据准备
    xs = [i for i in range(len(y1))]
    # y_line = [0.5 for i in range(len(pearson))]
    # 准备画布
    fig = plt.figure()
    ax = plt.gca()
    # 设置折线大小
    plt.plot(xs, y1, color="#44cef6", linestyle="-", marker="o", label="原始损失")
    plt.plot(xs, y2, color="r", linestyle="-", marker="^", label="优化损失")
    plt.legend(loc="best")
    plt.xlim((0, 330))
    plt.ylim((0.1, 2))
    plt.xlabel("样本编号")
    plt.ylabel("RON 损失值")
    # 设置图例的字体及大小
    plt.tick_params(labelsize=15, width=1, direction='out', top='off', bottom='on', left='on', right='off')
    labels = ax.get_xticklabels() + ax.get_yticklabels()
    [label.set_fontname('黑体') for label in labels]
    ax.spines['top'].set_color('none')
    ax.spines['right'].set_color('none')
    plt.grid(axis="y")
    # 设置横纵坐标的名称以及对应字体格式
    font_format = {'family': 'Times New Roman', 'size': 18}
    # plt.xlabel(pearson, font_format)
    plt.savefig(figname, dpi=1000)
    plt.show()
def load_file(file):
    data = pd.read_excel(file)
    # print(list(data.columns))
    # print(len(list(data.columns)))
    old = data.iloc[:, [1]].values
```

```
    old = [i[0] for i in old]
    # print(old)
    new = data.iloc[:, [2]].values
    new = [i[0] for i in new]
    plot_lines(old, new, "RON 优化")
load_file("RON 最优化.xlsx")
```

问题 3 代码

```
import numpy as np
import matplotlib.pyplot as plt
import lightgbm as lgb
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

import warnings
# filter warnings
warnings.filterwarnings('ignore')
#  正常显示中文
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
#  正常显示符号
from matplotlib import rcParams
rcParams['axes.unicode_minus']=False
from sklearn.datasets import load_iris
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import time
import csv
import re
from openpyxl import load_workbook    #  读取 xlsx 类型的文件需要专门的读取程序

def normalization(data):    # normalization 和 normalization1 是对所有数据进行归一化，不太合理。而
normalization2 可以对列进行归一化
    _range = np.max(data) - np.min(data)
    return (data - np.min(data)) / _range

def standardization(data):
    mu = np.mean(data, axis=0)
    sigma = np.std(data, axis=0)
    return (data - mu) / sigma
```

```
# 如果归一化后的范围是[-1, 1]的话，可以将 normalization()函数改为：
def normalization1(data):
    _range = np.max(abs(data))
    return data / _range

def normalization2(data):
    minVals = data.min(0)
    maxVals = data.max(0)
    ranges = maxVals - minVals
    m = data.shape[0]
    normData = data - np.tile(minVals, (m, 1))
    normData = normData/np.tile(ranges, (m, 1))
    return normData, ranges, minVals

def draw(x_train_label, x_test_label, x_label, train_y, test_y, original_y, picture_path):
    plt.xlabel('样本编号')
    plt.ylabel('RON 损失值')
    # plt.xlim(xmax=9, xmin=0)
    # plt.ylim(ymax=9, ymin=0)
    # 画两条（0-9）的坐标轴并设置轴标签 x，y
    colors1 = '#C0504D'   # 点的颜色
    colors2 = '#00EEEE'   #'#2894FF'
    colors3 = '#FF6600'
    area1 = np.pi * 2 ** 2   # 点面积
    area2 = np.pi * 3 ** 2   # 点面积
    area3 = np.pi * 4 ** 2   # 点面积
    # 画散点图
    plt.scatter(x_train_label, train_y, marker='^', s=area2, c=colors1, alpha=1, label='训练集')
    plt.scatter(x_test_label, test_y, marker='*', s=area3, c=colors2, alpha=1, label='测试集')
    plt.scatter(x_label, original_y, marker='o', s=area1, c=colors3, alpha=1, label='原始数据')
    # plt.plot([0, 9.5], [9.5, 0], linewidth='0.5', color='#000000')
    plt.legend()
    plt.savefig(picture_path, dpi=300)
    plt.show()

def p_words(string):
    string_list = re.findall(r"\d+\.\d+", string)
    return string_list[0]



plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# matplotlib 画图中中文显示会有问题，需要这两行设置默认字体
```

