基于遗传算法和梯度提升决策树的乙醇制备 C4 烯烃模型

摘　要

本文基于题目给出的不同催化剂组合在一定温度内制备 C4 烯烃的性能数据表，对数据进行了处理与分析。利用了梯度提升决策树算法研究了不同催化剂组合、装料方式以及温度对于乙醇转化率和 C4 烯烃选择性的影响，利用遗传算法研究了使 C4 烯烃收率尽可能高的催化剂组合与温度，并针对题目给出的数据和建立的模型，设计了 5 次新的实验。

针对**问题 1.** 根据附件 1 中的性能数据，我们分别对温度与乙醇转化率和 C4 烯烃选择性大小的相关系数作了计算，并进行了显著性检验，再通过曲线拟合进一步确定关系，得出随着反应温度的提高，乙醇的转化率和 C4 烯烃的选择性基本呈现上升的趋势，呈显著正相关。对于附件 2 给定催化剂组合在一次实验不同时间的测试结果，运用线性回归和多项式回归的方法，得出乙醇转化率以及碳数为 4-12 的脂肪醇与时间呈负相关，乙烯转化率和乙醛选择性与时间呈正相关，C4 烯烃以及甲基苯甲醛和甲基苯甲醇可以认为是稳定的。

针对**问题 2.** 我们选择梯度提升决策树模型进行回归分析，包括了 XGBoost 算法以及 lightGBM 算法。以题目中的装料方式，催化剂数据和温度数据作为自变量，以乙醇转化率和 C4 烯烃选择性分别作为因变量进行了回归分析。并将实验数据按照 9：1 的比例划分为训练集和测试集，运用网格搜索选择最优的超参数得到 XGBoost 和 lightGBM 模型后，再通过测试集和预测值的R^2以及 *MSE* 选择更优的模型，最后根据训练完成的模型的 feature_importance 参数获得不同种类的催化剂以及温度对于乙醇转化率和 C4 烯烃选择性的影响。最终，针对催化剂和温度对于乙醇转化率的影响，选择了 XGBoost 模型；针对催化剂和温度对于 C4 烯烃选择性的影响，选择了 lightGBM 模型。结论是对于乙醇转化率来说，Co 含量、HAP 质量和乙醇浓度这三个变量的影响近似，温度对于乙醇转化率有决定性的影响，装料方式的影响并不大。对于 C4 烯烃选择性来说，温度这个参数有一定影响，但不具有决定性作用，Co/SiO2 的质量对于 C4 烯烃选择性具有显著的影响。

针对**问题 3.** 在问题 2.的基础上，我们基于建立好的 XGBoost 模型以及 lightGBM 模型，通过遗传算法，寻找最佳的催化剂组合及温度，使相同实验条件下 C4 烯烃收率尽可能高。通过选取不同的先验信息分别输入模型，在其附近确定最优点，并最终确定了最优催化剂组合与温度条件。

针对**问题 4.** 在得到前三问的结果后，针对问题 3. 生成的最优催化剂组合与温度条件中 Co/SiO2 质量、HAP 质量已经超出题给的数据范围，增加两组验证性实验；针对附件二增加一组实验判断 350℃时给定的催化剂组合最终 C4 烯烃收率是否稳定；针对样本数据中范围不够大的问题，增加两组实验，探究更高温度对乙醇转化率、C4 烯烃的选择性的影响以及更长反应时间对于乙烯和乙醛选择性的影响，使得本模型具有更广的应用价值。

关键词：曲线拟合　多项式回归　遗传算法　梯度提升决策树　烯烃制备

一、 问题重述

1.1 问题背景

在化工产品的制造和医学药品的生产过程中，常常使用乙醇作为 C4 烯烃生产与制备的原材料。在这一过程中，影响 C4 烯烃的选择性和 C4 烯烃收率的要素主要包括催化剂的组合与温度。因此，为了探究乙醇催化偶合制备 C4 烯烃的最适工艺条件，对催化剂组合的选择与设计具有重要的研究价值。某一化工实验室在不同温度和不同催化剂组合的条件下做了一系列的实验，需要解决以下问题：

1.2 问题的提出

问题 1. 针对附件 1 中不同的催化剂组合，研究温度对乙醇转化率、C4 烯烃的选择性的影响；针对附件 2 中给定的催化剂组合在 350℃不同时间下的测试结果进行分析。

