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	安徽财经大学
参赛队号	**20103780003**
队员姓名	**1.桂扬**
	2.程理政
	3.章玉玲

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 基于支持向量机降低 S-Zorb 装置辛烷值损失优化方法

摘　　要：

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。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的前提下，尽量降低汽油辛烷损失值。

对于问题一和问题二，我们将利用个案剔除、均值填补、高相关滤波、低方差滤波等方法对原始数据进行建模前的预处理。

对于问题三，我们将建立辛烷值（RON 损失预测模型）主要有以下四个步骤：一、确定数据的整定，样本筛选逻辑方法，包括个案剔除法，均值插补法，最大最小值限幅及根据拉依达准则（3σ）准则去除异常值，得到纯净的数据。二、寻找建模主要变量，做降维处理。首先做 IV 值筛选，剔除预测能力较差的变量。再用高相关滤波法，设置低方差阈值等方法多次降维。三、用最终筛选的 30 个以下的变量建立回归模型。分别建立数据挖掘中的常见模型，如线性回归，回归决策树、回归神经网络，最近邻域（KNN）和支持向量机（SVM）等。四、模型检验。将样本分成训练集和测试集，并通过改进的 Bootstrap 来决定模型最终的准确度，以此作为选定最终模型的标准。

对于问题四，本问题可归结为优化问题。即使用问题三中最终保留的支持向量回归机模型作为预测模型，在该预测模型的基础上带入 RON 损失降幅高于 30%的样本的原料数据，使用差分进化算法找出对应原料的最优操作变量。在与真实产品 RON 值做对比可以发现，本文使用差分进化算法对于各类不同的样本能将损失值基本控制 0.9 以下，相较于历史数据的 1.37 个单位具有很大的改善。

针对问题五，探索在给定 133 号样本的原料数据的数据时，参考附件四中的Δ值，将其设置为探索步长，找出各变量数据对应模型的最优解，并列出几个主要影响变量的可视化优化过程。

本文综合运用数理统计、支持向量回归机、差分进化算法等数学方法，利用 R 语言与 python，结合多种模型对所提出问题进行了研究，具有很好的实用性与推广性。 最后，总结了模型的优点与不足，为后续研究如何减少辛

烷值的损失提供了一个新的思路。

关键词：汽油精制;辛烷值损失;优化操作；IV 值；支持向量回归机；差分进化算法

一、问题重述.....4
1.1 问题背景.....4
1.2 需要解决的问题.....5
二、问题分析.....5
2.1 问题一分析.....6
2.2 问题二分析.....7
2.3 问题三分析.....7
2.4 问题四分析.....7
2.5 问题五分析.....7
三、模型的假定与约定.....7
四、符号说明及名词定义.....8
五、模型的建立与求解.....8
5.1 问题一求解.....8
5.2 问题二求解.....9
5.3 问题三求解.....12
5.4 问题四求解.....19
5.5 问题五求解.....22
六、模型评价.....25
七、参考文献.....25
八、附录.....26

一、问题重述

1.1 问题背景

汽油燃烧造成的汽车尾气排放对世界各地生态都造成了显著的消极影响，各国出台了日益严格的车用汽油标准，对硫含量、烯烃含量、辛烷值等提出了新的要求。

目前，我国原油大部分依赖于中东地区的含硫和高硫原油，成品汽油的主要调和成分超过 70%来源于催化裂化（FCC）汽油。FCC 汽油具有硫含量高和烯烃含量高的特点，其中，烯烃是 FCC 汽油中的高辛烷值组分，同时性质较为活泼，在对 FCC 汽油进行加氢脱硫的同时必然伴随烯烃加氢饱和，烯烃加氢饱和变成烷烃后会导致辛烷值（RON）下降。而辛烷值是反映汽油燃烧性能的最重要指标，因此，可以通过优化催化汽油加氢装置的操作条件来减少烯烃加氢饱和、降低辛烷值的损失。[1]

S-Zorb 是美国康菲石油公司（Conoco Phillips）开发的一项生成无硫汽油的工艺技术，主要用于催化汽油的脱硫。它能以较低的辛烷值损耗生产低硫汽油。S-Zorb 技术基于吸附作用原理对汽油进行脱硫，通过吸附剂选择性地吸附汽油中硫醇、二硫化物、硫醚和噻吩类等含硫化合物的硫原子而达到脱硫的目的,然后对吸附剂再生,使其变为二氧化硫进入再生烟气中,烟气再去硫磺或碱洗。在 S-Zorb 过程中有六步主要的化学反应:(1)硫的吸附(2)烯烃加氢(3) 烯烃加氢异构化(4)吸附剂氧化(5)吸附剂还原(6)尾气中和。前三个反应在反应器中进行，第四个反应在再生器内进行，第五个反应在还原器内进行。[2] S-Zorb 工艺流程示意图如图 1 所示：

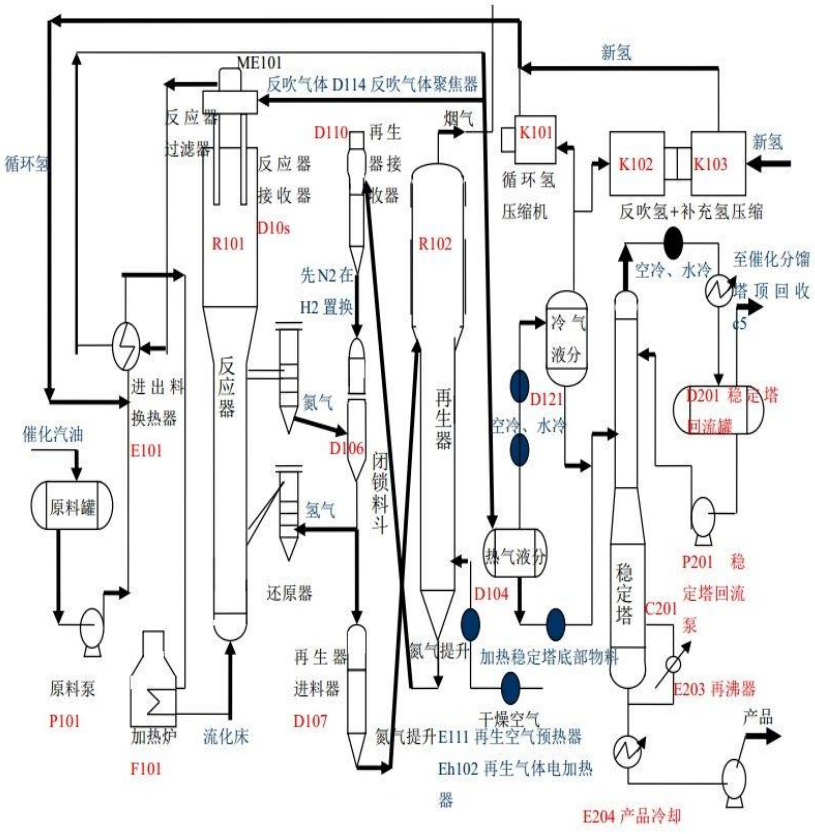

图 1 S-Zorb 工艺流程示意图

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，操作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。因此，降低汽油精制装置辛烷值损失成为提高企业经济效益、缓解环境压力的重中之重，对于应对 2023 年即将实施的国 VI-B 车用汽油标准问题具有一定的指导意义。[3]

1.2 需要解决的问题

1.确定数据的整定，样本筛选逻辑方法。

原始数据中，大部分变量数据正常，但每套装置的数据均有部分位点存在问题，存在大量缺失值和异常值，因此对原始数据进行处理后才可以使用。

2.寻找建模主要变量

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）。因此，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具有代表性、独立性的主要变量（30 个以下），并要求详细说明建模主要变量的筛选过程及其合理性。

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。

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。

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，对 133 号样本，要求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。

二、问题分析

文章整体思路如图 2 所示：

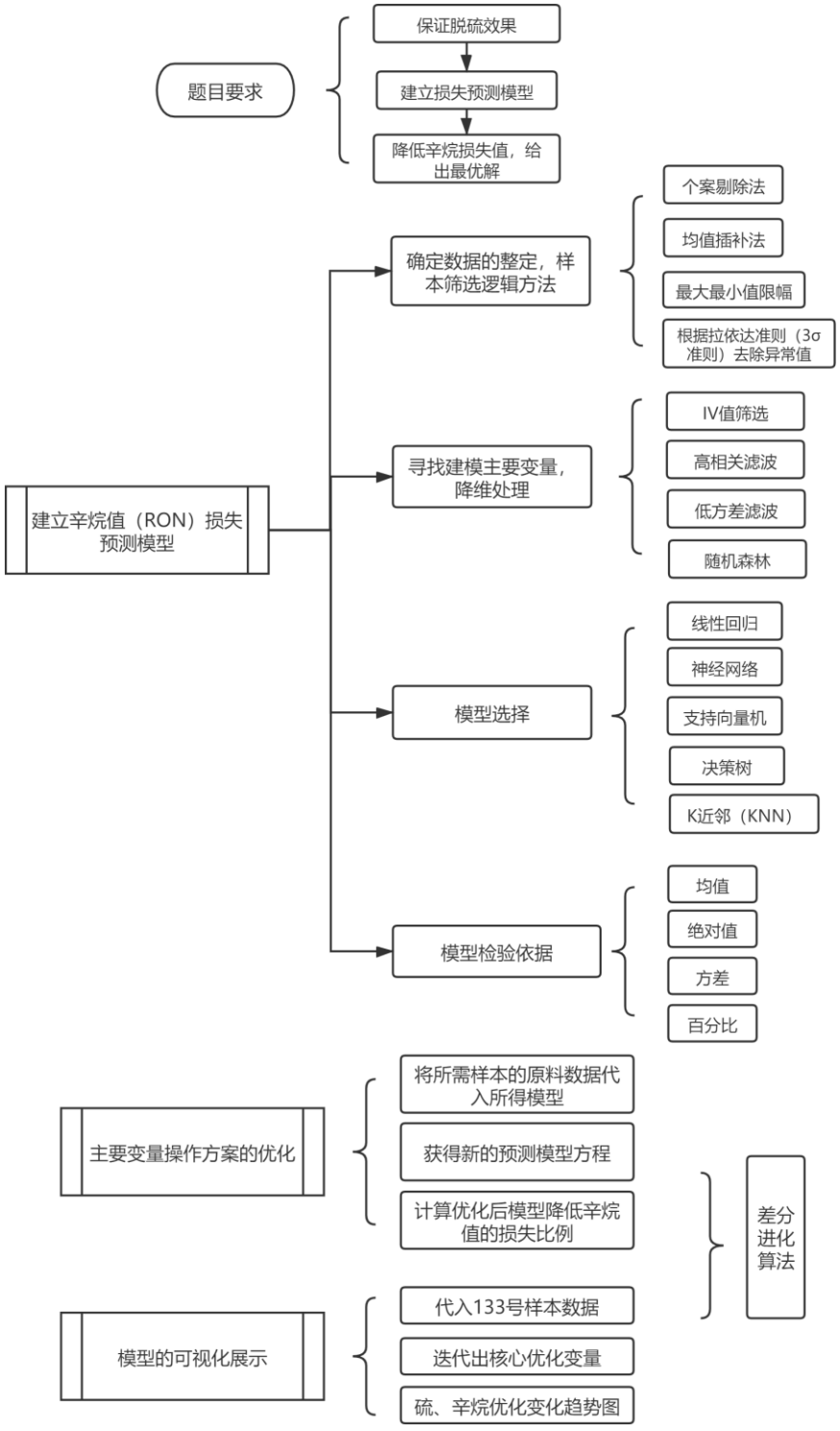

图 2 论文行文思路图

2.1 问题一分析

对于原题所给的数据，存在着缺失值和异常值。首先对 285 号和 313 号数据样本进行预处理，依据个案剔除法、均值填补法、根据拉依达准则（3σ 准则）去除异常值，并将处理后的数据分别加入到附件一中相应的样本号中。

2.2 问题二分析

问题二要求在题中所给的 367 个变量中分析影响模型的主要变量，对数据进行降维处理。常见的降维方法有主成分分析，但是主成分分析一般适用于线性数据降维，由于变量（控制变量）之间具有高度非线性和相互强耦联的关系，同时考虑到主成分分析主要通过将变量重新整合产生新的变量，新的主成分并不是由实际系统产生的，因此在进行 PCA 变换后会丧失数据的解释性。而我们希望能够直接筛选变量，用剩余变量解释模型，因此放弃常用的主成分分析，选择更加直观的 IV 值、高相关滤波、低方差滤波等方法删除影响较弱的变量。首先筛选出相关性低且方差高的部分变量作为固定变量，然后改变阈值，放宽条件，扩大变量个数，通过迭代的方式将新筛选出的变量代入模型中，比对在实际应用中的效果，最终筛选出适合的变量。

2.3 问题三分析

本题要求通过数据挖掘技术建立辛烷值（RON）损失预测模型。使用问题二最终筛选的 30 个以下的变量，将预测模型的变量控制在合适的范围内。数据挖掘中的常见模型有线性回归，回归树、回归神经网络，最近邻域（KNN）和支持向量机（SVM）等方式，分别建立上述模型，通过将样本分成训练集和测试集，并用改进的 Bootstrap 来决定模型最终的准确度。将模型依照准确度相互比较之后决定最终使用的模型。

2.4 问题四分析

问题四要求给出辛烷值（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。使用问题三种建立的最终模型，对每一个降幅大于 30%的样本，将样本的原料数据代入问题三的模型中获得新的预测模型方程。通过使用差分进化算法获得每一个方程的最优解，从而获得对应样本原料的最优操作条件，并通过与原有数据中的 RON 损失值做对比，观察该模型相较于原 RON 损失较低的样本是否能进一步优化。

2.5 问题五分析

问题五要求获得 133 号样本优化过程的可视化展示，通过重复问题四中的主要步骤，通过迭代得出在该样本原料的基础上的最优解，并将该过程可视化。

三、模型的假定与约定

1.催化裂化汽油精制过程是连续的。

2.整个生产过程处于稳态，且所有变量测量值均服从正态分布，即过失误差不存在。过失误差来源包括一些非随机事件，例如仪表故障、系统误差、模型误差、管道泄漏或实际工况不符合稳态假设等。

3.在某次观测期内，原料性质、待生吸附剂和再生吸附剂的性质数据保持不变。

4.在下文中，因所有操作变量都处于 S-ZORB 装置中，则把变量名称 S-ZORB.XXX 简化为 XXX，例如：S-ZORB.TE_2104.DACA.PV 简写为 TE_2104.DACA.PV。

5.为了在代码中便利使用，以 x1,x2,x3.....x29 来表示问题二中筛选出的 29 个核心变量，例如：用 x1 表示 S-ZORB.PDT_3002.DACA

四、符号说明及名词定义

表 1 专业名词及符号对照表

序号	*符号*	**符号说明**
1	IV	information value（信息量）
2	WOE	weight of evidence（证据权重）
3	w	权重向量
4	b	偏置项
5	C	惩罚系数
6	Qi	相对误差绝对值
7	ξ 和 $\xi*$	非负松弛变量
8	σ	标准误差

五、模型的建立与求解

5.1 问题一求解

（1）个案剔除法。查询 285 号和 313 号数据样本中的缺失值，计算缺失值比例。设定缺失值比例阈值为 20%。对于只含有部分时间点的位点，如果其残缺数据超过 20%，无法补充，将此类位点删除。

285 号数据样本中删除的位点见表 2：

表 2 变量及其缺失值数

变量名	FT_1501.PV	FT_1002.PV	FC_1202.PV
缺失个数	40	40	40
FT_1501.TOTAL	FT_5102.PV	FT_2901.DACA	FC_1104.DACA
40	40	40	40
FT_2803.DACA	FT_1502.DACA	TEX_3103A.DACA	FT_5102.DACA.PV
40	40	40	40

313 号数据样本中删除的位点见表 3：

表 3 变量及其缺失值数

变量名	FT_1501.PV	FT_1002.PV	FT_1501.TOTAL
缺失个数	40	40	40
FT_2901.DACA	FC_2432.DACA	FC_1104.DACA	FT_2803.DACA
40	16	40	40
FT_1502.DACA	FC_2432.PIDA.SP	TEX_3103A.DACA	--
40	16	40	--

（2）均值插补。对于残缺数据小于 20%的的位点，空值处用其前后两个小时数据的平均值代替。由筛选结果可知，第 313 号样本需要插补的变量及其缺失个数如表 4 所示：

表 4 变量及其缺失值数

变量名	FT_1204.PV	FT_1204.PV	FT_1204.DACA.PV
缺失个数	2	6	2

（3）题目要求删除附件一 325 个样本中数据全部为空值的位点，虽然数据中不存在全部为空值的位点，但我们依然删除了缺失值比例超过 20%的位点，并且对缺失值比例小于 20%的位点进行均值插补处理。

（4）根据附件一中的历史经验数据，总结出原始数据变量的操作范围，采用最大最小值的限幅方法，剔除 285 号和 313 号数据中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。

3σ 准则：设对被测量变量进行等精度测量，得到 x1，x2，……，xn，算出其算术平均值 x 及剩余误差 vi=xi-x（i=1,2,...,n），并按贝塞尔公式算出标准误差 σ，若某个测量值 xb 的剩余误差 vb（1<=b<=n），满足|vb|=|xb-x|>3σ，则认为 xb 是含有粗大误差值的坏值，应予剔除。贝塞尔公式如下：

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

通过以上方式，我们完成了对 285 号和 313 号数据的预处理，并替换到附件一中的相应位置。

5.2 问题二求解

优化的主要目标是将辛烷值（RON）损失降幅大于 30%。依据积累的大量历史数据，其汽油产品辛烷值损失平均为 1.37 个单位，因此，我们将 0.959 个单位为分类依据，将样本做 0-1 分类，辛烷值损失小于 0.959 记为 1，大于 0.959 记为 0，发现 17 个样本数据损失低于 0.959 个单位。