```
all_data = []
# f_csv = load_workbook('325 个样本数据 666.xlsx')
with open("fina_30_samples.csv", "r", encoding="utf-8") as f:    # 数据已放附录
# with open("325 个样本数据 888.csv", "r", encoding="utf-8") as f:
    #   遇到编码问题，可以考虑把 xlsx 文件转化为 utf-8 格式的 csv 文件
    f_csv = csv.reader(f)
    for row in f_csv:
        row = [float(p_words(item)) for item in row]
        # 在进行正则匹配时，一定要先把 csv 里面存储的数据转化为数值格式，并设置小数点
        all_data.append(row)
all_data = np.array(all_data)
data_label = all_data[:, 0]
target = all_data[:, -1]
data, _, _ = normalization2(all_data[:, 1:-1])   # normalization 和 normalization1 是对所有数据进行归一化，不太合理。而 normalization2 可以对列进行归一化

print(data[0])
print(len(data[0]))
print(target[0])
train_size = int(len(data)*0.7)
x_train = data[:train_size]
x_test = data[train_size:]
y_train = target[:train_size]
y_test = target[train_size:]
x_train_label = data_label[:train_size]
x_test_label = data_label[train_size:]
# x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3, random_state=20)
# x_train_label, x_test_label = train_test_split(data_label, test_size=0.3, random_state=20)
x_label = data_label

models = [LinearRegression(normalize=True), MLPRegressor(alpha=0.01), DecisionTreeRegressor(), SVR(),GradientBoostingRegressor(), lgb.LGBMRegressor(objective='regression', num_leaves=31, learning_rate=0.05, n_estimators=20)]
models_str = ['LinearRegression', 'MLPRegressor', 'DecisionTree', 'SVR', 'GBDT', 'lightGBM']

for name, model in zip(models_str, models):
    print('开始训练模型：'+name)
    model = model     # 建立模型
    model.fit(x_train, y_train)
    startTime = time.time()
    y_train_pred = model.predict(x_train)
    y_test_pred = model.predict(x_test)
    stopTime = time.time()
```

```
    save_path = ".\\\\画图数据\\" + name + ".tif"
    draw(x_train_label, x_test_label, x_label, y_train_pred, y_test_pred, target, save_path)

    Run_Time = stopTime - startTime
    MSE = mean_squared_error(y_test, y_test_pred) ** 0.5
    MAE = mean_absolute_error(y_test, y_test_pred)
    R2 = r2_score(y_test, y_test_pred)
    print('The rmse of prediction is:', MSE)
    print('The mae of prediction is:', MAE)
    print('The r2 of prediction is:', R2)
    print('The Run_Time of prediction is:', Run_Time)
```

问题 4 和问题 5 代码

```
#数据处理与硫含量估计网络
import pandas as pd
from sklearn.model_selection import train_test_split
def std_data(feature):
    feature = (feature - feature.min()) / (feature.max() - feature.min())
    # mean = feature.min(axis=0)
    # feature -= mean
    # std = feature.std(axis=0)
    # feature /= std
    print(feature)
    return feature
def load_data(path):
    train_data = pd.read_excel(path, sheet_name='train')
    test_data = pd.read_excel(path, sheet_name='Test')
    i = train_data.columns[1]
    print(i)
    test_index = len(train_data)
    feature_target = pd.concat([train_data, test_data], axis=0, ignore_index=True)
    feature_target = pd.get_dummies(feature_target, prefix=['NAME'])
    train_x, train_y = feature_target.iloc[:test_index, 0:-1], feature_target['thermal'][0:test_index]
    test_x, test_y = feature_target.iloc[test_index+1:, 0:-1], feature_target['thermal'][test_index+1:]
    x_size = len(train_x.columns)
    train_x.loc[:,i:'T/K'] = std_data(train_x.loc[:,i:'T/K'])
    test_x.loc[:,i:'T/K'] = std_data(test_x.loc[:,i:'T/K'])
    print('next \n',train_x,train_y,test_x,test_y)
    return train_x,train_y,test_x,test_y,x_size
def split_data(path):
    df = pd.read_excel(path, sheet_name='Sheet1')
    x = df.iloc[:, :-2]
    x = std_data(x)
```

```
    print(x)
    print(df)
    # x = pd.get_dummies(x, prefix=['cate'])
    y = df['μg/g']
    x.to_excel('x.xlsx')