问题 2. 研究不同的温度以及催化剂的组合对于乙醇转化率以及 C4 烯烃的选择性大小的影响。

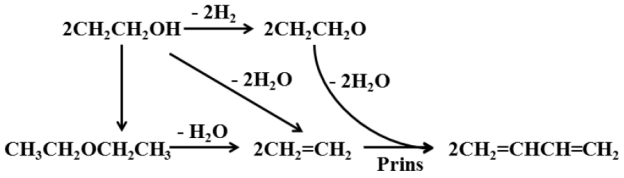
问题 3. 研究在同样的实验条件下如何选择温度与催化剂的组合，使得 C4 烯烃吸收率尽可能高；如果温度低于 350℃，又应该怎样选择温度与催化剂的组合，使得 C4 烯烃吸收率尽可能高。

问题 4. 若可以增加 5 次实验，应该怎样进行设计。

二、 问题分析

2.1 问题一的分析

问题一中的第一问首先对不同催化剂组合下温度分别与乙醇转化率和 C4 烯烃选择性大小进行显著性检验，并计算相关系数的大小。接着通过对附件中给出的数据进行曲线拟合的方法进一步确定关系，在拟合的过程中，比较了多种拟合函数拟合的效果；问题一中的第二问运用线性回归和多项式回归的方法，分别将乙醇，碳数为 4-12 的脂肪醇，乙烯和乙醛作为因变量，将时间作为自变量进行回归分析。回归的模型次数从一次到四次，根据曲线的趋势判断反应产物的稳定性，选择最优的模型。最后，我们发现附件中各产物以及中间产物的关系符合普林斯机理：



图片 1

2.2 问题二的分析

问题二选择梯度上升树的模型进行回归分析，具体运用了 XGBoost 算法以及 lightGBM 算法，将题目提供的装料方式，催化剂数据和温度数据作为自变量，将乙醇转化率和 C4 烯烃选择性分别作为因变量进行了回归分析。将实验数据按照 9：1 的比例划分为训练集和测试集，首先运用网格搜索选择最优的超参数得到 XGBoost 和 lightGBM 模型。再通过测试集和预测值的R^2以及 MSE 选择更优的模型。最后通过获得训练完的模型的 feature_importance 参数获得不同种类的催化剂以及温度对于乙醇转化率和 C4 烯烃选择性的影响。

2.3 问题三的分析

在问题二中，我们建立了基于 XGBoost 以及 lightGBM 模型的乙醇转化率以及 C4 烯烃选择性预测模型。在问题三我们基于该模型，通过遗传算法，寻找最佳的催化剂组合及温度，使相同

实验条件下 C4 烯烃收率尽可能高。通过选取不同的先验信息分别输入模型，在其附近确定最优点。

2.4 问题四的分析

在得到前三问的结果后，针对问题三生成的最优催化剂组合与温度中的 Co/SiO2 质量(mg)、HAP 质量(mg)已经超出附件 1 所给出的数据范围，增加两组验证性实验；同时，针对附件二增加一组实验判断 350 度时给定的催化剂组合最终 C4 烯烃收率是否稳定，使得本模型的建立更加完备；针对样本数据中范围不够大的问题，增加两组实验，探究更高温度对乙醇转化率、C4 烯烃的选择性的影响以及更长反应时间对于乙烯和乙醛选择性的影响，使得本模型具有更广的应用范围。

三、 模型假设

1. 假设在解决问题 2.和问题 3.时中间产物对模型的结果不造成影响。
2. 假设实验中是用较为精密的仪器对实验数据进行的测量，即数据中大多为正常值，异常值只占少数。
3. 假设各项性能数据表中各中间产物的选择性与乙醇的转化率具备相关性。
4. 假设各个催化剂组合的数据均服从正态分布。