（1）信息量 IV 值

通过将样本分为两类，进而引入信息量 IV 值（Information Value）来对输入变量进行编码和预测能力评估。特征变量 IV 值的大小即表示该变量预测能力的强弱。因为 IV 值的计算是以 WOE 为基础的。因此首先需要计算 WOE，WOE 的全称是“weight of evidence”，即证据权重。直观上讲，WOE 是对原始变量的一种编码形式，要对一个变量进行 WOE 编码，首先需要把这个变量进行分组处理，即分箱或者离散化，常用离散化的方法有等宽分组，等高分组，或者利用决策树来分组。分组后，对于第 i 组，WOE 的计算公式见下图。

$$WOE_i = \ln\left(\frac{p_{good}}{p_{bad}}\right) = ln\left(\frac{good\%}{bad\%}\right)$$

IV (information value) 衡量的是某一个变量的信息量，公式如下：

$$\mathrm{IV} = \sum_{i=1}^{N}(good\% - bad\%) * WOE_i$$

N 为分组的组数，IV 可用来表示一个变量的预测能力。表 5 是不同 IV 值区间所对应的预测能力：

表 5 IV 值的预测能力分组表

IV	预测能力
<0.03	无预测能力
0.03-0.09	低
0.1-0.29	中
0.3-0.49	高
>=0.5	极高且可疑

我们将不同变量的 IV 值进行排序，删除 IV 值低于 0.03 的变量（即无预测能力的变量），最终剩余变量 347 个。

（2）高相关滤波 (High Correlation Filter)

如果两个变量之间是高度相关的，这意味着它们具有相似的趋势并且可能携带类似的信息。同理，这类变量的存在会降低某些模型的性能（例如线性和逻辑回归模型）。

作为一般准则，我们应该保留那些与目标变量显示高相关性的变量。首先，删除因变量，并将剩余的变量保存在新的数据列中，然后计算独立数值变量之间的相关性。

由图 3 可知，不同色块代表不同变量之间的相关系数，颜色越深，相关程度越高。高相关滤波认为当两列数据变化趋势相似时，它们包含的信息也相似。这样，使用相似列中的一列就可以满足机器学习模型。相关系数大于某个阈值的两列只保留一列。由于相关系数对范围敏感，所以在计算之前也需要对数据进行归一化处理。通过删去相关系数大于 0.5 的变量中 IV 值较低的变量，最终保留变量 79 个。

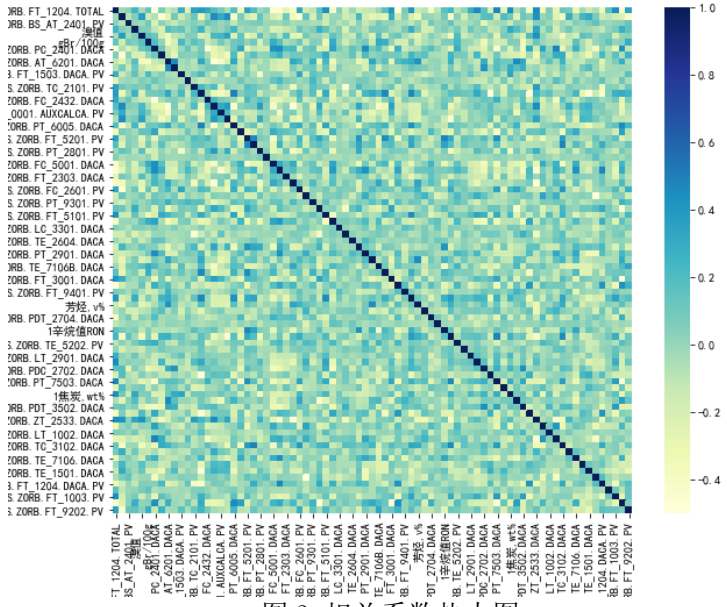

图 3 相关系数热力图

（3）低方差滤波（Low Variance Filter）

如果数据集中某列的数值基本一致，即它的方差非常低，我们通常认为低方差变量携带的信息量也很少，可以把它直接删除。根据附件四，方差受数据的量纲影响，因此在采用该方法之前，我们需要对数据进行归一化处理。首先，计算所有变量数据的方差，然后设置方差阈值，删去方差小于阈值 0.02 的对应变量，保留较高方差的特征变量 46 个。图 4 直观的反映了各个变量的方差：

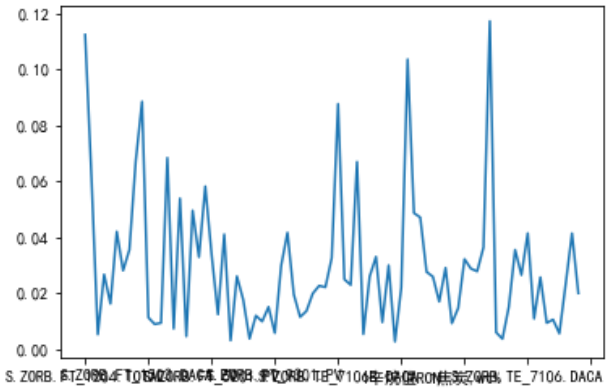

图 4 变量方差示意图

（4）随机森林/组合树 (Random Forests)

组合决策树通常又被称为随机森林，它在进行特征选择与构建有效的分类器时非常有用。一种常用的降维方法是对目标属性产生许多巨大的树，然后根据对每个属性的统计结果找到信息量最大的特征子集。例如，我们能够对一个非常巨大的数据集生成非常层次非常浅的树，每颗树只训练一小部分属性。如果一个属性经常成为最佳分裂属性，那么它很有可能是需要保留的信息特征。对随机森林数据属性的统计评分会向我们揭示与其它属性相比，哪个属性才是预测能力最好的属性。最后通过排序的方式保留评分前 29 的变量作

为数据的最终变量。

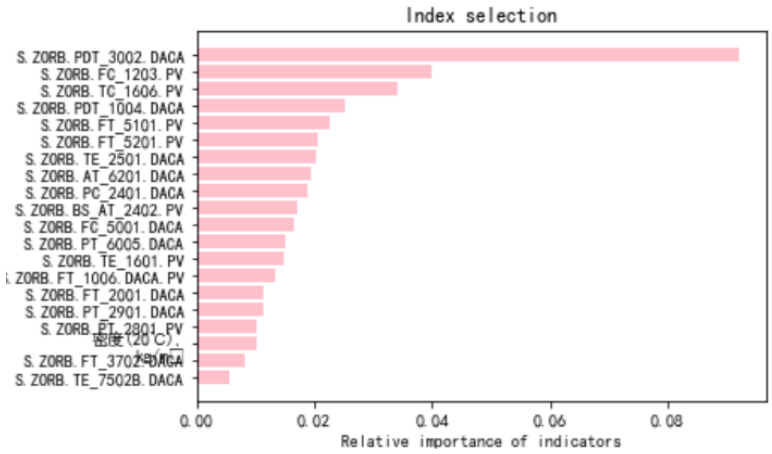

图 5 随机森林数据属性评分柱状图

通过上述步骤筛选，最终保留的变量为表 6 展示的变量：

表 6 保留的变量

S. ZORB. PDT_3002. DACA	S. ZORB. TE_1608. PV	S. ZORB. PDT_3602. DACA	S. ZORB. BS_AT_2402. PV	S. ZORB. TE_5202. PV
S. ZORB. PDT_3502. DACA	S. ZORB. FT_5201. PV	S. ZORB. AT. 0010. DACA. PV	S. ZORB. FC_5001. DACA	S. ZORB. TE_9301. PV
S. ZORB. TC_5005. PV	S. ZORB. TE_2501. DACA	S. ZORB. TC_1606. PV	S. ZORB. PT_6005. DACA	S. ZORB. FT_5101. PV
S. ZORB. LI_9102. DACA	S. ZORB. AT_6201. DACA	S. ZORB. FT_3001. DACA	S. ZORB. TE_1601. PV	S. ZORB. LT_1301. DACA
S. ZORB. SIS_TE_2802	S. ZORB. PC_2401. DACA	S. ZORB. PDT_1004. DACA	S. ZORB. FT_1006. DACA. PV	S. ZORB. LT_2901. DACA
S. ZORB. FC_1203. PV	S. ZORB. FT_1003. PV	原料辛烷值 RON	原料焦炭,wt%	

5.3 问题三求解

考虑到建立建立辛烷值（RON）损失预测模型的复杂性，本文选用的常见的几个数据挖掘模型，采用相互对比的方式选择出最优的建模方案。对于模型错误率的评估，采用 0.632 自助法，最终选择表现最佳的方案作为该题的最终预测模型。

一般而言，Bootstrap 估计的错误率的不确定性会比 K 折交叉验证得到的要小。然而，平均而言，63.2%的样本点在 Bootstrap 中出现过至少一次，因此，其估计的偏差与 2 折交叉验证相似，如果训练集样本量很小，那么这个估计偏差就是一个问题，但是随着训练集样本量增大，问题的严重性也逐渐降低。

为了消除偏差，使用对原始的 Bootstrap 过程进行了改进的 0.632 自助法。0.632 自助法是针对该问题提出的估计方法，它得到的模型效果估计是两种估计量的组合，其中一种是简单 Bootstrap 估计，而另一个是通过重复预测训练集得到的估计(即显性错误率)。

如果将错误率来作为分类模型的特征，那么 0.632 法的结果如下：

$$0.632 \times \text{simple bootstrap estimate} + 0.368 \times \text{apparent error rate}$$

（1）多元回归模型

多元回归分析是指分析被解释变量与若干个解释变量之间的相关关系，其基本思想是虽然自变量和因变量之间没有确定性的函数关系，但可设法找出最能代表它们之间关系的数学表达形式。

模型I基于相关分析与多元线性回归的方法，选择可能影响汽油辛烷值的因素，由于模型中的操作变量（控制变量）之间具有高度非线性和相互强耦联的关系，不能简单使用多元线性回归，通过分别求出其与辛烷值的相关系数，而多项式回归可以处理相当一类非线性问题，因此引入多项式，并通过是否显著判断该因素是否予以考虑，最后利用多元线性回归方法求出各个影响因素与辛烷值的表达式。

在这里，我们对同一组数据分别建立无交互项的多元回归模型，和有交互项的多项式回归模型，并对回归结果的调整后 R 方进行比较，挑选出调整后 R 方最高的模型，进行预测，具体步骤如图 6 所示：

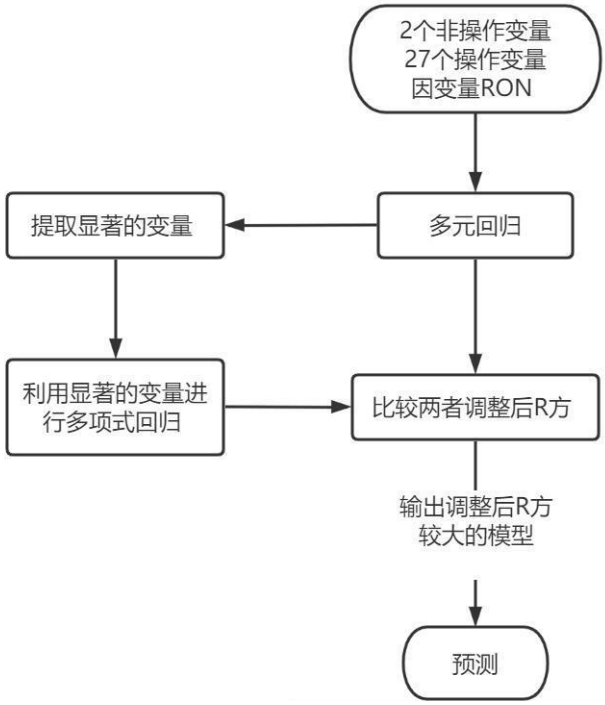

图 6 多元多项式回归过程

现将该样本进行标准化处理，尽量保证样本服从正态分布，然后进行多项式回归，表 8 为我们利用组合的回归模型对测试集进行预测的结果：

表 8 多元回归模型预测结果

预测值	-1.526	0.165	-0.550	1.330	0.560
真实值	-2.190	0.206	-0.403	1.334	0.247

-0.137	-0.943	-1.983	0.723	0.894	—
-0.332	-0.555	-1.469	0.420	0.562	—

我们再绘制残差图，如图 7 所示：

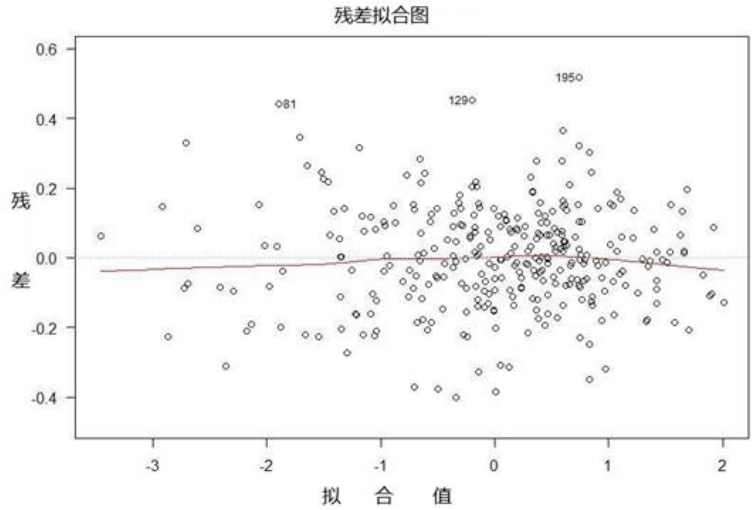

图 7 残差拟合图

上图表现了残差与真实值之间的关系，回归模型的残差在 0 附近呈现均匀分布，这表明残差与估计值无关，说明模型具有有效性，即该模型能很好的解释自变量与因变量之间的关系。

（2）基于 BP 神经网络的回归模型

人工神经网络是模拟人思维的一种方式，通过神经元之间同时相互作用的动态过程来完成信息处理，这是一个非线性动力学系统，其特色在于信息的分布式存储和并行协同处理。

误差反向传播神经网络简称为 BP(Back Propagation)网络，它是一种多层神经网络，每一层都由若干个神经元组成。如图所示为一个 BP 神经网络的结构图，它的左、右各层之间各个神经元实现全连接，即左层的每一个神经元与右层的每个神经元都有连接，而上下各神经元之间无连接。BP 神经网络按有监督学习方式进行训练，当一对学习模式提供给网络后，其神经元的激活值将从输入层经各隐含层向输出层传播，在输出层的各神经元输出对应于输入模式的网络响应。然后，按减少希望输出与实际输出误差的原则，从输出层经各隐含层、最后回到输入层逐层修正各连接权。由于这种修正过程是从输出到输入逐层进行的，所以称它为“误差逆传播算法”。随着这种误差逆传播训练的不断进行，网络对输入模式响应的正确率也将不断提高。

BP 神经网络对非线性数据有良好的处理效果，其突出优点就是具有很强的非线性映射能力和柔性的网络结构。结构如下图 8 所示。

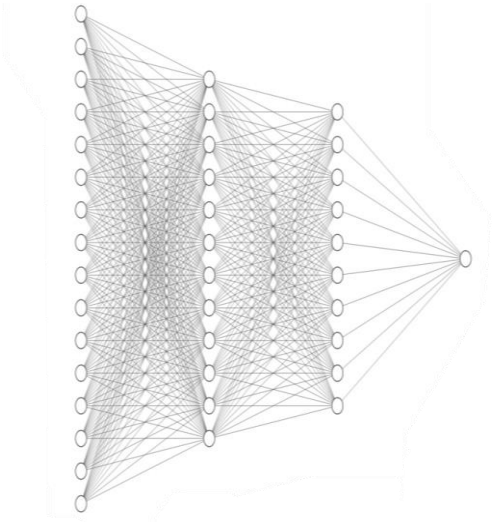

图 8 双隐藏层神经网络结构示意图

在这里，我们的有 29 个输入变量和一个输出变量，所以，我们的输入层神经元有 29 个，输出层神经元有 1 个，参照以往经验，我们创建将 2 个隐藏层，并且在一定的实数范围内进行迭代，为两个隐藏层选取合适的神经元数，使得我们的模型预测效果在一定程度上得到了优化，下表 5 显示了神经网络模型的两个隐藏层在不同神经元数目下的预测准确度：

表 5 不同神经元结构下的预测误差

神经元结构	平均绝对误差	均方误差
(3, 2)	0.518163	0.387106
(6, 2)	0.386137	0.290691
(11, 2)	0.346356	0.164523
(14, 2)	0.220389	0.077481
(8, 5)	0.215518	0.088856
(9, 6)	0.345811	0.170755
(7, 7)	0.283987	0.122296
(11, 7)	0.281094	0.117683
(12, 11)	0.246174	0.091763
(14, 14)	0.239415	0.082379

由上表可知当隐藏层神经元结构为(14,2)和(8,5)时，预测准确率较高，这里我们选择(8,5)的隐藏层结构构建神经网络模型，并设置损失函数为二次代价函数(Quadratic cost)，学习率为 0.005，迭代 500 次，则对测试集进行预测的结果如表 6 所示：

表 6 神经网络模型预测结果

预测值	-1.982	0.254	-0.479	1.536	0.168
真实值	-2.190	0.206	-0.403	1.334	0.247
-0.403	-1.321	-1.573	0.509	0.444	—
-0.332	-0.555	-1.469	0.420	0.562	—

通过观察上表可知，进行参数优化后的神经网络模型能够较好的对 RON 值进行预测建模。

（3）支持向量回归机(LS-SVR)