    train_x, test_x, train_y, test_y = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=0)
    # train_data = train_x.join(train_y)
    # test_data = test_x.join(test_y)
    # train_data.to_csv('data/train_data.csv')
    # test_data.to_csv('data/test_data.csv')
    train_x_num = train_x.loc[:, '硫含量, μ g/g':'S-ZORB.PC_1001A.PV']
    # train_x_lab = train_x.iloc[:, 389:]
    # std_data(train_x_num)
    train_x = train_x_num # .join(train_x_lab)
    test_x_num = test_x.loc[:, '硫含量, μ g/g':'S-ZORB.PC_1001A.PV']
    # test_x_lab = test_x.iloc[:, 389:]
    # std_data(test_x_num)
    test_x = test_x_num #.join(test_x_lab)
    data_df_size = len(train_x.columns)
    return train_x, train_y, test_x, test_y, data_df_size
if __name__ == '__main__':
    my_path = r'data/jm_odata.xlsx'
    print(split_data(my_path))
from keras import models, layers
from keras.callbacks import EarlyStopping
from process_data import *
import numpy as np
import matplotlib.pyplot as plt
#  正常显示中文
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
#  正常显示符号
from matplotlib import rcParams
rcParams['axes.unicode_minus']=False
def bulid_model(x_size):
    model = models.Sequential()
    model.add(layers.Dense(64, input_dim=x_size, activation='relu'))
    model.add(layers.Dense(64, input_dim=x_size, activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='adam', loss='mse',metrics=['mae'])
    return model
def draw(x_train_label, x_test_label, x_label, train_y, test_y, original_y, picture_path):
    plt.xlabel('样本编号')
```

```
    plt.ylabel('产品硫含量')
    # plt.xlim(xmax=9, xmin=0)
    # plt.ylim(ymax=9, ymin=0)
    # 画两条（0-9）的坐标轴并设置轴标签 x，y
    # x1 = np.random.normal(2, 1.2, 300)  # 随机产生 300 个平均值为 2，方差为 1.2 的浮点数，即第一簇点的 x 轴坐标
    # y1 = np.random.normal(2, 1.2, 300)  # 随机产生 300 个平均值为 2，方差为 1.2 的浮点数，即第一簇点的 y 轴坐标
    # x2 = np.random.normal(7.5, 1.2, 300)
    # y2 = np.random.normal(7.5, 1.2, 300)
    colors1 = '#C0504D'  # 点的颜色
    colors2 = '#00EEEE'  #'#2894FF'
    colors3 = '#FF6600'
    area1 = np.pi * 2 ** 2  # 点面积
    area2 = np.pi * 3 ** 2  # 点面积
    area3 = np.pi * 4 ** 2  # 点面积
    # 画散点图
    plt.scatter(x_train_label, train_y, marker='^', s=area2, c=colors1, alpha=1, label='训练集')
    plt.scatter(x_test_label, test_y, marker='*', s=area3, c=colors2, alpha=1, label='测试集')
    plt.scatter(x_label, original_y, marker='o', s=area1, c=colors3, alpha=1, label='原始数据')
    # plt.plot([0, 9.5], [9.5, 0], linewidth='0.5', color='#000000')
    plt.legend()
    plt.savefig(picture_path, dpi=300)
    plt.show()
if __name__ == '__main__':
    my_path = r'data/jm_odata.xlsx'
    df = pd.read_excel(my_path, sheet_name='Sheet1')
    # x = pd.get_dummies(x, prefix=['cate'])
    y = df['μg/g']
    my_x, my_y, my_test_x, my_test_y, my_data_size = split_data(my_path)
    # my_model = bulid_model(my_data_size)
    # early_stopping = EarlyStopping(patience=10)
    # my_model.fit(my_x, my_y, epochs=1000, batch_size=4, validation_data=[my_test_x, my_test_y],
    #                      callbacks=[early_stopping])
    # my_model.save('modelS.h5')
    my_model = models.load_model('modelS.h5')
    loss = my_model.evaluate(my_test_x, my_test_y)
    print(loss)
    re = my_model.predict(my_test_x)
    draw(my_y.index,  my_test_y.index, y.index, my_model.predict(my_x), my_model.predict(my_test_x), y, '1.png')
    re = re.reshape((1, len(re)))
    print(re)
    re_df = pd.DataFrame({'re': re[0],'rt':list(my_test_y)})
```

```
        re_df.to_csv('data/re.csv')
        np.savetxt('reslut.txt', re, delimiter=",")
#PSO 算法求解
import numpy as np
import matplotlib.pyplot as plt
from keras.models import load_model
import pandas as pd
# 正常显示中文
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
# 正常显示符号
from matplotlib import rcParams
rcParams['axes.unicode_minus']=False
modelS = load_model('modelS.h5')
modelRON = load_model('modelRON.h5')
data = pd.read_excel('std_odata.xlsx')
data_max = np.array(data[-2:-1])[0]
data_min = np.array(data[-1:])[0]
def fit_fun(x,y):
        x = np.array([x])
        v = modelRON.predict(x)+0.6
        return v
def limit_fun(x):
        x = np.array([x])
        v = modelS.predict(np.array(x)) + 0.8
        print(v)
        return v
def PSO(sample):
        N = 20
        D = 365
        T = 50
        c1,c2 =1.5,1.5
        Wmax,Wmin = 0.8,0.4
        yuanliao = sample[0:11]
        caozuo_max = data_max[11:365]
        caozuo_min = data_min[11:365]
        Xmax = np.append(yuanliao, caozuo_max)
        Xmin = np.append(yuanliao, caozuo_min)
        Vmax = np.append([0]*11, [1]*354)
        Vmin = np.append([0]*11, [-1]*354)
        x = np.random.random(size=[N, D])*(Xmax-Xmin)+Xmin
        v = np.random.random(size=[N, D])*(Vmax-Vmin)+Vmin
        p = x
        p_best = np.ones(N)
```