四、 符号说明

符号	说明
R^2	拟合优度
SSE	误差平方和
MSE	均方误差
feature_importance	特征重要性
P(k)	第 k 个种群
M(k)	配对池

五、 模型的建立与求解

5.1 问题一模型的建立与求解

模型的建立

5.1.1 显著性检验与相关性

皮尔逊相关系数的使用条件：

1. 各个催化剂组合的数据均服从正态分布。
2. 不存在异常值
3. 各组样本之间是独立抽样的

皮尔逊相关系数：

皮尔逊相关系数主要用于衡量两个变量 X 与 Y 之间的相关程度，总体相关系数计算方法如下：

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{1}$$

即皮尔逊相关系数等于两个变量之间的协方差和标准差的比值。皮尔逊相关系数的值介于-1

到 1 之间，若两个变量间服从线性关系，皮尔逊相关系数的值接近 1 说明两变量有较强的正相关性，接近-1 说明两变量有较强的负相关性。

皮尔逊相关系数的 p 值：

当两变量的皮尔逊相关系数的绝对值较大时，他们之前的相关关系很有可能是由偶然因素引起的，因此需要对相关关系的显著性水平进行判断。

这里使用假设检验进行判断，原假设与备择假设分别为：

H_0：两变量有线性关系

H_1：两变量无线性关系

假设 H_0 成立，计算无相关性的概率 P。若 P 很小，则不存在相关性的概率很小，我们就可以拒绝 H_0。一般 p<0.05 即可拒绝原假设。

5.1.2 曲线拟合算法简介

已知$\boldsymbol{R}$中若干个不同的点x_i，每个点x_i对应一个数值y_i，且这些点是实测的或已经得到的，作一条指定类型的曲线，使得该曲线能在一定的条件或意义下逼近这一系列数据，这样的方法称为曲线的拟合。对于本题来说，已知 250℃、275℃、300℃、325℃、350℃下不同催化剂组合对应的乙醇转化率和 C4 烯烃的选择性大小，因此可以通过建立拟合函数的方法分析温度对乙醇转化率、C4 烯烃的选择性的影响。

5.1.3 确定本题拟合函数

从附件得到的散点图可以先假设其函数表达式为$y = kx + b$，也可假设为二次多项式或者其他多项式，先以一次多项式为例。

5.1.4 求解最小二乘法

由最小二乘法的定义可知：

$$\hat{y_i} = kx_i + b(\text{一次多项式}) \tag{2}$$

$$L = arg\min_{k,b}(\sum_{i=1}^{n}(y_i - kx_i - b)^2) \tag{3}$$

为了找出一组k, b使得L最小，

$$\begin{cases}\dfrac{\partial L}{\partial k} = 0\\ \dfrac{\partial L}{\partial b} = 0\end{cases} \tag{4}$$

得到：

$$\hat{k} = \frac{n\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}y_i\sum_{i=1}^{n}x_i}{n\sum_{i=1}^{n}x_i^2 - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}x_i} \tag{5}$$

$$\hat{b} = \frac{\sum_{i=1}^{n}x_i^2\sum_{i=1}^{n}y_i - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}x_iy_i}{n\sum_{i=1}^{n}x_i^2 - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}x_i} \tag{6}$$

当拟合函数变换时：

$$\hat{y_i} = ax_i^2 + bx_i + c(\text{二次多项式}) \tag{7}$$

$$\hat{y_i} = ax_i^3 + bx_i^2 + cx_i + d(\text{三次多项式}) \tag{8}$$

$$\hat{y_i} = ae^{-(\frac{(x-b)}{c})^2}(\text{高斯多项式}) \tag{9}$$

$$\hat{y_i} = acoswx + bsinwx + c(\text{傅里叶多项式}) \tag{10}$$

$$\hat{y}_i = ae^{bx}(\text{指数多项式}) \tag{11}$$

模型的求解

5.1.5 运用 MATLAB 求解

调用 MATLAB 中的 CFTOOL 工具箱，可以实现多种拟合算法，确保拟合结果精准、可靠。

	乙醇转化率					C4烯烃选择性(%)				
催化剂	一次多项式	二次多项式	三次多项式	指数多项式	高斯多项式	二次多项式	三次多项式	指数多项式	高斯多项式	傅里叶多项式
A1	SSE: 50.51 R-square: 0.9321	SSE: 15.08 R-square: 0.9797	SSE: 10.09 R-square: 0.9864	SSE: 14.97 R-square: 0.9799	SSE: 13.64 R-square: 0.9817	SSE: 15.9 R-square:	/	SSE: 46.94 R-square: 0.752	SSE: 12.15 R-square:	
A2	SSE: 27.67 R-square: 0.99	SSE: 24.69 R-square: 0.9911	SSE: 1.949 R-square: 0.9993	SSE: 243.5 R-square: 0.9122	SSE: 8.98 R-square: 0.9968	SSE: 7.239 R-square: 0.9803	SSE: 4.046 R-square: 0.989	/	/	SSE: 3.147 R-square: 0.9914
A3	SSE: 194.3 R-square: 0.9643	SSE: 183.3 R-square: 0.9663	SSE: 69.12 R-square: 0.9873	SSE: 601.9 R-square: 0.8893	SSE: 55.15 R-square: 0.9899	SSE: 100.1 R-square: 0.9551	SSE: 4.51 R-square: 0.998	SSE: 477.7 R-square: 0.7857	SSE: 7.808 R-square: 0.9965	
A4	SSE: 24.62 R-square: 0.995	SSE: 20.41 R-square: 0.9959	SSE: 6.286 R-square: 0.9987	SSE: 346.6 R-square: 0.9301	SSE: 24.15 R-square: 0.9951	SSE: 19.17 R-square: 0.9765	SSE: 1.237 R-square: 0.9985	SSE: 27.23 R-square: 0.9667	SSE: 23.13 R-square: 0.9717	SSE: 0.8612 R-square: 0.9989
A5	SSE: 352.6 R-square: 0.873	SSE: 16.57 R-square: 0.994	SSE: 15.42 R-square: 0.9944	/	/	SSE: 7.738 R-square: 0.9905	SSE: 0.01072 R-square: 1	SSE: 7.894 R-square: 0.9904	SSE: 5.771 R-square: 0.9929	
A6	SSE: 122.2 R-square: 0.9675	SSE: 53.02 R-square: 0.9859	SSE: 4.513 R-square: 0.9988	SSE: 122.7 R-square: 0.9674	SSE: 30.6 R-square: 0.9919	SSE: 41.5 R-square:	SSE: 0.0532 R-square:	SSE: 30.58 R-square: 0.9598	/	
A7	SSE: 2.584 R-square: 0.9987	SSE: 0.7034 R-square: 0.9997	SSE: 0.5757 R-square: 0.9997	/	/	SSE: 0.1888 R-square:	SSE: 0.02382 R-square: 1	SSE: 1.722 R-square: 0.9968	SSE: 1.613 R-square:	