将支持向量分类的方法推广到解决回归问题，称为支持向量回归。由支持向量分类产生的模型仅依赖训练数据的子集，因为创建模型的代价函数并不考虑超过边界的训练点。类似地，由支持向量回归产生的模型仅依赖训练数据的子集，因为创建模型的代价函数忽略任何接近模型预测的训练数据。

首先利用训练集的 142 份样本通过函数拟合的方式构建模型，以预测集的 325 份样本检验模型的预测能力，根据模型精度和预测能力为评判依据筛选出最优指数。然后采用支持向量回归机（least squares support vector regression，LS-SVR）算法对模型进行优化。

SVR 问题可以理解为：给定输入-输出样本集$\{(x_i,y_i)\}(\mathrm{i}=1,2,\ldots,\mathrm{M})$，其中，是第 i 个 m 维输入向量，是第 i 个标量输出，M 是样本数。通过函数$\mathrm{g(x)}$，将 m 维的输入向量映射到$\mathrm{l}(\mathrm{l}>\mathrm{m})$维的特征空间，SVR 期望能在特征空间中构造一个最优超平面，其中，W 是 l 维的权重向量，b 是偏置项，使得所有样本点到最优超平面的距离都不大于给定的精度$\varepsilon(\varepsilon \geq 0)$。考虑到误差，引入惩罚系数 C(C>0)和非负松弛变量 ξ 和，于是构造最优超平面便转化为求解凸二次优化问题：

$$\begin{cases} \min\ Q(\boldsymbol{W},b,\xi,\xi^*)=\dfrac{1}{2}\|\boldsymbol{W}\|^2+C\sum\limits_{i=1}^{M}(\xi_i+\xi_i^*) \\ \text{s.t.}\ \ y_i-f(\boldsymbol{x}_i)\leqslant\varepsilon+\xi_i \\ \qquad f(\boldsymbol{x}_i)-y_i\leqslant\varepsilon+\xi_i^* \\ \qquad \xi_i\geqslant 0,\xi_i^*\geqslant 0 \end{cases} \tag{1}$$

通过引入非负拉格朗日乘子和，将式(1)的求解转化为其对偶问题的求解：

$$\begin{cases} \max\ Q(\alpha,\alpha^*)=-\sum\limits_{i=1,j=1}^{M}(\alpha_i-\alpha_i^*)\cdot \\ \qquad\qquad (\alpha_j-\alpha_j^*)\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}_i)\boldsymbol{g}(\boldsymbol{x}_j)- \\ \qquad\qquad \varepsilon\sum\limits_{i=1}^{M}(\alpha_i+\alpha_i^*)+\sum\limits_{i=1}^{M}y_i(\alpha_i-\alpha_i^*) \\ \text{s.t.}\ \ \sum\limits_{i=1}^{M}(\alpha_i-\alpha_i^*)=0 \\ \qquad 0\leqslant\alpha_i,\alpha_i^*\leqslant C \end{cases} \tag{2}$$

求解式(2)后，便得到最优超平面即回归函数为：

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} (\alpha_i - \alpha_i^*)\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}_i)\boldsymbol{g}(\boldsymbol{x}) + b \qquad (3)$$

引入满足 Merce 条件的核函数代替，则可以避免对映射函数的求解：

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} (\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}) + b \qquad (4)$$

式(4)即为基于 SVR 的回归函数，系数和以及 b 通过样本训练多次迭代得出。

表 7 SVM 模型预测结果

预测值	-1.72722	0.179465	-0.32959	0.972799	0.394805
真实值	-2.19019	0.20638	-0.40292	1.333582	0.247
-0.29448	-0.76844	-1.46197	0.315643	0.924058	—
-0.33183	-0.55524	-1.46919	0.419635	0.561804	—

通过观察上面的数据，我们可以初步发现，SVM 回归模型能够做出较为精准的预测。

（4）决策树回归模型

决策树是一种树形结构，每个内部节点表示一个判断属性，每个分支代表一个判断结果的输出，最后每个叶节点代表一种分类结果。

决策树是一种十分常用的分类方法，需要监管学习（有教师的 Supervised Learning），监管学习就是给出一堆样本，每个样本都有一组属性和一个分类结果，即分类结果已知，通过学习这些样本建立一棵决策树，利用它能够对新的数据给出正确的分类。同时，我们也可以用类似的原理，利用决策树进行回归建模。

图 9 为根据样本数据得出的决策树模型示意图：

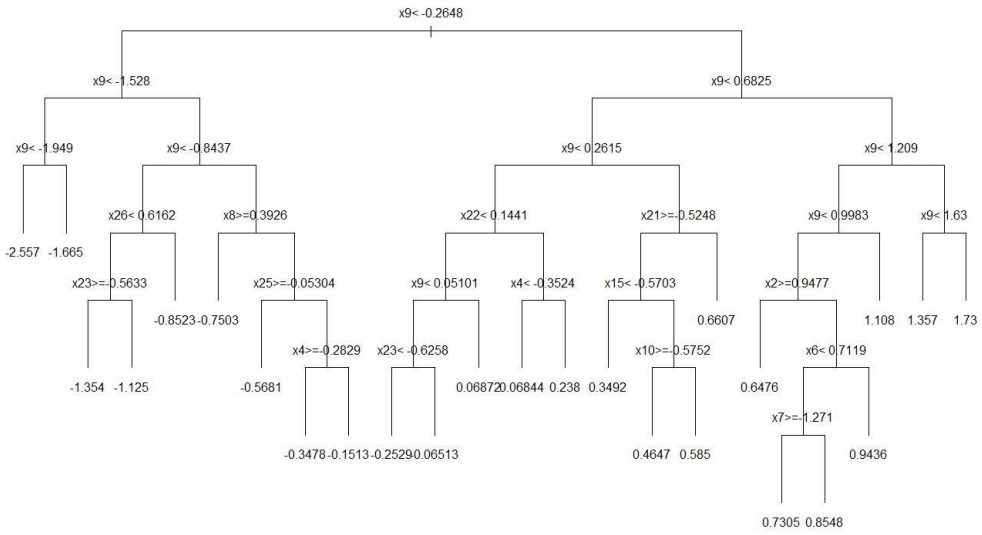

图 9 决策树回归示意图

依托回归树，对测试集进行预测的结果如表 8 所示：

表 8 回归树模型预测结果

预测值	-2.55722	0.068724	-0.56811	1.357476	0.349185
真实值	-2.19019	0.20638	-0.40292	1.333582	0.247
-0.56811	-0.56811	-1.3541	0.237974	0.730533	—
-0.33183	-0.55524	-1.46919	0.419635	0.561804	—

从上表中，我们可以看出，回归树模型的预测准确率并不高。

（5）KNN 回归算法

KNN 回归算法是一种常用的非参数回归方法，较神经网络而言，不需要前期进行参数或结构的学习和寻优，具有思路简单、应用灵活、对异常值不敏感的优点。KNN 算法不仅可以用于分类，还可以用于回归。通过找出一个样本的 k 个最近邻居，将这些邻居的某个（些）属性的平均值赋给该样本，就可以得到该样本对应属性的值。

由于实际运行现场不可避免地存在噪声等因素，且辛烷值（因变量）的测量，数据中可能存在远离训练集中大部分点的点，即离群点。离群点不能反映汽油精制过程中操作变量的正确数值，有可能是存在偏差的点。从预测角度来说，当选取 k 值较小时，离群点不会影响预测精度，当选择 k 值较大时，会造成预测精度降低；从运算效率角度 来看，KNN 回归算法距离度量会遍历训练集全体，所以离群点存在会使运算效率降低， 增加存储成本，因此提出一种改进文献的剪辑离群点流程的训练集优化方法，具体步骤如下：

1）取训练集中的某点$X_i = (x_i^1, x_i^2, \dots, x_i^n, y_i)(i = 1,2, \dots)$

作为测试点，训练集中剩余点组成新训练集对进行 KNN 回归预测，得到输出预测值。

2）对训练集中每一个点遍历步骤 1），得到对应的预测值。

3）求得预测值与实际输出值的相对误差绝对值

$$Q_i = \left| \frac{\hat{y}_i - y_i}{y_i} \times 100\% \right|$$

其中， y_i为实际输出值， $\hat{y}_i(\mathrm{i} = 1,2,\dots)$为预测输出值。

4）设定阈值θ_1，当$Q_i \geq \theta_1$时，从训练集中剔除 $x_i(\mathrm{i} = 1,2,\dots)$ ，剩余点组成训练集 D_T。

利用 KNN 模型，我们对测试集进行预测的结果如表 9 所示：

表 9 KNN 模型预测结果

预测值	-1.2539	0.22669	0.125141	0.821772	0.431821
真实值	-2.19019	0.20638	-0.40292	1.333582	0.247
-0.13076	-0.8538	-0.88223	0.318085	0.990344	—
-0.33183	-0.55524	-1.46919	0.419635	0.561804	—

通过上表，可以初步观察到，在该情境下用 KNN 进行回归预测还是存在部分偏差的，现在我们通过绘制预测值与真实值之间的散点图，来直观的对预测结果进行分析，结果如图 10 所示：

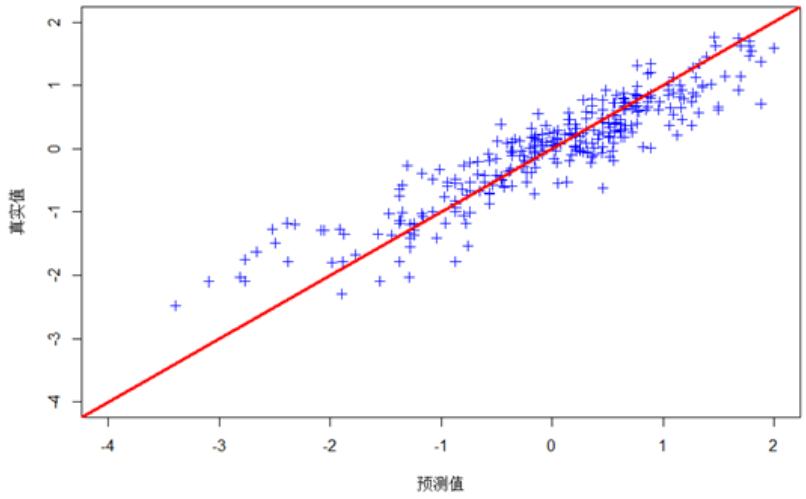

图 10 预测值与真实值的散点图

观察发现，尽管在-1 到 1 这个区间内，预测值接近真实值，但是当真实值向这个区间外偏移时，预测值就会逐步偏离红线，与真实值偏离越来越大。

（6）重抽样检验

表 10 为通过使用 0.632 自助法得出的各模型的误差。分别对 5 个回归模型的预测准确性从 4 个指标(平均误差，平均绝对误差，均方误差，平均绝对百分比误差)进行比较，结果发现 SVM 模型的预测准确率在所有模型中是最优的：

表 10 回归模型的预测准确性检验

	lm	net	svm	lmtree	knn
平均误差	-0.0095	0.01125	0.000215	-0.00431	-0.00458
平均绝对误差	0.162685	0.220674	0.159673	0.183166	0.352572
均方误差	0.051982	0.087092	0.046702	0.063915	0.21913
平均绝对百分比误差	0.734135	0.941953	0.534182	0.982392	1.264679

通过上表可以看出，svm 模型在所有指标上的表现都是最优的，其次就是多元回归模型和回归树模型。

5.4 问题四求解

对于问题四，根据 2.4 节的分析，本问题可归结为优化问题。即运用差分进化算法对问题三中表现最好的预测模型——支持向量回归机，进行最优化求解。

（1）差分进化算法

差分进化算法是一种随机的启发式搜索算法，有较强的鲁棒性和全局寻优能力。它从数学的角度看，是一种随机搜索算法，从工程角度看是一种自适应迭代寻优过程。除了具有较好的收敛性外，差分进化算法非常利于理解和执行，他只包含为数不多的控制参数，并且在整个迭代过程中，这些参数值可以保持不变。

算法的特点：结构简单，容易使用；性能优越，对于大空间、非线性和不可求导的连续问题，其求解的效率比其他进化方法好；自适应性，差分进化算法会根据不同函数进行自动调整，从而提高搜索质量；差分进化算法具有内在的并行性，可协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力；算法通用，差分进化算法操作十分简单，易于编程实现。因此在本题中选用该算法作为最优解的求解算法。

（2）差分进化算法的操作步骤

1) 初始化：将样本导入 R，将 29 个维数为 2 的实数值向量参数作为每一代的种群，每个个体表示：

$$x_{i,G}（i = 1,2,\dots,NP）$$

式中，i 表示个体在种群中的序列，G 表示进化代数：NP 表示种群规模，在最小化工程中 NP 保持不变。在差分进化算法在差分进化算法研究中，一般假定所有限机初始化种群均符合均匀概率分布。设参

$$x_{ji,0} = rand[0,1] * \left(x_j^{(U)} - x_j^{(L)}\right) + x_j^{(L)}$$

$$(i = 1,2,\dots,NP; j = i = 1,2,\dots,D)$$

式中，rand[0,1]表示在[0,1]之间产生的均匀随机数。

2) 变异：对于每个目标向$x_{ji}^{(L)}(i = 1,2,\dots,NP)$，基本差分进化算法的变异向量由下式产生：

$$v_{i,G+1} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G})$$

式中，随机选择的序号 r1、r2 和 r3 互不相同，且 r1、r2 和 r3 与目标向量序号 i 也不同。

3) 交叉：为了增加干扰参数向量的多样性，引入交叉操作，则试验向量变为：

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1})$$

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, 若 rand(j) \le CR 或 j = rnbr(i) \\ x_{ji,G+1}, 若 rand(j) > CR 且 j \ne rnbr(i) \end{cases}$$

$$(\mathrm{i}=1,2,\ldots,\mathrm{NP};\mathrm{j}=\ \mathrm{i}=1,2,\ldots,\mathrm{D})$$

式中，randb(j)表示产生[0,1]之间随机数发生器的第j个估计值；rnbr(i) $\in (1,2,..,D)$表示一个随机选择的序列，用它来确保$u_{ji,G+1}$至少从$v_{ji,G+1}$获得一个参数；CR 表示交叉算子.

4) 选择：为决定试验向量$u_{i,G+1}$是否会成为下一代中的成员，按照贪装准则将试验向量与当前种群中的目标向量$x_{i,G}$进行比较。如果目标函数要被最小化，那么具有较小目标函数值的向量将在下一代种群中出现。下一代中的所有个体都比当前种群的对应个体更佳或者至少一样好。

边界条件的处理：根据附件 4 提供的变量范围设置边界约束，将即将超出边界约束的个体设置为临界边界值。

算法流程如图 11 所示：

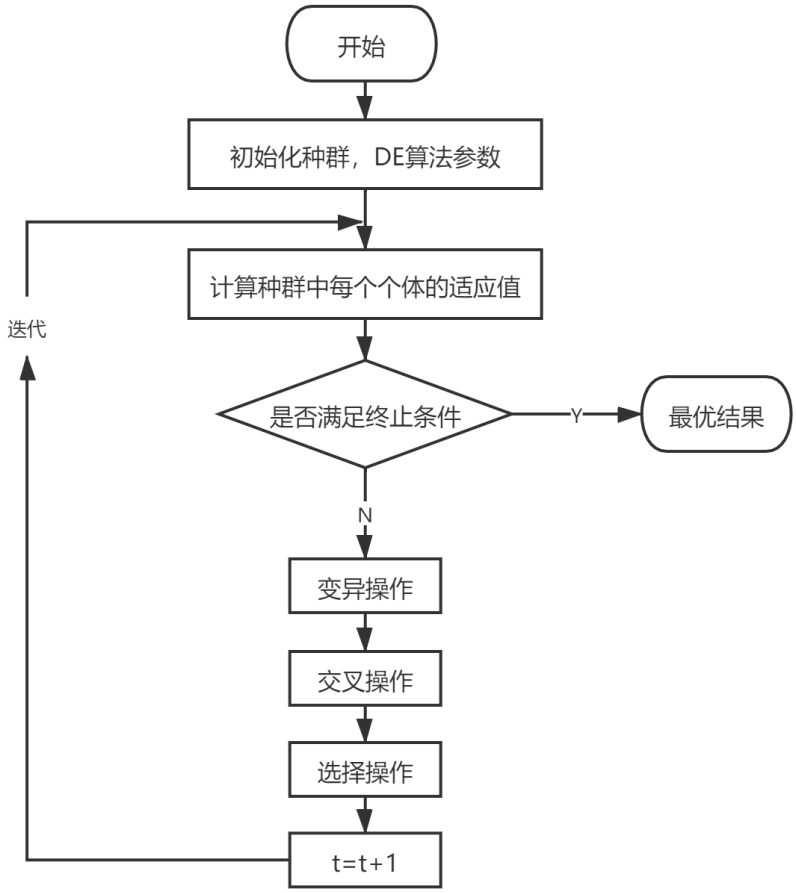

图 11 差分进化算法图示

算法求解结果与原始样本产品的 RON 的损失值对比如表 11 所示：

表 11 RON 损失值对比表

样本	产品真实 RON 损失值	最优解 RON 损失值	硫含量（μg/g）

1	0.91	0.87919	4.776179
2	0.9	0.90728	4.311224
3	0.81	0.886682	4.849325
4	0.2	0.946485	4.29459
5	0.61	0.889005	4.719536
6	0.94	0.906927	4.992575
7	0.92	0.892261	4.591199
8	0.54	0.91306	4.505323
9	0.44	0.916346	4.75605
10	0.94	0.914536	4.483968
11	0.71	0.884991	4.054165
12	0.71	0.912651	4.997577
13	0.92	0.886127	4.85293
14	0.7	0.908555	3.979212
15	0.74	0.902565	4.668876
16	0.84	0.892911	4.707273
17	0.65	0.887898	4.871724