```
    y = sample[-2]
    for i in range(N):
        p_best[i] = fit_fun(x[i][:],y)
    g = np.ones(D)
    g_best = float('inf')
    for i in range(N):
        if p_best[i] < g_best and limit_fun(x[i][:]) < 5:
            g = p[i][:]
            g_best = p_best[i]
    gb = np.ones(T)
    for i in range(T):
        for j in range(N):
            if fit_fun(x[j][:],y) < p_best[j] and limit_fun(x[j][:]) < 5:
                p[j][:] = x[j][:]
                p_best[j] = fit_fun(x[j][:],y)
            if p_best[j] < g_best:
                g = p[j][:]
                g_best = p_best[j]
            w = Wmax - (Wmax-Wmin)*i/T
            v[j][:] = w*v[j][:] + c1*np.random.random()*(p[j][:]-x[j][:]) + c2*np.random.random()*(g-
x[j][:])
            x[j][:] = x[j][:] + v[j][:]
            for ii in range(D):
                if v[j][ii] > Vmax[ii] or v[j][ii] < Vmin[ii]:
                    v[j][ii] = np.random.random()*(Vmax[ii]-Vmin[ii]) + Vmin[ii]
                if Xmin[ii] > x[j][ii] or x[j][ii] > Xmax[ii]:
                    x[j][ii] = np.random.random()*(Xmax[ii]-Xmin[ii]) + Xmin[ii]
        gb[i] = g_best
    loss = 1 - gb[-1] / y
    # print(g,'gb',gb)
    print(y, loss)
    plt.plot(gb)
    plt.xlabel('样本编号')
    plt.ylabel('适应度值')
    plt.show()
    return g, gb,y,loss
g_list = []
gb_list = []
y_list = []
for index in range(325):
    my_sample = np.array(data.iloc[index:index+1, :367])[0]
    r1,r2,ry,rl = PSO(my_sample)
    y_list.append([ry,r2[-1],rl])
    g_list.append(r1)
```

```
        gb_list.append(r2)
        # if index ==3:
        #         break
np.savetxt('ng',g_list)
np.savetxt('ngb',gb_list)
np.savetxt('ny',y_list)
#画图程序
import csv
from matplotlib import pyplot as plt
# 正常显示中文
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
# 正常显示符号
from matplotlib import rcParams
rcParams['axes.unicode_minus']=False
data = pd.read_excel('新建 Microsoft Excel 工作表.xlsx')
result = list(data['%'])
def plot_scatter(pearson, figname):
        # 数据准备
        xs = [i for i in range(len(pearson))]
        y_line = [0.3 for i in range(len(pearson))]
        # 准备画布
        fig = plt.figure()
        ax = plt.gca()
        # 设置折线大小
        plt.scatter(xs, pearson, s=50, color="#44cef6", marker="o", alpha=0.5)
        plt.plot(xs, y_line, color="r", linestyle="--", label="降幅阈值(大于 30%)")
        plt.legend(loc="best")
        plt.xlim((0, 325))
        plt.ylim((-0.1, 1.1))
        plt.xlabel("样本编号")
        plt.ylabel("辛烷值损失降幅")
        # 设置图例的字体及大小
        plt.tick_params(labelsize=15, width=1, direction='out', top='off', bottom='on', left='on', right='off')  # 刻
度值尺寸
        labels = ax.get_xticklabels() + ax.get_yticklabels()
        [label.set_fontname('Times New Roman') for label in labels]
        ax.spines['top'].set_color('none')
        ax.spines['right'].set_color('none')
        plt.grid(axis="y")
        # 设置横纵坐标的名称以及对应字体格式
        font_format = {'family': 'Times New Roman', 'size': 18}
        # plt.xlabel(pearson, font_format)
        plt.savefig(figname, dpi=1000)
```

```
        plt.show()
plot_scatter(result,'4jieguo.png')
```