在与真实产品 RON 值做对比可以发现，本文使用差分进化算法对于各类不同的样本能将损失值基本控制 0.9 以下，相较于历史数据的 1.37 个单位具有很大的改善。优化过的模型样本中硫含量均低于 5μg/g。差分进化算法本质是一种随机搜索算法，通常需要进行上千次的迭代过程，但是所需要时间较长。由于时间有限我们无法进行如此大量的迭代，因此上表数据仅能代表本次迭代的结果，最终最优解的 RON 可能回小于上表数据。

5.5 问题五求解

（1）求最优解

首先，我们将 133 号样本的原材料 RON 和焦炭含量代入问题 4 中的差分进化算法，求得最优解，其中，我们设置种群数量为 30，变量维数为 29，最大进化代数为 600，变异算子为 0.5，交叉算子为 0.1，其迭代的目标函数曲线如图 12 所示：

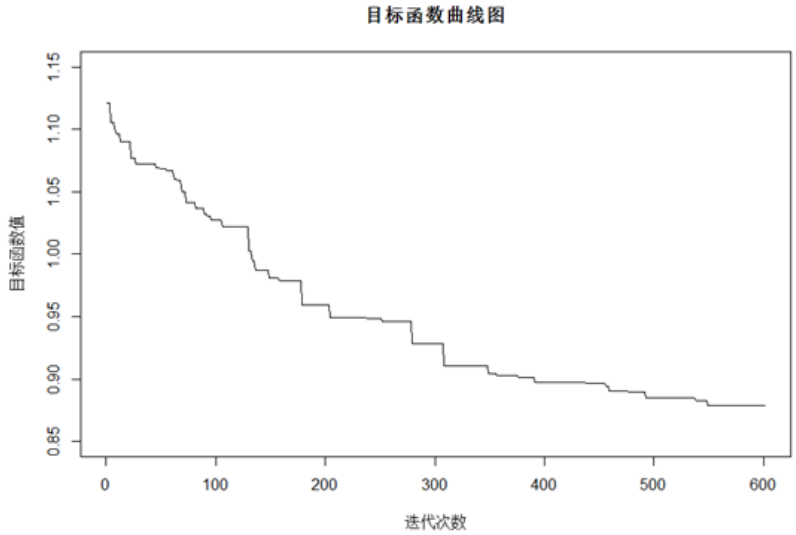

图 12 目标函数曲线图

由上图可知，在 400 代之前，目标函数值迅速下降，在 400 代以后，目标函数值基本不变，当迭代次数到达 600 代时，目标函数达到最优，目标函数值为 0.8785725，且硫含量等于 4.350393，不大于 5，满足优化条件。

（2）变量值的调整及可视化

依据第上文中多元回归的结果，我们提取出 3 个与 RON 损失量关系最显著的变量 x7,x8,x20，即 TE_5202.PV， PDT_3602.DACA 和 AT_6201.DACA。

现在，我们固定其他变量，并按照 1 的步长调整 x7 变量，按照 0.1 的步长调整 x8 变量，按照 1 的步长调整 x20 变量，得到相应的 RON 损失值，具体如表 12 所示：

表 12 RON 损失值变动表

x7	x8	x20
0.918727	0.930908	0.879031
0.905317	0.915029	0.878904
0.894282	0.901389	0.878794
0.885979	0.890499	0.878702
0.880682	0.882786	0.878629
0.878572	0.878572	0.878572
0.879726	0.878051	0.878534
0.884109	0.88127	0.878514

0.891581	0.888133	0.878511
0.901901	0.898399	0.878527
0.914741	0.911698	0.87856

为了更直观的进行分析，我们分别绘制了相应的 RON 损失值变动图，从图 13-15 可以看出，RON 的损失程度和 x7，x8，x20 均呈非线性相关，在分别在 x7=6.5，x8=7.5，x20=9.6 处取得最小值，并且 RON 损失值随着自变量的变动缓慢变化，没有出现剧烈震荡，这体现了我们的优化方案的稳健性和有效性。

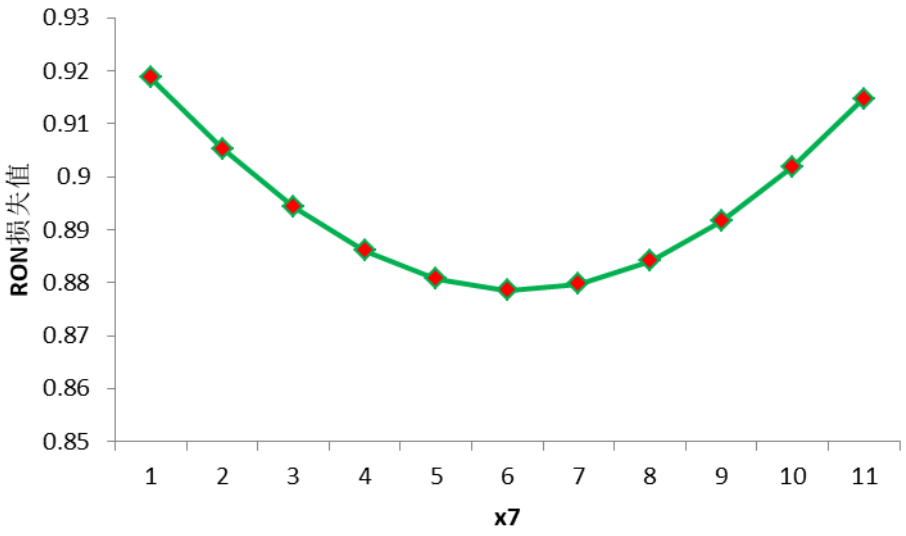

图 13 RON 损失值-x7 变动图

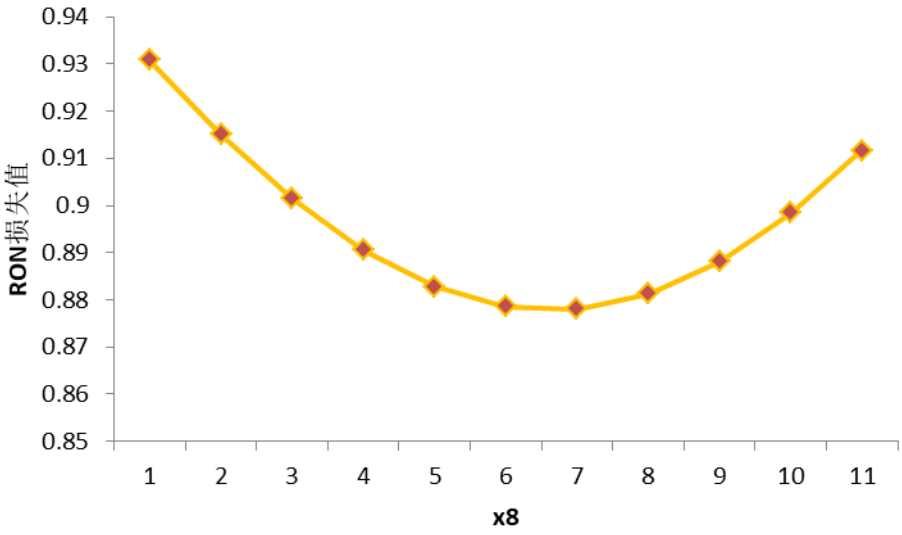

图 14 RON 损失值-x8 变动图

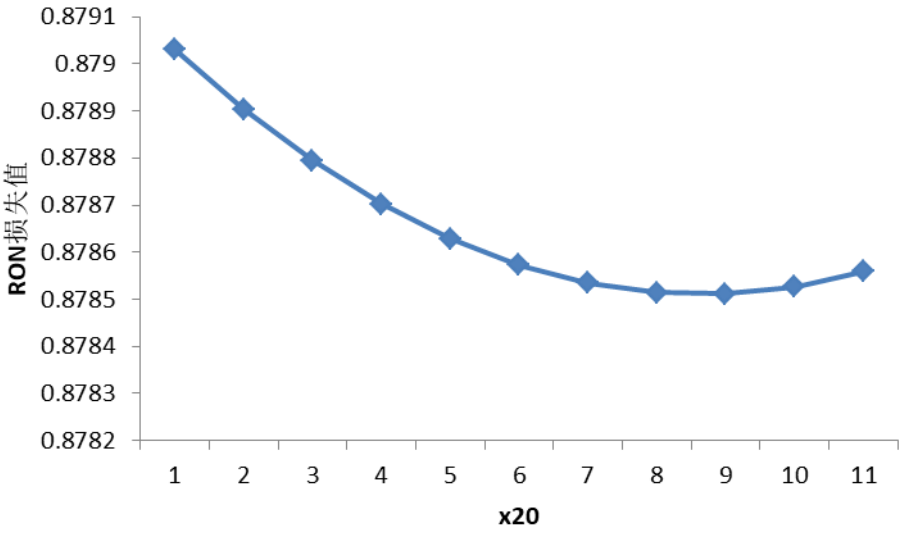

图 15 RON 损失值-x20 变动图

六、模型评价

该模型的设计通过引入降维之后的变量，使模型变量更加精简，同时便于更快的使用迭代的方式找出模型的最优解。模型的普适性较强，能够较好的减少一般原材料在加过程中 RON 的损失，将损失值降低到 0.9 左右。但是该模型并未充分利用原材料损失很小的那部分样本中所包含的信息，对该部分样本无法在原有的操作基础对其进行进一步优化使其达到更好的效果。同时在降维时的筛选并没考虑到操作的具体流程，可能会忽略了部分重要的变量，而保留的变量中有部分也未能在模型展现出一定的相关性，因此在变量的选择上也存在着一定的改进空间。

七、参考文献

[1]黄宏林,李烨,谷晓琳.优化操作条件降低汽油加氢装置辛烷值损失[J].石油化工应用,2015,34(12):116-118.
[2]刘琪.S Zorb 装置降低汽油烯烃含量的探索[J].中外能源,2020,25(01):75-80.
[3]刘昕.1100kt/a 催化汽油精制装置降低辛烷值损失的措施[J].中氮肥,2020(01):69-71.
[4]赵小燕,李成文,朱玉新.优化操作降低汽油加氢装置重汽油辛烷值损失[J].化工管理,2015(14):164.
[5]胡跃梁,孙启明.S Zorb 吸附脱硫装置运行过程中存在问题分析及应对措施[J].石油炼制与化工,2013,44(07):69-72.
[6]刘长良,张书瑶,王梓齐,陈颖,白康.基于改进 KNN 回归算法的风电机组齿轮箱状态监测[J].中国测试:1-8.
[7]白杨,姚桂林.一种基于 KNN 后处理的鲁棒性抠图方法[J].计算机应用与软件,2020,37(09):170-175.

[8]郭正军,娄丽娟,王玉杰,王海岭,姚丰菊,张瑞岭.基于 BP 神经网络模型的河南省严重精神障碍患者服药依从性影响因素分析[J].郑州大学学报(医学版),2020(05):689-693.
[9]杨桂元，朱家明.数学建模竞赛优秀论文评析[M].中国科技大学出版社.2013.2.
[10]何晓群.多元统计分析[M].中国人民大学出版社.2016.3.

八、附录

8.1 代码

```
############问题 1#########
####对 285 进行处理####

### 导包

library(VIM)
library(psych)
library(lattice)
library(mice)
library(MASS)

### 读取数据

raw285 <- read.csv("data/raw285.csv", header = T)
raw285[1:5, 1:3]
dim(raw285)

### 自定义函数

countNaN <- function(myline) {
  return(sum(is.na(myline)))
}

### 将 0 值转换为空值
deal285 <- raw285
deal285M <- as.matrix(deal285)
deal285M[which(deal285M == 0)] <- NaN
deal285DF <- as.data.frame(deal285M)


## 得到缺失数据的变量
cNa <- apply(deal285DF, 2, countNaN)

#### 问题 1--第 1 问

### 删除大量缺失的数据
#按照 80%准则，删除缺失值大于 40*20%的变量
cNa[which(cNa > 8)]
```

```
#存在 11 个变量完全缺失,按照题目要求将其删除
#      S.ZORB.FT_1501.PV       S.ZORB.FT_1002.PV       S.ZORB.FC_1202.PV
#40                      40                      40
#S.ZORB.FT_1501.TOTAL       S.ZORB.FT_5102.PV     S.ZORB.FT_2901.DACA
#40                      40                      40
#S.ZORB.FC_1104.DACA     S.ZORB.FT_2803.DACA     S.ZORB.FT_1502.DACA
#40                      40                      40
#S.ZORB.TEX_3103A.DACA S.ZORB.FT_5102.DACA.PV
#40                      40

#绘制 295 的缺失值图

### 这里读取附件 1 总数据应该 1 去除的变量
numDeleAll <- read.csv("data/numDeleAll.csv")
colnames(numDeleAll) <- c("varName", "index")

length(numDeleAll$index)
length(which(cNa > 8)) #11
deleNum_this <- unique(numDeleAll$index, which(cNa > 8))
length(deleNum_this) #17

Dealed285_f1 <- deal285DF[, -deleNum_this]
dim(Dealed285_f1) #40 337

### 问题 1--第 3 问(无缺失值,第 3 问不处理)


### 问题 1_1-3 小闻处理后数据导出
write.csv(Dealed285_f1, "output/W1_deleteed285.csv")


####处理 313 文件


### 导包

library(VIM)
library(psych)
library(lattice)
library(mice)
library(MASS)

### 读取数据

raw313 <- read.csv("data/raw313.csv", header = T)
raw313[1:5, 1:3]
dim(raw313) #40 354

### 自定义函数

countNaN <- function(myline) {
```

```
  return(sum(is.na(myline)))
}

meanNaN <- function(myline) {
  myline[is.na(myline)] <- mean(myline, na.rm = T)
  return(myline)
}


### 将 0 值转换为空值
deal313 <- raw313
deal313M <- as.matrix(deal313)
deal313M[which(deal313M == 0)] <- NaN
deal313DF <- as.data.frame(deal313M)


## 得到缺失数据的变量
cNa <- apply(deal313DF, 2, countNaN)

#### 问题 1--第 1 问

### 删除大量缺失的数据
#按照 80%准则，删除缺失值大于 40*20%的变量
cNa[which(cNa > 8)]
#存在 10 个变存在大量缺失,按照题目要求将其删除
# ctrl+shift+c 实现多行注释
# S.ZORB.FT_1501.PV       S.ZORB.FT_1002.PV    S.ZORB.FT_1501.TOTAL
# 40                      40                      40
# S.ZORB.FT_2901.DACA     S.ZORB.FC_2432.DACA     S.ZORB.FC_1104.DACA
# 40                      16                      40
# S.ZORB.FT_2803.DACA     S.ZORB.FT_1502.DACA S.ZORB.FC_2432.PIDA.SP
# 40                      40                      16
# S.ZORB.TEX_3103A.DACA
# 40

### 这里读取附件 1 总数据应该 1 去除的变量
numDeleAll <- read.csv("data/numDeleAll.csv")
colnames(numDeleAll) <- c("varName", "index")

length(numDeleAll$index)
length(which(cNa > 8)) #10
deleNum_this <- unique(numDeleAll$index, which(cNa > 8))
length(deleNum_this) #17
Dealed313_f1 <- deal313DF[, -deleNum_this]
dim(Dealed313_f1) #40 337

### 问题 1--第 3 问
#绘制缺失值图(已输出: 问题 1 第 1 问缺失值图.jpeg)
#aggr(Dealed313_f1, prop = F, numbers = T)

#查看存在缺失值的变量
```

```
cNa2 <- apply(Dealed313_f1, 2, countNaN)
cNa2[which(cNa2 > 0)]

# S.ZORB.FT_1204.PV     S.ZORB.FT_2431.DACA S.ZORB.FT_1204.DACA.PV
# 2                       6                       2

## 均值填补
num_NaN <- which(cNa2 > 0)
Dealed313_f2 <- Dealed313_f1

for (item in num_NaN) {
  Dealed313_f2[, item] <- meanNaN(Dealed313_f2[, item])
}

#查看存在缺失值的变量
cNa3 <- apply(Dealed313_f2, 2, countNaN)
cNa3[which(cNa3 > 0)]
#无缺失值
#integer(0)

### 问题 1_1-3 小闻处理后数据导出
write.csv(Dealed313_f1, "output/W1_deleteed313.csv")


####对 325 行数据进行处理(对问题 1--第 2 小问进行处理)

rawAll <- read.csv("data/data_raw.csv")
dim(rawAll)

#### 自定义函数

countNaN <- function(myline) {
  return(sum(is.na(myline)))
}

meanNaN <- function(myline) {
  myline[is.na(myline)] <- mean(myline, na.rm = T)
  return(myline)
}

#拉格朗日插值
lagrange <- function(vectorX, vectorY, newX) {
  #print(vectorX)
  #print(vectorY)
  n <-length(vectorX)
  newY <- c()
  for (xnum in c(1:length(newX))) {
    x <- newX[xnum]
    lagr <- 0
    for (i in 1:n) {
      Li <- 1
```

```
      for (j in 1:n) {
        if (i!=j)
          Li <- Li*(x - vectorX[j])/(vectorX[i] - vectorX[j])
      }
      lagr <- Li*vectorY[i] + lagr
    }
    newY <- c(newY, lagr)
  }
  return(newY)
}

mean40range <- function(myline) {
  num <- which(is.na(myline))
  print(paste("====缺失值数为:", length(num), sep = ""))
  lenL <- length(myline)
  for (item in num) {
      if (item <= 40) {
        print(item)
        print("小--")
        meanNum <- mean(myline[c(1:item, (item + 1):(2*item + 1))], na.rm = T)
        if (is.na(meanNum)) {
          meanNum <- mean(myline[c(1:100)],na.rm = T)
          print(meanNum)
        }
        print(paste("均值为:", meanNum, sep = ""))
        myline[item] <- meanNum
      }
      if (item >= (lenL-40)) {
        print(item)
        print("大--")
        meanNum <- mean(myline[c((2*item - lenL -1):(item-1), item:lenL)], na.rm = T)
        if (is.na(meanNum)) {
          meanNum <- mean(myline[c((lenL-100):lenL)],na.rm = T)
          print(meanNum)
        }
        print(paste("均值为:", meanNum, sep = ""))
        myline[item] <- meanNum
      }

    if (item > 40 & item < (lenL-40)) {
      print(item)
      print("中间--")
      meanNum <- mean(c((num-40):num, (num+1):(num+40)), na.rm = T)
      print(paste("均值为:", meanNum, sep = ""))
      myline[item] <- meanNum
    }
  }
  num <- which(is.na(myline))
  print(paste("====剩余缺失值数为:", length(num), sep = ""))
  return(myline)
}
```

```
### 将0值转换为空值
dealAll <- rawAll
dealAllM <- as.matrix(dealAll)
dealAllM[which(dealAllM == 0)] <- NaN
dealAllDF <- as.data.frame(dealAllM)

##得到缺失数据的变量
cNaAll <- apply(dealAllDF, 2, countNaN)

##大量缺失的变量
cNaAll[which(cNaAll > 65)] #325*0.2=65

# S.ZORB.FC_2301.PV       S.ZORB.FT_1501.PV       S.ZORB.FT_5104.PV
# 145                     288                     126
# S.ZORB.FT_9101.PV       S.ZORB.FT_1002.PV       S.ZORB.FC_1202.PV
# 134                     137                     219
# S.ZORB.FC_3103.PV    S.ZORB.FT_1002.TOTAL    S.ZORB.FT_1501.TOTAL
# 214                     187                     123
# S.ZORB.FT_5102.PV     S.ZORB.FT_2901.DACA     S.ZORB.FC_1104.DACA
# 109                     137                     307
# S.ZORB.FT_2803.DACA     S.ZORB.FT_1502.DACA   S.ZORB.TEX_3103A.DACA
# 297                     308                     214
# S.ZORB.FT_5102.DACA.PV S.ZORB.FT_5204.DACA.PV
# 134                     84

## 少量缺失的变量
cNaAll[which(cNaAll <= 65 & cNaAll > 0)]

# S.ZORB.FT_9301.PV             S.ZORB.FT_9402.PV             S.ZORB.FT_1003.PV
# 4                             1                             4
# S.ZORB.FT_1004.PV           S.ZORB.FT_3303.DACA           S.ZORB.LT_2901.DACA
# 19                             3                             2
# S.ZORB.FC_2432.DACA           S.ZORB.FT_2303.DACA           S.ZORB.FT_2302.DACA
# 2                             10                            3
# S.ZORB.FT_2002.DACA          S.ZORB.PDT_3601.DACA           S.ZORB.FT_3702.DACA
# 22                             1                             54
# S.ZORB.PT_7503.DACA        S.ZORB.FC_2432.PIDA.SP        S.ZORB.FT_1006.DACA.PV
# 1                             2                            34
# S.ZORB.FT_1006.TOTALIZERA.PV S.ZORB.FT_1503.TOTALIZERA.PV S.ZORB.FT_1504.TOTALIZERA.PV
# 14                             2                             2

####处理缺失值, 缺失值少的

num_65 <- which(cNaAll <= 65 & cNaAll > 0)
dealAllDF2 <- dealAllDF
for (item in num_65) {
  print(item)
  dealLine <- dealAllDF2[, item]
  dealAllDF2[, item] <- mean40range(dealLine)
  print("----------")
```

```
}

cNaAll2 <- apply(dealAllDF2, 2, countNaN)
which(cNaAll2 <= 65 & cNaAll2 > 0) #只剩下缺失值多的
length(which(cNaAll2 <= 65 & cNaAll2 > 0))

#去除缺失值过多的
dealAllDF3 <- dealAllDF2[, -which(cNaAll2 > 0)]

dim(dealAllDF2)
dim(dealAllDF3)

cNaAll <- apply(dealAllDF3, 2, countNaN)
which(cNaAll > 0)

numDeleAll <- which(cNaAll2 > 0)

write.csv(numDeleAll, "data/numDeleAll.csv") #要删除的列号
write.csv(dealAllDF3, "output/W1_deleteedAll.csv")

########问题 2#########
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor

#使用 seaborn 做可视化
import seaborn as sns
%matplotlib inline
from scipy import stats
import math
from sklearn.utils.multiclass import type_of_target
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False
mydata = pd.read_excel("F:/附件一：325 个样本数据.xlsx",encoding='utf8')
mydata_y_1 = mydata.drop(['RON 损失','硫含量,μg/g','辛烷值 RON','成效'], 1)
mydata_y = mydata.drop(['RON 损失','硫含量,μg/g','辛烷值 RON'], 1)
 # 保存变量中的缺失值
data = train.replace(0,np.nan)
a = train.isnull().sum()/len(train)*100
# 保存列名

variables = train.columns

variable = []

for i in a.index:
```

```
    if a.loc[i]<=20:   #setting the threshold as 20%

        variable.append(train[i])
# 获取缺失值比例 > 20% 的列
missing_df = missing_values_table(data)
missing_columns = list(missing_df[missing_df['% of Total Values'] > 50].index)
print('We will remove %d columns.' % len(missing_columns))

# 删除缺失值比例高于 50%的列
data = data.drop(columns = list(missing_columns))
def iv_woe(data:pd.DataFrame, target:str, bins:int = 10) -> (pd.DataFrame, pd.DataFrame):
    """计算 woe 和 IV 值

    参数:
    - data: dataframe 数据
    - target: y 列的名称
    - bins: 分箱数（默认是 10）
    """
    newDF,woeDF = pd.DataFrame(), pd.DataFrame()
    cols = data.columns
    for ivars in cols[~cols.isin([target])]:
        # 数据类型在 bifc 中、且数据>10 则分箱
        if (data[ivars].dtype.kind in 'bifc') and (len(np.unique(data[ivars]))>10):
            binned_x = pd.qcut(data[ivars], bins,  duplicates='drop')
            d0 = pd.DataFrame({'x': binned_x, 'y': data[target]})
        else:
            d0 = pd.DataFrame({'x': data[ivars], 'y': data[target]})
        d = d0.groupby("x", as_index=False).agg({"y": ["count", "sum"]})
        d.columns = ['Cutoff', 'N', 'Events']
        d['% of Events'] = np.maximum(d['Events'], 0.5) / d['Events'].sum()
        d['Non-Events'] = d['N'] - d['Events']
        d['% of Non-Events'] = np.maximum(d['Non-Events'], 0.5) / d['Non-Events'].sum()
        d['WoE'] = np.log(d['% of Events']/d['% of Non-Events'])
        d['IV'] = d['WoE'] * (d['% of Events'] - d['% of Non-Events'])
        d.insert(loc=0, column='Variable', value=ivars)
        print("Information value of " + ivars + " is " + str(round(d['IV'].sum(),6)))
        temp =pd.DataFrame({"Variable" : [ivars], "IV" : [d['IV'].sum()]}, columns =
["Variable", "IV"])
        newDF=pd.concat([newDF,temp], axis=0)
        woeDF=pd.concat([woeDF,d], axis=0)
    return newDF
IV=iv_woe(mydata_y,'成效')
IV_sort = IV.sort_values(by='IV',ascending = False)
x=0
for i in range(len(IV_sort)):
    if IV_sort.iloc[i,1]>0.03:
        x = x+1
v = IV_sort.head(x)
data_sort = mydata_y[v['Variable']]
def get_var_no_colinear(cutoff, df):
    corr_high = df.corr().applymap(lambda x: np.nan if abs(x)>cutoff else x).isnull()
```

```
    col_all = corr_high.columns.tolist()
    del_col = []
    i = 0
    while i < len(col_all)-1:
        ex_index                                                                    =
corr_high.iloc[:,i][i+1:].index[np.where(corr_high.iloc[:,i][i+1:])].tolist()
        for var in ex_index:
            col_all.remove(var)
        corr_high = corr_high.loc[col_all, col_all]
        i += 1
    return col_all
data_cor = get_var_no_colinear(0.5, data_sort)
data_corr = mydata_y[data_cor]
corr_train = data_corr.corr()
len(data_cor)
# 指定画幅
plt.figure(figsize=(10,8))
# 绘制热力图
sns.heatmap(corr_train , cmap='YlGnBu')
plt.savefig("热力图.png")
#归一化
data_fea = data_corr.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
var = data_fea.var()#获取每一列的方差
var.plot()
plt.savefig("方差图.png")
 cols = data_fea.columns
col = [ ]

for i in range(0,len(var)):
    if var.iloc[i] >= 0.02:#将阈值设置为0.001
        col.append(cols[i])

print("高于阈值的特征共%d 个；"%len(col),"\n 它们分别是：", col)
data_end = mydata_y[col]
data_fea = data_end.iloc[:,1:]#取数据中指标所在的列

model = RandomForestRegressor(random_state=1, max_depth=10)
data_fea = data_fea.fillna(0)#随机森林只接受数字输入，不接受空值、逻辑值、文字等类型
data_fea=pd.get_dummies(data_fea)
model.fit(data_fea,mydata_y.成效)
#根据特征的重要性绘制柱状图
features = data_fea.columns
importances = model.feature_importances_
indices = np.argsort(importances[0:20])  # 因指标太多，选取前 10 个指标作为例子
plt.title('Index selection')
plt.barh(range(len(indices)), importances[indices], color='pink', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative importance of indicators')
plt.show()
plt.savefig("随机森林.jpg")
a = features.T
```

```
c = importances.T
b =pd.DataFrame()
b['features']=a
b['importances']=c
b = b.sort_values(by='importances',ascending = False)
e = b.head(29)
data_end1 = mydata_y[e['features']]
#########问题 3########

####问题 3：回归####

####导包####

library(neuralnet)
library(h2o)
library(rpart)
library(e1071)
library(caret)

####初始化环境####

h2o.init()

####读取数据####

data_W3 <- read.csv("data/W2_data02.csv", header = T)

dim(data_W3)
head(data_W3)

####整理数据格式####
name_W3 <- names(data_W3)
RonNum <-  which(name_W3 == "RON")
tempRon <- data_W3["RON"]
new_data_W3 <- data_W3[, -RonNum]
new_data_W3$RON <- tempRon
head(new_data_W3)

#描述性统计
summary(new_data_W3)

####标准化数据####
scale_data_W3 <- as.data.frame(scale(new_data_W3, center = T, scale = T))
head(scale_data_W3)
dim(scale_data_W3)

####自建函数####
my_lmFunc <- function(trainDf, testDf, lineTF = T) {
  my_list <- list()
  if (lineTF == T) {
      n <- names(trainDf)
```

```
    myFormula <- as.formula(paste("RON ~",
                                  paste(n[!(n %in% "RON")],
                                        collapse = " + ")))
    lmTest <- lm(myFormula, data = trainDf)
  } else if (lineTF == F) {
    ##lm 简单回归(T)
    lm_step1 <- my_lmFunc(trainDf, testDf, lineTF = T)
    #提取 summary 中的 p 值
    lm_p_value <- summary(lm_step1$model)$coefficients[, 4]
    #取出哪些变量显著
    remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])

    nameTemp <- remarkableValue
    print(nameTemp)
    nameSplicing <- c()
    #nameMult <- c()
    for (i in c(1 : length(nameTemp))) {
      # nameMult <- c(nameMult,
      #               paste("I(", nameTemp[i], "^2)", sep = ""))
      for (j in c(i : length(nameTemp))) {
        nameSplicing <- c(nameSplicing,
                          paste("I(", nameTemp[i], "*", nameTemp[j], ")",
                                sep = ""))
      }
    }
    #print(nameSplicing)
    myFormula <- as.formula(paste("RON ~",
                                  paste(nameTemp, collapse = " + "),
                                  "+",
                                  paste(nameSplicing, collapse = " + ")))
    #print(myFormula)
    lmTest <- lm(myFormula, data = trainDf)
  }
  #模型
  my_list$model <- lmTest
  #测试集预测值
  my_list$preTest <- predict(lmTest, testDf)
  #训练集预测值
  my_list$preTrain <- predict(lmTest, trainDf)
  return(my_list)
}

my_network <- function(trainDf, testDf, hidden = c(8, 5)) {
  my_list <- list()
  numY <- length(trainDf)
  h2o_deep <- h2o.deeplearning(1:(numY-1), numY,
                               as.h2o(trainDf),
                               hidden = hidden,
                               export_weights_and_biases = T,
                               loss = "Quadratic",
                               epochs = 100)
```

```
  my_list$model <- h2o_deep
  my_list$preTest <- as.vector(h2o.predict(h2o_deep, as.h2o(testDf)))
  my_list$preTrain <- as.vector(h2o.predict(h2o_deep, as.h2o(trainDf)))
  return(my_list)
}

my_svm <- function(trainDf, testDf, costs = 1, gammas = 0.01, kernel = "radial") {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  best.svm <- best.tune(svm,
                        myFormula,
                        data = trainDf,
                        kernel = kernel,cost = costs,gamma = gammas)

  my_list$model <- best.svm
  my_list$preTest <- predict(best.svm, testDf)
  my_list$preTrain <- predict(best.svm, trainDf)

  return(my_list)
}

my_lmtree <- function(trainDf, testDf, cp = 0.0001) {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  lmtree <- rpart(myFormula,
                  data = trainDf, cp = cp)
  my_list$model <- lmtree
  my_list$preTest <- predict(lmtree, testDf)
  my_list$preTrain <- predict(lmtree, trainDf)

  return(my_list)
}

my_knn <- function(trainDf, testDf, knum = 5) {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  my_knnreg <- knnreg(myFormula, trainDf, k = knum)

  my_list$model <- my_knnreg
  my_list$preTest <- predict(my_knnreg, testDf)
  my_list$preTrain <- predict(my_knnreg, trainDf)

  return(my_list)
}

my_checkout <- function(preData, RealData) {
  lenData <- length(preData)
  ME <- sum(RealData - preData)/lenData
```

```
  MAD <- sum(abs(RealData - preData))/lenData
  MSE <- sum((RealData - preData)^2)/lenData
  MPE <- sum(abs(RealData - preData)/abs(RealData))/lenData
  checkDf <- c(ME, MAD, MSE, MPE)
  names(checkDf) <- c("ME", "MAD", "MSE", "MAPE")
  return(checkDf)
}

my_overlapping <- function(dataDF, knum = 8, med = "my_lm") {
  outList <- list()
  numSample <- dim(dataDF)[1]
  #预测矩阵
  pre_temp <- matrix(nrow = knum, ncol = round(numSample/knum))
  #误差矩阵
  check_temp <- matrix(nrow = knum, ncol = 4) #4 为检验统计量的个数
  colnames(check_temp) <- c("ME", "MAD", "MSE", "MPE")
  for (item in c(1:knum)) {
    test_num <- sample(c(1:numSample), size = round(numSample/knum))
    #print(test_num)
    #print("---------")
    testDf <- dataDF[test_num, ]
    test_y <- testDf[, "RON"]
    trainDf <- dataDF[-test_num, ]
    train_y <- trainDf[, "RON"]

    #这里调用方法
    if (med == "my_lm") {
      model_A <- my_lmFunc(trainDf, testDf, T)
      model_B <- my_lmFunc(trainDf, testDf, F)

      #查看调整后的 R 方值
      adjR_A <- summary(model_A$model)$adj.r.squared
      adjR_B <- summary(model_B$model)$adj.r.squared

      if (adjR_A > adjR_B) {
        print("我是多元线性回归")
        modeling <- model_A
      } else {
        print("我是多项式模型")
        modeling <- model_B
      }

    } else if (med == "my_net") {
      print("我是 NETWORK")
      modeling <- my_network(trainDf, testDf, hidden = c(8, 5))
    } else if (med == "my_svm") {
      print("我是 SVM")
      modeling <- my_svm(trainDf, testDf)
    } else if (med == "my_tree") {
      print("我是 tree")
      modeling <- my_lmtree(trainDf, testDf)
```

```
    } else if (med == "my_knn") {
      print("我是knn")
      modeling <- my_knn(trainDf, testDf)
    }

    pre_temp[item, ] <- modeling$preTest
    #0.632*test_check + 0.368*train_check
    test_check <- my_checkout(modeling$preTest, test_y)
    train_check <- my_checkout(modeling$preTrain, train_y)

    check_output <- 0.632*test_check + 0.368*train_check
    #train_check <- my_checkout()
    check_temp[item, ] <- check_output
  }
  outList$pre <- pre_temp
  outList$check <- check_temp
  return(outList)
}

###函数调用
#要使用的方法

medName <- c("my_lm", "my_net", "my_svm", "my_tree", "my_knn")

for (item in medName) {
  assign(paste(item, "_outcome", sep = ""),
         my_overlapping(scale_data_W3, med = item))
}


my_lm_outcome$pre


# my_overlapping(scale_data_W3, med = "my_net")
# my_svm_outcome

mean_check_lm <- apply(my_lm_outcome$check, 2, mean)
mean_check_net <- apply(my_net_outcome$check, 2, mean)
mean_check_svm <- apply(my_svm_outcome$check, 2, mean)
mean_check_tree <- apply(my_tree_outcome$check, 2, mean)
mean_check_knn <- apply(my_knn_outcome$check, 2, mean)
check_df <- data.frame(lm = mean_check_lm,
                       net = mean_check_net,
                       svm = mean_check_svm,
                       tree = mean_check_tree,
                       knn = mean_check_knn)

####输出检验矩阵####
write.csv(check_df, "output/check_df.csv")

####展示预测值####
```

```
numS <- dim(scale_data_W3)[1]
chosNum <- sample(c(1:numS), size = 10)

my_show_train <- scale_data_W3[-chosNum, ]
my_show_test <- scale_data_W3[chosNum, ]


###多元回归###

show_lmA <- my_lmFunc(my_show_train, my_show_test, T)
show_lmB <- my_lmFunc(my_show_train, my_show_test, F)

summary(show_lmA$model) #0.9581
summary(show_lmB$model) #0.9664

plot(show_lmB$model, las = 1)

show_pre_lm <- show_lmB$preTest
show_real_lm <- my_show_test[, length(my_show_test)]

my_checkout(show_pre_lm, show_real_lm)

show_df_lm <- data.frame(pre = show_pre_lm, real = show_real_lm)
write.csv(show_df_lm, "output/show_df_lm.csv")


###神经网络


my_network <- function(trainDf, testDf, hidden = c(8, 5), epochs = 100) {
  my_list <- list()
  numY <- length(trainDf)
  h2o_deep <- h2o.deeplearning(1:(numY-1), numY,
                               as.h2o(trainDf),
                               hidden = hidden,
                               export_weights_and_biases = T,
                               epochs = epochs,
                               loss = "Quadratic")
  my_list$model <- h2o_deep
  my_list$preTest <- as.vector(h2o.predict(h2o_deep, as.h2o(testDf)))
  my_list$preTrain <- as.vector(h2o.predict(h2o_deep, as.h2o(trainDf)))
  return(my_list)
}


check_matr <- matrix(0, ncol = 4, nrow = 120)
check_matr_name <- c()
index_c <- 1
for (i in 1:15) {
  for (j in i:15) {
```

```
    print(paste(j, "--", i, sep = ""))
    check_matr_name <- c(check_matr_name, paste(j, "_", i, sep = ""))
    tempNet <- my_network(my_show_train, my_show_test, hidden = c(j, i))
    print(my_checkout(tempNet$preTest, my_show_test[, length(my_show_test)]))
    check_matr[index_c,      ]     <-     my_checkout(tempNet$preTest,     my_show_test[,
length(my_show_test)])
    index_c <- index_c + 1
  }
}

rownames(check_matr) <- check_matr_name

colnames(check_matr) <- c("ME", "MAD", "MSE", "MPE")

min_net_number <- apply(check_matr, 2, min)

which(check_matr[, 2] == min_net_number[2])
which(check_matr[, 3] == min_net_number[3])
which(check_matr[, 4] == min_net_number[4])

write.csv(check_matr, "output/check_NetWork_Matrix.csv")


show_network <- my_network(my_show_train, my_show_test, hidden = c(8, 5),
                           epochs = 500)


show_pre_network <- show_network$preTest
show_real_network <- my_show_test[, length(my_show_test)]

my_checkout(show_pre_network, show_real_network)

show_df_network <- data.frame(pre = show_pre_network, real = show_real_network)
write.csv(show_df_network, "output/show_df_network.csv")


plot(show_network$model)

show_network$model


###SVM##

show_svm <- my_svm(my_show_train, my_show_test)
show_pre_svm <- show_svm$preTest
show_real_svm <- my_show_test[, length(my_show_test)]

my_checkout(show_pre_svm, show_real_svm)

show_df_svm <- data.frame(pre = show_pre_svm, real = show_real_svm)
write.csv(show_df_svm, "output/show_df_svm.csv")
```

```
###回归树###

colnames(my_show_train) <- c(paste("x", 1:29, sep = ""), "RON")
colnames(my_show_test) <- c(paste("x", 1:29, sep = ""), "RON")

show_lmtree <- my_lmtree(my_show_train, my_show_test)
show_pre_lmtree <- show_lmtree$preTest
show_real_lmtree <- my_show_test[, length(my_show_test)]

my_checkout(show_pre_lmtree, show_real_lmtree)

show_df_lmtree <- data.frame(pre = show_pre_lmtree, real = show_real_lmtree)
write.csv(show_df_lmtree, "output/show_df_lmtree.csv")

plot(show_lmtree$model, uniform = T,
     branch = 1, compress = T)
par(xpd = TRUE)
text(show_lmtree$model, use.n = F, all = F, bg = T)


### knn ###


show_knn <- my_knn(my_show_train, my_show_test)
show_pre_knn <- show_knn$preTest
show_real_knn <- my_show_test[, length(my_show_test)]

my_checkout(show_pre_knn, show_real_knn)

show_df_knn <- data.frame(pre = show_pre_knn, real = show_real_knn)
write.csv(show_df_knn, "output/show_df_knn.csv")

yyy <- show_knn$preTrain
xxx <- my_show_train[, "RON"]
plot(xxx, yyy, xlab = "预测值", ylab = "真实值",
     xlim = c(-4, 2), ylim = c(-4, 2), col = "blue", pch = 3)
abline(0, 1, col = "red", lwd = 3)


####变量挑选####
# aleady_value <- read.csv("", header = T)
# temp_value <- read.csv("", header = T)
#
#
# ##整理数据格式##
# name_aleady_value <- names(aleady_value)
# RonNum2 <-  which(name_aleady_value == "RON")
# tempRon2 <- data_W3["RON"]
# new_aleady_value <- data_W3[, -RonNum2]
# new_aleady_value$RON <- tempRon2
```

```
# head(new_aleady_value)
#
# #### 挑选变量####
#
# my_choseValue <- function(aleady_value, temp_value, knum) {
#   name_temp_value <- names(temp_value)
#   lenVal <- length(name_temp_value)
#   if (knum >= lenVal) {
#     print("输入无效!!!")
#     break
#   } else {
#     zuhe <- t(combn(lenVal, knum))
#     index_test <- dim(zuhe)[1]
#     for (item in c(1:index_test)) {
#       col_index <- zuhe[item, ]
#       choseed_Val <- temp_value[, col_index]
#       #效果检验函数(调用上面的函数)
#       outComing <- my_testDf_Func(aleady_value, choseed_Val)
#
#
#     }
#   }
# }
#
# name_temp_value <- names(temp_value)
#
#
#
#
# ####测试代码####
#
# ##lm 简单回归(T)
# test_lm1 <- my_lmFunc(scale_data_W3, scale_data_W3, lineTF = T)
# summary(test_lm1$model)
# # Residual standard error: 0.2075 on 296 degrees of freedom
# # Multiple R-squared:  0.9607,    Adjusted R-squared:  0.9569
# # F-statistic: 258.1 on 28 and 296 DF,  p-value: < 2.2e-16
#
# summary(test_lm1$model)$adj.r.squared #0.9569351
#
# #多项式回归
test_lm2 <- my_lmFunc(scale_data_W3, scale_data_W3, lineTF = F)
summary(test_lm2$model)
# # Residual standard error: 0.2028 on 280 degrees of freedom
# # Multiple R-squared:  0.9644,    Adjusted R-squared:  0.9589
# # F-statistic: 172.6 on 44 and 280 DF,  p-value: < 2.2e-16
#
# names(summary(test_lm2$model))
# # [1] "call"          "terms"         "residuals"     "coefficients"
# # [5] "aliased"       "sigma"         "df"            "r.squared"
# # [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
#
# summary(test_lm2$model)$adj.r.squared #0.9588524
#
# #提取 summary 中的 p 值
# lm_p_value <- summary(test_lm1$model)$coefficients[, 4]
# #取出哪些变量显著
# remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])
#
# ###neuralnet
#
# n = names(scale_data_W3)
# myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
#
# #未标准化前预测结果相同，标准化后，预测结果不同
# out_neuralnet <- neuralnet(myFormula, data = scale_data_W3,
#                            hidden = c(5, 3), linear.output = T)
#
# #plot(out_neuralnet)
#
# pre_neuralnet <- compute(out_neuralnet, data_W3)
# pre_neuralnet$net.result
#
# class(pre_neuralnet$net.result)
#
# my_checkout(pre_neuralnet$net.result, scale_data_W3$RON)
#
# ### h2o.deeplearning
# #library(h2o)
#
# h2o.init()
# h2o_deep_data_W3 <- h2o.deeplearning(1:(length(scale_data_W3)-1), length(scale_data_W3),
#                              as.h2o(scale_data_W3),hidden=c(5,3),
#                              export_weights_and_biases=T)
#
# h2o_pre <- h2o.predict(h2o_deep_data_W3, as.h2o(scale_data_W3))
# head(h2o_pre)
# class(h2o_pre) #H2OFrame
# my_checkout(as.vector(h2o_pre), scale_data_W3$RON)
# # ME          MAD          MSE          MPE
# # -0.04129501  0.56056624  0.52518669  1.94182117
#
# plot(h2o_deep_data_W3)
#
# ###
# library(e1071)
#
# costs <- 1
# gammas <- 0.01
# kernel <- "radial"
# n = names(scale_data_W3)
# myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
```

```
#
# best.svm <- best.tune(svm,
#                       myFormula,
#                       data=scale_data_W3,kernel = kernel,cost = costs,gamma = gammas)
#
# svm_pre <- predict(best.svm, scale_data_W3)
# class(svm_pre) #numeric
#
# my_checkout(svm_pre, scale_data_W3$RON)
# # ME         MAD         MSE         MPE
# # -0.05824511  0.43850015  0.39410866  1.29859733
#
# ##回归树
# library(rpart)
# n <- names(scale_data_W3)
# myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
# lmtree <- rpart(myFormula,
#             data = scale_data_W3, cp = 0.0001)
#
# lmtree_pre <- predict(lmtree, scale_data_W3)
# class(lmtree_pre) #numeric
#
# my_checkout(lmtree_pre, scale_data_W3$RON)
# # ME           MAD           MSE           MPE
# # -8.155869e-18  3.310258e-01  1.967090e-01  1.211644e+00
# n = names(scale_data_W3)
# myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
# my_test_knnreg <- knnreg(myFormula, scale_data_W3, k = 5)
#
# pre_test_knn <- predict(my_test_knnreg, scale_data_W3)
# #class(pre_test_knn) #numeric
#
# my_checkout(pre_test_knn, scale_data_W3$RON)
# # ME         MAD         MSE         MPE
# # -0.02480938  0.35028092  0.24141016  1.15098557


#########问题 4########


####问题 4：优化####

#注意: 只有操作变量可以进入方程
#所以, 在问题 2 得到的变量中，需要把非操作变量删去
#注意, 这里的 RON 是减小值
#可以不需要方程
#再次进行第三题的操作，对新的数据集进行拟合，选取拟合效果最好的模型，但这次没有训练集，是全部样本

####导包####
```

```
library(neuralnet)
library(h2o)
library(rpart)
library(e1071)
library(caret)

####初始化环境####

h2o.init()



####读取数据####

#自变量与因变量数据
data_W4 <- read.csv("data/W4_data02.csv", header = T)
head(data_W4)
dim(data_W4) #325  31

#读取自变量限制
limit_M <- read.csv("data/limit_Val.csv", header = T)

limit_M[, 1:4]
rownames(limit_M) <- limit_M$X
limit_M <- limit_M[, -1]
limit_M_Mat_t <- t(limit_M)
head(limit_M_Mat_t)
dim(limit_M_Mat_t) #354 2

###为原材料添加新的限制####
limit_new_df <- data.frame(Down = c(0, 0), Up = c(0, 0))
rownames(limit_new_df) <- c("Raw_RON", "JiaoTan")

limit_M_Mat_t <- rbind(limit_M_Mat_t, limit_new_df)
dim(limit_M_Mat_t) #356 2

dim(data_W4)
head(data_W4)

#改变名字的 data_W4

nameChange_data_W4 <- data_W4
colnames(nameChange_data_W4) <- c(paste("x", seq(1:(dim(nameChange_data_W4)[2] - 2)), sep =
""),
                                  "RON", "LiuHanLiang")

head(nameChange_data_W4)

#RON 和 LiuHanLiang 矩阵
dfRon_F <- nameChange_data_W4[, -length(nameChange_data_W4)]
dfHLL_F <- nameChange_data_W4[, -(length(nameChange_data_W4) - 1)]
```

```
head(dfRon_F)
head(dfHLL_F)



####数据准备####

##中心化标准化
scale_data_W4 <- as.data.frame(scale(data_W4, center = T, scale = T))
head(scale_data_W4)
dim(scale_data_W4)

###构建数据集
##RON 作为因变量的数据集
dfRon <- scale_data_W4[, -length(scale_data_W4)]
head(dfRon)
dim(dfRon)
colName_Ron <- colnames(dfRon)

#更改自变量名
colnames(dfRon) <- c(paste("x", seq(1:(dim(dfRon)[2] - 1)), sep = ""), "RON")
head(dfRon)

##LiuHanLiang 作为因变量的数据集
dfHLL <- scale_data_W4[, -(length(scale_data_W4) - 1)]
head(dfHLL)
dim(dfHLL)

#提取变量名
colName_HLL <- colnames(dfHLL)

#更改自变量名
colnames(dfHLL) <- c(paste("x", seq(1:(dim(dfHLL)[2] - 1)), sep = ""), "LiuHanLiang")
head(dfHLL)

#需要的自变量名登记
name_ZiVal <- colName_HLL[-length(colName_HLL)]
length(name_ZiVal) #27

#全部变量名登记
name_AllVal <- rownames(limit_M_Mat_t)
length(name_AllVal) #354

#判断需要的自变量位置及提取限制
ZiVal <- c()
for (item in 1:length(name_ZiVal)) {
  temp <- name_ZiVal[item]
  num <- which(temp == name_AllVal)
  ZiVal <- c(ZiVal, num)
}
```

```
head(name_AllVal[ZiVal])
head(name_ZiVal)
#OK!!!

#需要的限制数据集
limit_Need <- as.data.frame(limit_M_Mat_t[ZiVal, ])
rownames(limit_Need)

#对限制数据集进行改名
limit_Need_ChangeName <- limit_Need
rownames(limit_Need_ChangeName) <- paste("x", seq(1:(dim(limit_Need_ChangeName)[1])), sep =
"")

####方法区####
my_lmFunc <- function(trainDf, testDf, lineTF = T, yName) {
  my_list <- list()
  if (lineTF == T) {
    n <- names(trainDf)
    myFormula <- as.formula(paste(yName, "~",
                                  paste(n[!(n %in% yName)],
                                        collapse = " + ")))
    lmTest <- lm(myFormula, data = trainDf)
  } else if (lineTF == F) {
    ##lm 简单回归(T)
    lm_step1 <- my_lmFunc(trainDf, testDf, lineTF = T, yName = yName)
    #提取 summary 中的 p 值
    lm_p_value <- summary(lm_step1$model)$coefficients[, 4]
    #取出哪些变量显著
    remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])
    nameTemp <- remarkableValue
    print(nameTemp)
    nameSplicing <- c()
    #nameMult <- c()
    for (i in c(1 : length(nameTemp))) {
      # nameMult <- c(nameMult,
      #               paste("I(", nameTemp[i], "^2)", sep = ""))
      for (j in c(i : length(nameTemp))) {
        nameSplicing <- c(nameSplicing,
                          paste("I(", nameTemp[i], "*", nameTemp[j], ")",
                                sep = ""))
      }
    }
    #print(nameSplicing)
    myFormula <- as.formula(paste(yName, "~",
                                  paste(nameTemp, collapse = " + "),
                                  "+",
                                  paste(nameSplicing, collapse = " + ")))
    #print(myFormula)
    lmTest <- lm(myFormula, data = trainDf)
  }
```

```
  #模型
  my_list$model <- lmTest
  #测试集预测值
  my_list$preTest <- predict(lmTest, testDf)
  #训练集预测值
  my_list$preTrain <- predict(lmTest, trainDf)
  return(my_list)
}

####多项式回归####
RON_lm <- my_lmFunc(dfRon, dfRon,
                    lineTF = F, yName = "RON")
summary(RON_lm$model) # 0.2547

LHL_lm <- my_lmFunc(dfHLL, dfHLL,
                    lineTF = F, yName = "LiuHanLiang")
summary(LHL_lm$model) #0.2929

Ron_value <- summary(RON_lm$model)$coefficients[, 4]
remarkable_Ron_value <- names(Ron_value[which(Ron_value < 0.05)])
length(remarkable_Ron_value) #7
Ron_string <- paste(remarkable_Ron_value, collapse = " + ")
Ron_Formula <- as.formula(paste("RON ~", Ron_string))
# RON ~ x5 + x20 + I(x2 * x24) + I(x3 * x3) +
#   I(x5 * x20) + I(x7 * x19) + I(x19 * x20)


#利用表格查看变量个数
temp <- gsub("[\\+|\\I(|\\)|\\*]+", "", Ron_string)
tempTable <- strsplit(temp, split = "[ ]+")
table(tempTable)
# x19  x2 x20 x24  x3  x5  x7
# 2   1   3   1   2   2   1

length(table(tempTable)) #7

name_Use_Ron <- names(table(tempTable))
#"x19" "x2"  "x20" "x24" "x3"  "x5"  "x7"

####利用得到的变量建立多项式方程(RON)####
lm_Ron_new <- lm(Ron_Formula, data = nameChange_data_W4[, c(name_Use_Ron, "RON")])
summary(lm_Ron_new) #0.2188
#yRon = -0.7971 + 0.0003889*x5 +2.334*x20 - 0.02861*x2*x24 + 0.00004222*x3^2 - 0.004541*x5*x20
- 0.00074*x7*x19 + 0.0237*x19*x20

#Ron 的限制矩阵
Ron_Limit_Df <- limit_Need_ChangeName[name_Use_Ron, ]

LHL_value <- summary(LHL_lm$model)$coefficients[, 4]
remarkable_LHL_value <- names(LHL_value[which(LHL_value < 0.05)])
length(remarkable_LHL_value) #4
```

```
LHL_string <- paste(remarkable_LHL_value, collapse = " + ")
LHL_Formula <- as.formula(paste("LiuHanLiang ~", LHL_string))
#LiuHanLiang ~ x5 + x25 + I(x9 * x18) + I(x16 * x26)

#这里的字符串操作可以使用正则表达式
temp <- gsub("[\\+|\\I(|\\)|\\*]+", "", LHL_string) #替换全部参数
tempTable <- strsplit(temp, split = "[ ]+") #字符串分隔
table(tempTable)
# x16 x18 x25 x26  x5  x9
# 1   1   1   1   1   1

length(table(tempTable)) #6

name_Use_LHL <- names(table(tempTable))
#"x16" "x18" "x25" "x26" "x5"  "x9"

#LHL 的限制矩阵
LHL_Limit_Df <- limit_Need_ChangeName[name_Use_LHL, ]

####利用得到的变量建立多项式方程(LHL)####
lm_LHL_new <- lm(LHL_Formula, data = nameChange_data_W4[, c(name_Use_LHL, "LiuHanLiang")])
summary(lm_LHL_new) # 0.1318
#yLhl = -17.22 + 0.05035*x5 + 0.02646*x25 - 0.00003203*x9*x18 + (6.215e-08)*x16*x26

####自定义函数####

my_lmFunc <- function(trainDf, testDf, lineTF = T) {
  my_list <- list()
  if (lineTF == T) {
    n <- names(trainDf)
    myFormula <- as.formula(paste("RON ~",
                                  paste(n[!(n %in% "RON")],
                                        collapse = " + ")))
    lmTest <- lm(myFormula, data = trainDf)
  } else if (lineTF == F) {
    ##lm 简单回归(T)
    lm_step1 <- my_lmFunc(trainDf, testDf, lineTF = T)
    #提取 summary 中的 p 值
    lm_p_value <- summary(lm_step1$model)$coefficients[, 4]
    #取出哪些变量显著
    remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])

    #添加对截距项处理
    remarkableValue <- remarkableValue[!(remarkableValue %in% "(Intercept)")]

    nameTemp <- remarkableValue
    print(remarkableValue)
    nameSplicing <- c()
    #nameMult <- c()
    for (i in c(1 : length(nameTemp))) {
```

```
      # nameMult <- c(nameMult,
      #               paste("I(", nameTemp[i], "^2)", sep = ""))
      for (j in c(i : length(nameTemp))) {
        print(j)
        nameSplicing <- c(nameSplicing,
                          paste("I(", nameTemp[i], "*", nameTemp[j], ")",
                                sep = ""))
      }
    }
    #print(nameSplicing)
    myFormula <- as.formula(paste("RON ~",
                                  paste(nameTemp, collapse = " + "),
                                  "+",
                                  paste(nameSplicing, collapse = " + ")))
    #print(myFormula)
    lmTest <- lm(myFormula, data = trainDf)
  }
  #模型
  my_list$model <- lmTest
  #测试集预测值
  my_list$preTest <- predict(lmTest, testDf)
  #训练集预测值
  my_list$preTrain <- predict(lmTest, trainDf)
  return(my_list)
}


my_network <- function(trainDf, testDf, hidden = c(5, 3)) {
  my_list <- list()
  numY <- length(trainDf)
  h2o_deep <- h2o.deeplearning(1:(numY-1), numY,
                               as.h2o(trainDf),
                               hidden = hidden,
                               export_weights_and_biases = T)
  my_list$model <- h2o_deep
  my_list$preTest <- as.vector(h2o.predict(h2o_deep, as.h2o(testDf)))
  my_list$preTrain <- as.vector(h2o.predict(h2o_deep, as.h2o(trainDf)))
  return(my_list)
}

my_svm <- function(trainDf, testDf, costs = 1, gammas = 0.01, kernel = "radial") {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  best.svm <- best.tune(svm,
                        myFormula,
                        data = trainDf,
                        kernel = kernel,cost = costs,gamma = gammas)

  my_list$model <- best.svm
  my_list$preTest <- predict(best.svm, testDf)
```

```
  my_list$preTrain <- predict(best.svm, trainDf)

  return(my_list)
}

my_svm_LHL <- function(trainDf, testDf, costs = 1, gammas = 0.01, kernel = "radial") {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("LiuHanLiang ~", paste(n[!(n %in% "LiuHanLiang")], collapse
= " + ")))
  best.svm <- best.tune(svm,
                        myFormula,
                        data = trainDf,
                        kernel = kernel,cost = costs,gamma = gammas)

  my_list$model <- best.svm
  my_list$preTest <- predict(best.svm, testDf)
  my_list$preTrain <- predict(best.svm, trainDf)

  return(my_list)
}

my_lmtree <- function(trainDf, testDf, cp = 0.0001) {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  lmtree <- rpart(myFormula,
                  data = trainDf, cp = cp)
  my_list$model <- lmtree
  my_list$preTest <- predict(lmtree, testDf)
  my_list$preTrain <- predict(lmtree, trainDf)

  return(my_list)
}



my_checkout <- function(preData, RealData) {
  lenData <- length(preData)
  ME <- sum(RealData - preData)/lenData
  MAD <- sum(abs(RealData - preData))/lenData
  MSE <- sum((RealData - preData)^2)/lenData
  MPE <- sum(abs(RealData - preData)/abs(RealData))/lenData
  checkDf <- c(ME, MAD, MSE, MPE)
  names(checkDf) <- c("ME", "MAD", "MSE", "MPE")
  return(checkDf)
}

####计算因变量####

#回归的
```

```
# ComputeFunc <- function(yname, xVec) {
#   if (yname == "RON") {
#     #"x19" "x2"  "x20" "x24" "x3"  "x5"  "x7"
#     x5 <- xVec["x5"]
#     x20 <- xVec["x20"]
#     x19 <- xVec["x19"]
#     x2 <- xVec["x2"]
#     x24 <- xVec["x24"]
#     x3 <- xVec["x3"]
#     x7 <- xVec["x7"]
#        y = -0.7971 + 0.0003889*x5 +2.334*x20 - 0.02861*x2*x24 + 0.00004222*x3^2 -
0.004541*x5*x20 - 0.00074*x7*x19 + 0.0237*x19*x20
#   } else if (yname == "LiuHanLiang") {
#     #"x16" "x18" "x25" "x26" "x5"  "x9"
#     x16 <- xVec["x16"]
#     x18 <- xVec["x18"]
#     x25 <- xVec["x25"]
#     x26 <- xVec["x26"]
#     x5 <- xVec["x5"]
#     x9 <- xVec["x9"]
#     y = -17.22 + 0.05035*x5 + 0.02646*x25 - 0.00003203*x9*x18 + (6.215e-08)*x16*x26
#   }
#   return(y)
# }


#回归树的


####差分进化算法####

CFJH <- function(NP = 20, D = 29, G = 100, FFF = 0.5, CR = 0.1,
                 limitDf, nameX, yName, ComputeFunc) {

  outComeing <- list()

  NP <- NP
  D <- D
  G <- G
  FFF <- FFF
  CR <- CR
  limitDf <- limitDf[nameX, ]

  x <- matrix(0, nrow = D, ncol = NP) #初始种群
  rownames(x) <-nameX
  v <- matrix(0, nrow = D, ncol = NP) #变异种群
  rownames(v) <-nameX
  u <- matrix(0, nrow = D, ncol = NP) #选择种群
  rownames(u) <-nameX
  x <- matrix(runif(D*NP, 0, 1),
              nrow = D, ncol = NP)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
```

```
  rownames(x) <-nameX
###计算目标函数
  Ob <- c()
  for (item in 1:NP) {
    tempX <-as.data.frame(t(x[, item]))
    names(tempX) <- nameX
    Ob[item] <- predict(ComputeFunc, tempX)
  }
  traceMy <- c()
  traceMy[1] <- min(Ob)

  for (gen in 1:G) {
    for (m in 1:NP) {

      r1 <- sample(1:NP, 1)

      while (r1 == m) {
        r1 <- sample(1:NP, 1)
      }

      r2 <- sample(1:NP, 1)

      while ((r2 == m) | (r2 == r1)) {
        r2 <- sample(1:NP, 1)
      }

      r3 <- sample(1:NP, 1)

      while ((r3 == m) | (r3 == r1) | (r3 == r2)) {
        r3 <- sample(1:NP, 1)
      }
      v[, m] <- x[, r1] + FFF*(x[, r2]-x[, r3])
    }
    r <- sample(1:D, 1)
    for (n in 1:D) {
      cr <- runif(1, 0, 1)
      if ((cr <= CR) | (n == r)) {
        u[n, ] <- v[n, ]
      } else {
        u[n, ] <- x[n, ]
      }
    }

    for (n in 1:D) {
      for (m in 1:NP) {
        if (u[n, m] < limitDf[n, 1]) {
          u[n, m] <- limitDf[n, 1]
        }

        if (u[n, m] > limitDf[n, 2]) {
          u[n, m] <- limitDf[n, 2]
```

```
        }

      }
    }
    Ob1 <- c()
    for (m in 1:NP) {

      tempU <- as.data.frame(t(u[, m]))
      names(tempU) <- nameX
      Ob1[m] <- predict(ComputeFunc, tempU)
    }

    for (m in 1:NP) {
      if (Ob1[m] < Ob[m]) {
        x[, m] <- u[, m]
      }
    }

    for (m in 1:NP) {
      tempX <-as.data.frame(t(x[, m]))
      names(tempX) <- nameX
      Ob[item] <- predict(ComputeFunc, tempX)
    }
    traceMy[gen + 1] <- min(Ob)
    print(paste("第", gen, "次迭代", sep = ""))
  }

  outComeing$trace <- traceMy
  outComeing$x <- x
  outComeing$Ob <- Ob
  return(outComeing)
}


####模拟退火算法####

#没写完

# MNTH <- function(D = 7, L = 200, K = 0.998, S = 0.01, TTT = 100, YZ = 1e-8, P = 0,
#                  limitDf, nameX, yName) {
#
#   outComeing <- list()
#
#   D = D
#   L = L
#   K = K
#   S = S
#   TTT = TTT
#   YZ = YZ
#   P = P
#   limitDf <- limitDf[nameX, ]
```

```
#   PreX <- runif(D, 0, 1)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
#   preBestX <- PreX
#   PreX <- runif(D, 0, 1)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
#   BestX <- PreX
#
# }

####求最优解####

# outRonCfjh <- CFJH(NP = 20, D = 7, G = 1000, FFF = 0.5, CR = 0.1,
#                    limit_Need_ChangeName, nameX = name_Use_Ron, yName = "RON")



####network, svm, lmtree####

#network
W4_net <- my_network(dfRon_F, dfRon_F)
check_net <- my_checkout(W4_net$preTrain, dfRon_F[, length(dfRon_F)])

#svm
W4_svm <- my_svm(dfRon_F, dfRon_F)
check_svm <- my_checkout(W4_svm$preTrain, dfRon_F[, length(dfRon_F)])

W4_svm_LHL <- my_svm_LHL(dfHLL_F, dfHLL_F)


#lmtree
W4_lmtree <- my_lmtree(dfRon_F, dfRon_F)
check_lmtree <- my_checkout(W4_lmtree$preTrain, dfRon_F[, length(dfRon_F)])


#lm
W4_lm <- my_lmFunc(dfRon_F, dfRon_F, F)
check_lm <- my_checkout(W4_lm$preTrain, dfRon_F[, length(dfRon_F)])


check_Matrix <- matrix(0, 4, 4)
check_Matrix[1, ] <- check_net
check_Matrix[2, ] <- check_svm
check_Matrix[3, ] <- check_lmtree
check_Matrix[4, ] <- check_lm

colnames(check_Matrix) <- names(check_lmtree)
rownames(check_Matrix) <- c("net", "svm", "lmtree", "lm")

check_Matrix
#          ME       MAD        MSE       MPE
# net     5.196782e-03 0.1582932 0.04585707 0.1464647
# svm    -5.375041e-03 0.1261623 0.03500909 0.1237396
# lmtree  3.125705e-17 0.1051905 0.02237275 0.1000444
```

```
# lm       3.254269e-14 0.1293222 0.02835740 0.1196551

#树模型预测准确性在这里较好


####符合要求的观测分别求出最优解####

df_Up30 <- dfRon_F[which(dfRon_F$RON < 1.37*0.7), ]

for (item in 1:dim(df_Up30)[1]) {
  limit_vec <- df_Up30[item, c(9, 24)]
  print(limit_vec)
  print("----得到数值----")
  limit_Need_ChangeName[9, c(1, 2)] <- c(limit_vec[1] - 0.1, limit_vec[1] + 0.1)
  limit_Need_ChangeName[23, c(1, 2)] <- c(limit_vec[2] - 0.1, limit_vec[2] + 0.1)
  outRonCfjh <- CFJH(NP = 30, D = 29, G = 300, FFF = 0.5, CR = 0.1,
                     limit_Need_ChangeName, nameX = colnames(dfRon_F[, -length(dfRon_F)]),
                     yName = "RON", ComputeFunc = W4_svm$model)
  print("------运行完毕------")
  assign(paste("outRonCfjh", item, sep = "_"), outRonCfjh)
}

goodYlist <- c()
goodXmatrix <- matrix(0, nrow = 29, ncol = dim(df_Up30)[1])

for (item in 1:dim(df_Up30)[1]) {
  outRonC <- get(paste("outRonCfjh", item, sep = "_"))
  goodNum <- which(outRonC$Ob == min(outRonC$Ob))
  goodY <- outRonC$Ob[goodNum]
  goodX <- outRonC$x[, goodNum]
  goodYlist <- c(goodYlist, goodY)
  goodXmatrix[, item] <- goodX

}

####输出结果####
write.csv(goodYlist, "output/Optimal_Y_Value.csv")
write.csv(goodXmatrix, "output/Optimal_X_Value.csv")


plot(outRonCfjh$trace, type = "l", main = "目标函数曲线图", xlab = "迭代次数", ylab = "目
标函数值",
     ylim = c(0.85, 1.15))


min_Svm_X <- outRonCfjh$x[, which(outRonCfjh$Ob == min(outRonCfjh$Ob))]
tempSvm_X <- as.data.frame(t(min_Svm_X))
names(tempSvm_X)
##将优化后的数值带入硫含量的回归模型中

predict(W4_svm_LHL$model, tempSvm_X) #4.73434
```

```
# class(outRonCfjh$x) #matrix
#
# class(outRonCfjh$x[, 1]) #numeric
#
# as.data.frame(t(outRonCfjh$x[, 1]))
#
# pre_www <- predict(W4_lmtree$model, as.data.frame(t(outRonCfjh$x[, 1])))
#
# predict(W4_lmtree$model, dfRon_F[1:10, ])

####输出结果####

write.csv(outRonCfjh$trace, "output/W4_outcome_trace.csv")
write.csv(outRonCfjh$x, "output/W4_outcome_x.csv")
write.csv(outRonCfjh$Ob, "output/W4_outcome_Ob.csv")

####测试,读取数据####

test_W4_data02 <- read.csv("data/W4_data02.csv", header = T)
head(test_W4_data02)

lm_test02 <- my_lmFunc(test_W4_data02, test_W4_data02, lineTF = F)
summary(lm_test02$model)

########问题 5#######

####问题 4：优化####

#注意: 只有操作变量可以进入方程
#所以, 在问题 2 得到的变量中，需要把非操作变量删去
#注意, 这里的 RON 是减小值
#可以不需要方程
#再次进行第三题的操作，对新的数据集进行拟合，选取拟合效果最好的模型，但这次没有训练集，是全部样本

####导包####

library(neuralnet)
library(h2o)
library(rpart)
library(e1071)
library(caret)

####初始化环境####

h2o.init()

####读取数据####

#自变量与因变量数据
```

```
data_W4 <- read.csv("data/W4_data02.csv", header = T)
head(data_W4)
dim(data_W4) #325  31

#读取自变量限制
limit_M <- read.csv("data/limit_Val.csv", header = T)

limit_M[, 1:4]
rownames(limit_M) <- limit_M$X
limit_M <- limit_M[, -1]
limit_M_Mat_t <- t(limit_M)
head(limit_M_Mat_t)
dim(limit_M_Mat_t) #354 2

###为原材料添加新的限制####
limit_new_df <- data.frame(Down = c(0, 0), Up = c(0, 0))
rownames(limit_new_df) <- c("Raw_RON", "JiaoTan")

limit_M_Mat_t <- rbind(limit_M_Mat_t, limit_new_df)
dim(limit_M_Mat_t) #356 2

dim(data_W4)
head(data_W4)

#改变名字的 data_W4

nameChange_data_W4 <- data_W4
colnames(nameChange_data_W4) <- c(paste("x", seq(1:(dim(nameChange_data_W4)[2] - 2)), sep =
""),
                                  "RON", "LiuHanLiang")

head(nameChange_data_W4)

#RON 和 LiuHanLiang 矩阵
dfRon_F <- nameChange_data_W4[, -length(nameChange_data_W4)]
dfHLL_F <- nameChange_data_W4[, -(length(nameChange_data_W4) - 1)]

head(dfRon_F)
head(dfHLL_F)



####数据准备####

##中心化标准化
scale_data_W4 <- as.data.frame(scale(data_W4, center = T, scale = T))
head(scale_data_W4)
dim(scale_data_W4)

###构建数据集
##RON 作为因变量的数据集
```

```
dfRon <- scale_data_W4[, -length(scale_data_W4)]
head(dfRon)
dim(dfRon)
colName_Ron <- colnames(dfRon)

#更改自变量名
colnames(dfRon) <- c(paste("x", seq(1:(dim(dfRon)[2] - 1)), sep = ""), "RON")
head(dfRon)

##LiuHanLiang 作为因变量的数据集
dfHLL <- scale_data_W4[, -(length(scale_data_W4) - 1)]
head(dfHLL)
dim(dfHLL)

#提取变量名
colName_HLL <- colnames(dfHLL)

#更改自变量名
colnames(dfHLL) <- c(paste("x", seq(1:(dim(dfHLL)[2] - 1)), sep = ""), "LiuHanLiang")
head(dfHLL)

#需要的自变量名登记
name_ZiVal <- colName_HLL[-length(colName_HLL)]
length(name_ZiVal) #27

#全部变量名登记
name_AllVal <- rownames(limit_M_Mat_t)
length(name_AllVal) #354

#判断需要的自变量位置及提取限制
ZiVal <- c()
for (item in 1:length(name_ZiVal)) {
  temp <- name_ZiVal[item]
  num <- which(temp == name_AllVal)
  ZiVal <- c(ZiVal, num)
}

head(name_AllVal[ZiVal])
head(name_ZiVal)
#OK!!!

#需要的限制数据集
limit_Need <- as.data.frame(limit_M_Mat_t[ZiVal, ])
rownames(limit_Need)

#对限制数据集进行改名
limit_Need_ChangeName <- limit_Need
rownames(limit_Need_ChangeName) <- paste("x", seq(1:(dim(limit_Need_ChangeName)[1])), sep =
"")

####方法区####
```

```
my_lmFunc <- function(trainDf, testDf, lineTF = T, yName) {
  my_list <- list()
  if (lineTF == T) {
    n <- names(trainDf)
    myFormula <- as.formula(paste(yName, "~",
                                  paste(n[!(n %in% yName)],
                                        collapse = " + ")))
    lmTest <- lm(myFormula, data = trainDf)
  } else if (lineTF == F) {
    ##lm 简单回归(T)
    lm_step1 <- my_lmFunc(trainDf, testDf, lineTF = T, yName = yName)
    #提取 summary 中的 p 值
    lm_p_value <- summary(lm_step1$model)$coefficients[, 4]
    #取出哪些变量显著
    remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])
    nameTemp <- remarkableValue
    print(nameTemp)
    nameSplicing <- c()
    #nameMult <- c()
    for (i in c(1 : length(nameTemp))) {
      # nameMult <- c(nameMult,
      #               paste("I(", nameTemp[i], "^2)", sep = ""))
      for (j in c(i : length(nameTemp))) {
        nameSplicing <- c(nameSplicing,
                          paste("I(", nameTemp[i], "*", nameTemp[j], ")",
                                sep = ""))
      }
    }
    #print(nameSplicing)
    myFormula <- as.formula(paste(yName, "~",
                                  paste(nameTemp, collapse = " + "),
                                  "+",
                                  paste(nameSplicing, collapse = " + ")))
    #print(myFormula)
    lmTest <- lm(myFormula, data = trainDf)
  }
  #模型
  my_list$model <- lmTest
  #测试集预测值
  my_list$preTest <- predict(lmTest, testDf)
  #训练集预测值
  my_list$preTrain <- predict(lmTest, trainDf)
  return(my_list)
}

####多项式回归####
RON_lm <- my_lmFunc(dfRon, dfRon,
                    lineTF = F, yName = "RON")
summary(RON_lm$model) # 0.2547

LHL_lm <- my_lmFunc(dfHLL, dfHLL,
```

```
                    lineTF = F, yName = "LiuHanLiang")
summary(LHL_lm$model) #0.2929

Ron_value <- summary(RON_lm$model)$coefficients[, 4]
remarkable_Ron_value <- names(Ron_value[which(Ron_value < 0.05)])
length(remarkable_Ron_value) #7
Ron_string <- paste(remarkable_Ron_value, collapse = " + ")
Ron_Formula <- as.formula(paste("RON ~", Ron_string))
# RON ~ x5 + x20 + I(x2 * x24) + I(x3 * x3) +
#   I(x5 * x20) + I(x7 * x19) + I(x19 * x20)




####自定义函数####

my_lmFunc <- function(trainDf, testDf, lineTF = T) {
  my_list <- list()
  if (lineTF == T) {
    n <- names(trainDf)
    myFormula <- as.formula(paste("RON ~",
                                  paste(n[!(n %in% "RON")],
                                        collapse = " + ")))
    lmTest <- lm(myFormula, data = trainDf)
  } else if (lineTF == F) {
    ##lm 简单回归(T)
    lm_step1 <- my_lmFunc(trainDf, testDf, lineTF = T)
    #提取 summary 中的 p 值
    lm_p_value <- summary(lm_step1$model)$coefficients[, 4]
    #取出哪些变量显著
    remarkableValue <- names(lm_p_value[which(lm_p_value < 0.1)])

    #添加对截距项处理
    remarkableValue <- remarkableValue[!(remarkableValue %in% "(Intercept)")]

    nameTemp <- remarkableValue
    print(remarkableValue)
    nameSplicing <- c()
    #nameMult <- c()
    for (i in c(1 : length(nameTemp))) {
      # nameMult <- c(nameMult,
      #               paste("I(", nameTemp[i], "^2)", sep = ""))
      for (j in c(i : length(nameTemp))) {
        print(j)
        nameSplicing <- c(nameSplicing,
                          paste("I(", nameTemp[i], "*", nameTemp[j], ")",
                                sep = ""))
      }
    }
    #print(nameSplicing)
    myFormula <- as.formula(paste("RON ~",
```

```
                                  paste(nameTemp, collapse = " + "),
                                  "+",
                                  paste(nameSplicing, collapse = " + ")))
    #print(myFormula)
    lmTest <- lm(myFormula, data = trainDf)
  }
  #模型
  my_list$model <- lmTest
  #测试集预测值
  my_list$preTest <- predict(lmTest, testDf)
  #训练集预测值
  my_list$preTrain <- predict(lmTest, trainDf)
  return(my_list)
}


my_svm <- function(trainDf, testDf, costs = 1, gammas = 0.01, kernel = "radial") {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("RON ~", paste(n[!(n %in% "RON")], collapse = " + ")))
  best.svm <- best.tune(svm,
                        myFormula,
                        data = trainDf,
                        kernel = kernel,cost = costs,gamma = gammas)

  my_list$model <- best.svm
  my_list$preTest <- predict(best.svm, testDf)
  my_list$preTrain <- predict(best.svm, trainDf)

  return(my_list)
}

my_svm_LHL <- function(trainDf, testDf, costs = 1, gammas = 0.01, kernel = "radial") {
  my_list <- list()
  n = names(trainDf)
  myFormula <- as.formula(paste("LiuHanLiang ~", paste(n[!(n %in% "LiuHanLiang")], collapse
= " + ")))
  best.svm <- best.tune(svm,
                        myFormula,
                        data = trainDf,
                        kernel = kernel,cost = costs,gamma = gammas)

  my_list$model <- best.svm
  my_list$preTest <- predict(best.svm, testDf)
  my_list$preTrain <- predict(best.svm, trainDf)

  return(my_list)
}
```

```
my_checkout <- function(preData, RealData) {
  lenData <- length(preData)
  ME <- sum(RealData - preData)/lenData
  MAD <- sum(abs(RealData - preData))/lenData
  MSE <- sum((RealData - preData)^2)/lenData
  MPE <- sum(abs(RealData - preData)/abs(RealData))/lenData
  checkDf <- c(ME, MAD, MSE, MPE)
  names(checkDf) <- c("ME", "MAD", "MSE", "MPE")
  return(checkDf)
}

####计算因变量####


#回归树的


####差分进化算法####

CFJH <- function(NP = 20, D = 29, G = 100, FFF = 0.5, CR = 0.1,
                 limitDf, nameX, yName, ComputeFunc) {

  outComeing <- list()

  NP <- NP
  D <- D
  G <- G
  FFF <- FFF
  CR <- CR
  limitDf <- limitDf[nameX, ]

  x <- matrix(0, nrow = D, ncol = NP) #初始种群
  rownames(x) <-nameX
  v <- matrix(0, nrow = D, ncol = NP) #变异种群
  rownames(v) <-nameX
  u <- matrix(0, nrow = D, ncol = NP) #选择种群
  rownames(u) <-nameX
  x <- matrix(runif(D*NP, 0, 1),
              nrow = D, ncol = NP)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
  rownames(x) <-nameX
  ###计算目标函数
  Ob <- c()
  for (item in 1:NP) {
    tempX <-as.data.frame(t(x[, item]))
    names(tempX) <- nameX
    Ob[item] <- predict(ComputeFunc, tempX)
  }
  traceMy <- c()
  traceMy[1] <- min(Ob)

  for (gen in 1:G) {
```

```
for (m in 1:NP) {

  r1 <- sample(1:NP, 1)

  while (r1 == m) {
    r1 <- sample(1:NP, 1)
  }

  r2 <- sample(1:NP, 1)

  while ((r2 == m) | (r2 == r1)) {
    r2 <- sample(1:NP, 1)
  }

  r3 <- sample(1:NP, 1)

  while ((r3 == m) | (r3 == r1) | (r3 == r2)) {
    r3 <- sample(1:NP, 1)
  }
  v[, m] <- x[, r1] + FFF*(x[, r2]-x[, r3])
}
r <- sample(1:D, 1)
for (n in 1:D) {
  cr <- runif(1, 0, 1)
  if ((cr <= CR) | (n == r)) {
    u[n, ] <- v[n, ]
  } else {
    u[n, ] <- x[n, ]
  }
}

for (n in 1:D) {
  for (m in 1:NP) {
    if (u[n, m] < limitDf[n, 1]) {
      u[n, m] <- limitDf[n, 1]
    }

    if (u[n, m] > limitDf[n, 2]) {
      u[n, m] <- limitDf[n, 2]
    }

  }
}
Ob1 <- c()
for (m in 1:NP) {

  tempU <- as.data.frame(t(u[, m]))
  names(tempU) <- nameX
  Ob1[m] <- predict(ComputeFunc, tempU)
}
```

```
    for (m in 1:NP) {
      if (Ob1[m] < Ob[m]) {
        x[, m] <- u[, m]
      }
    }

    for (m in 1:NP) {
      tempX <-as.data.frame(t(x[, m]))
      names(tempX) <- nameX
      Ob[item] <- predict(ComputeFunc, tempX)
    }
    traceMy[gen + 1] <- min(Ob)
    print(paste("第", gen, "次迭代", sep = ""))
  }

  outComeing$trace <- traceMy
  outComeing$x <- x
  outComeing$Ob <- Ob
  return(outComeing)
}


####模拟退火算法####

#没写完

# MNTH <- function(D = 7, L = 200, K = 0.998, S = 0.01, TTT = 100, YZ = 1e-8, P = 0,
#                  limitDf, nameX, yName) {
#
#   outComeing <- list()
#
#   D = D
#   L = L
#   K = K
#   S = S
#   TTT = TTT
#   YZ = YZ
#   P = P
#   limitDf <- limitDf[nameX, ]
#   PreX <- runif(D, 0, 1)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
#   preBestX <- PreX
#   PreX <- runif(D, 0, 1)*(limitDf[, 2] - limitDf[, 1]) + limitDf[, 1]
#   BestX <- PreX
#
# }

####求最优解####



#svm
```

```
W4_svm <- my_svm(dfRon_F, dfRon_F)
check_svm <- my_checkout(W4_svm$preTrain, dfRon_F[, length(dfRon_F)])

W4_svm_LHL <- my_svm_LHL(dfHLL_F, dfHLL_F)



#lm
W4_lm <- my_lmFunc(dfRon_F, dfRon_F, F)
check_lm <- my_checkout(W4_lm$preTrain, dfRon_F[, length(dfRon_F)])
summary(W4_lm$model) #X7 X8 X20
names(summary(W4_lm$model))

####输出结果##
write.csv(summary(W4_lm$model)$coefficients, "output/W4_lm_outcone.csv")

###交叉验证###
# check_Matrix <- matrix(0, 4, 4)
# check_Matrix[1, ] <- check_net
# check_Matrix[2, ] <- check_svm
# check_Matrix[3, ] <- check_lmtree
# check_Matrix[4, ] <- check_lm

# colnames(check_Matrix) <- names(check_lmtree)
# rownames(check_Matrix) <- c("net", "svm", "lmtree", "lm")
#
# check_Matrix
#         ME       MAD        MSE       MPE
# net     5.196782e-03 0.1582932 0.04585707 0.1464647
# svm    -5.375041e-03 0.1261623 0.03500909 0.1237396
# lmtree  3.125705e-17 0.1051905 0.02237275 0.1000444
# lm      3.254269e-14 0.1293222 0.02835740 0.1196551

#树模型预测准确性在这里较好


####符合要求的观测分别求出最优解####

df_Up30 <- dfRon_F[which(dfRon_F$RON < 1.37*0.7), ]

for (item in 1:dim(df_Up30)[1]) {
  limit_vec <- df_Up30[item, c(9, 24)]
  print(limit_vec)
  print("----得到数值----")
  limit_Need_ChangeName[9, c(1, 2)] <- c(limit_vec[1] - 0.1, limit_vec[1] + 0.1)
  limit_Need_ChangeName[23, c(1, 2)] <- c(limit_vec[2] - 0.1, limit_vec[2] + 0.1)
  outRonCfjh <- CFJH(NP = 40, D = 29, G = 500, FFF = 0.5, CR = 0.1,
                     limit_Need_ChangeName, nameX = colnames(dfRon_F[, -length(dfRon_F)]),
                     yName = "RON", ComputeFunc = W4_svm$model)
  print("------运行完毕------")
  assign(paste("outRonCfjh", item, sep = "_"), outRonCfjh)
```

```
}

goodYlist <- c()
goodXmatrix <- matrix(0, nrow = 29, ncol = dim(df_Up30)[1])

for (item in 1:dim(df_Up30)[1]) {
  outRonC <- get(paste("outRonCfjh", item, sep = "_"))
  goodNum <- which(outRonC$Ob == min(outRonC$Ob))
  goodY <- outRonC$Ob[goodNum]
  goodX <- outRonC$x[, goodNum]
  goodYlist <- c(goodYlist, goodY)
  goodXmatrix[, item] <- goodX

}

###计算硫含量

#导入之前计算的最优自变量数据
readDoodXmatrix <- read.csv("output/Optimal_X_Value.csv")

head(readDoodXmatrix)
dim(readDoodXmatrix) #29 18
readDoodXmatrix <- readDoodXmatrix[, -1]
dim(readDoodXmatrix) #29 17
rownames(readDoodXmatrix) <- paste("x", 1:29, sep = "")

LHL_list <- c()

for (item in 1:dim(readDoodXmatrix)[2]) {
  min_XX <- readDoodXmatrix[, item]
  names(min_XX) <- paste("x", 1:29, sep = "")
  tempSvm_XX <- as.data.frame(t(min_XX))
  LHL_c <- predict(W4_svm_LHL$model, tempSvm_XX)
  LHL_list <- c(LHL_list, LHL_c)
}

##含硫量输出
write.csv(LHL_list, "output/HanLiuLiang_Limit.csv")

####输出结果####
write.csv(goodYlist, "output/Optimal_Y_Value.csv")
write.csv(goodXmatrix, "output/Optimal_X_Value.csv")
write.csv(df_Up30$RON, "output/RealValueRon_Y.csv")

sum(goodYlist < df_Up30$RON)/length(goodYlist)

####133 号####
limit_133 <- dfRon_F[133, c(9, 24)]
limit_Need_ChangeName[9, c(1, 2)] <- c(limit_133[1] - 0.1, limit_133[1] + 0.1)
limit_Need_ChangeName[23, c(1, 2)] <- c(limit_133[2] - 0.1, limit_133[2] + 0.1)
outRonCfjh133 <- CFJH(NP = 30, D = 29, G = 600, FFF = 0.5, CR = 0.1,
```

```
                      limit_Need_ChangeName, nameX = colnames(dfRon_F[, -length(dfRon_F)]),
                      yName = "RON", ComputeFunc = W4_svm$model)

plot(outRonCfjh133$trace, type = "l", main = "目标函数曲线图", xlab = "迭代次数", ylab = "
目标函数值",
     ylim = c(0.85, 1.15))

min(outRonCfjh133$Ob) #0.8785725

min_Svm_X <- outRonCfjh133$x[, which(outRonCfjh133$Ob == min(outRonCfjh133$Ob))]
tempSvm_X <- as.data.frame(t(min_Svm_X))
names(tempSvm_X)
##将优化后的数值带入硫含量的回归模型中

predict(W4_svm_LHL$model, tempSvm_X) #4.350393

####输出结果####

write.csv(outRonCfjh133$trace, "output/W4_outcome_trace133.csv")
write.csv(outRonCfjh133$x, "output/W4_outcome_x133.csv")
write.csv(outRonCfjh133$Ob, "output/W4_outcome_Ob133.csv")


###

names(data_W4)[c(7, 8, 20)]
head(data_W4)


plot_ron_line <- function(indexNum = 7, bestX, modeling) {
  if (indexNum == 7) {
    step_L <- 1
  } else if (indexNum == 8) {
    step_L <- 0.1
  } else {
    step_L <- 1
  }
    baseNum <- bestX[indexNum]
    stepList <- baseNum + c(-5:5)*step_L
    step_y_list <- c()
    for (item in stepList) {
      bestX[indexNum] <- item
      temp_X <- as.data.frame(t(bestX))
      pre_y <- predict(modeling, temp_X)
      step_y_list <- c(step_y_list, pre_y)
    }
    return(step_y_list)
}

y_last_7 <-plot_ron_line(7, min_Svm_X, W4_svm$model)
y_last_8 <-plot_ron_line(8, min_Svm_X, W4_svm$model)
```

```
y_last_20 <-plot_ron_line(20, min_Svm_X, W4_svm$model)

y_last_df <- data.frame(y7 = y_last_7, y8 = y_last_8, y20 = y_last_20)
write.csv(y_last_df, "output/y_last_df.csv")
```