表 1（完整数据见附件）

由总体平方和、误差平方和、回归平方和、拟合优度的定义可知：

$$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{12}$$

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

$$SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 \tag{14}$$

$$R^2 = \frac{SSR}{SST} \tag{15}$$

当拟合函数为线性函数时，可通过R^2的大小来判断拟合的好坏，R^2越靠近于 1，说明误差平方和SSE越接近于 0，误差越小，拟合效果越好。对于本题建立模型时，由于引入了非线性函数进行曲线拟合，因此，在比较拟合的优劣时，首先比较SSE的大小，SSE越小，拟合的程度越好，当SSE接近或者差别不大或者需要比较的拟合函数为线性函数时，通过R^2的大小作进一步的判断，选择R^2靠近于 1 的拟合函数。最后，还要考虑拟合函数的形式是否简洁，避免拟合函数过于复杂，同时避免过拟合等现象。

在综合了对以上因素的考虑后，我们选出了附件 1 中不同催化剂的组合，乙醇转化率、C4 烯烃选择

性与温度关系的最适拟合函数，同时我们还分析得到了各组乙醇转化率、C4 烯烃选择性与温度的显著性与相关性：

催化剂组合	温度与乙醇转化率的相关系数	温度与C4烯烃选择性的相关系数	催化剂组合	温度与乙醇转化率的相关系数	温度与C4烯烃选择性的相关系数
A1	0.965**	0.887*	A12	0.963**	0.983**
A2	0.995**	0.914*	A13	0.936*	0.988**
A3	0.982**	0.955**	A14	0.964**	0.959**
A4	0.969**	0.851*	B1	0.962**	0.986**
A5	0.967**	0.985**	B2	0.929*	0.985**
A6	0.978**	0.950**	B3	0.890*	0.971**
A7	0.999**	0.968**	B4	0.899*	0.895*
A8	0.977**	0.992**	B5	0.912*	0.978**
A9	0.921*	0.997*	B6	0.940**	0.982**
A10	0.923*	0.861	B7	0.936**	0.994**
A11	0.903*	0.989**			

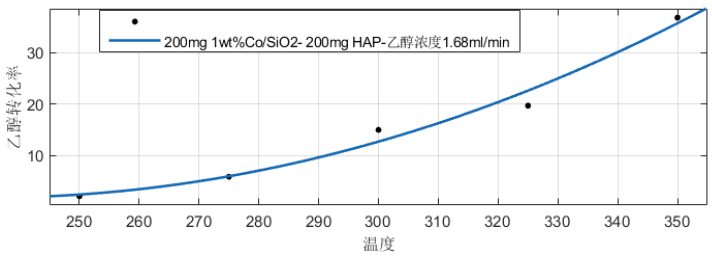
表 2

** 在 0.01 级别(双尾)，相关性显著。

* 在 0.05 级别(双尾)，相关性显著。

1.对于 200mg 1wt%Co/SiO2- 200mg HAP-乙醇浓度 1.68ml/min:

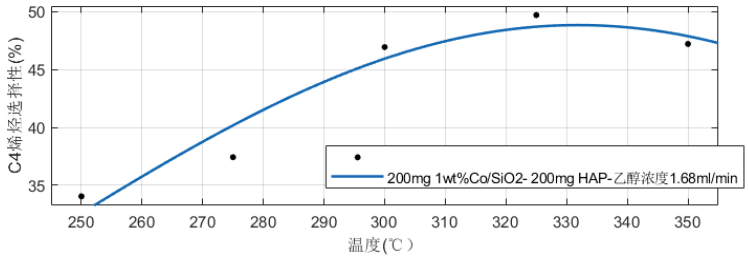
（1）乙醇转化率与温度：采用二次多项式拟合



Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.002545 (-0.002508, 0.007598)
p2 = -1.194 (-4.229, 1.841)
p3 = 141.8 (-308.9, 592.5)

图片 2

（2）C4 烯烃选择性与温度：采用高斯多项式拟合



General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 48.84 (42.11, 55.57)
b1 = 331.9 (273.8, 389.9)
c1 = 128.8 (12.53, 245)

图片 3

相关性

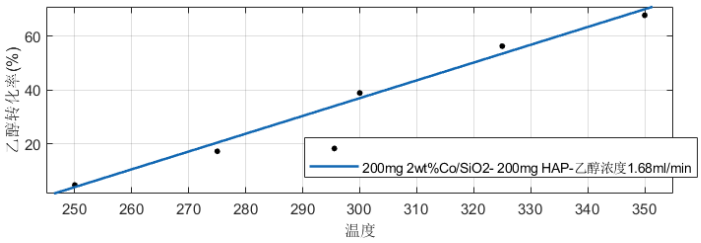
		温度	乙醇转化率(%)	乙烯选择性(%)	C4烯烃选择性(%)	乙醛选择性(%)	碳数为4-12脂肪醇选择性(%)	甲基苯甲醛和甲基苯甲醇选择性(%)	其他生成物的选择性(%)
温度	皮尔逊相关性	1	.965**	.943*	.887*	.928*	-.963**	.900*	.538
	Sig.（双尾）		.008	.016	.045	.023	.009	.037	.349
	个案数	5	5	5	5	5	5	5	5

表 3

同时，由皮尔逊相关系数和 *Sig* 的大小可知乙醇转化率与温度呈显著正相关，C_4 烯烃选择性与温度呈显著正相关，且乙醇转化率更加显著。

2.对于 200mg 2wt%Co/SiO2- 200mg HAP-乙醇浓度 1.68ml/min

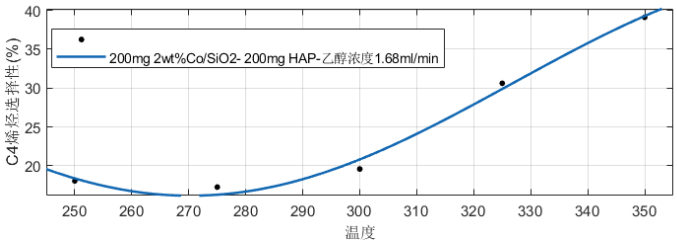
（1）乙醇转化率与温度：因具有较强线性，采用一次多项式拟合



Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 0.663 (0.5407, 0.7852)
p2 = -161.9 (-198.8, -125)

图片 4

（2）C4 烯烃选择性与温度：采用傅里叶多项式进行拟合



General model Fourier1:
f(x) = a0 + a1*cos(x*w) + b1*sin(x*w)
Coefficients (with 95% confidence bounds):
a0 = 30.58 (-49.86, 111)
a1 = -4.786 (-531.8, 522.2)
b1 = -13.58 (-130.9, 103.7)
w = 0.02781 (-0.1055, 0.1611)

图片 5

相关性

		温度	乙醇转化率(%)	乙烯选择性(%)	C4烯烃选择性(%)	乙醛选择性(%)	碳数为4-12脂肪醇选择性(%)	甲基苯甲醛和甲基苯甲醇选择性(%)	其他生成物的选择性(%)
温度	皮尔逊相关性	1	.995**	.892*	.914*	.980**	-.938*	.707	.979**
	Sig.（双尾）		.000	.042	.030	.003	.018	.182	.004
	个案数	5	5	5	5	5	5	5	5

表 4

同时，由皮尔逊相关系数和 *Sig* 的大小可知乙醇转化率与温度呈显著正相关，C_4 烯烃选择性与温度呈显著正相关，且乙醇转化率更加显著。

后文由于篇幅问题，不再在文章中附上相关性与显著性系数表和拟合得到的类型相同的曲线效果图，可见于附件之中。

3.对于 200mg 1wt%Co/SiO2- 200mg HAP-乙醇浓度 0.9ml/min

（1）乙醇转化率与温度：采用高斯多项式进行拟合。

（2）C4 烯烃选择性与温度：采用三次多项式进行拟合

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 88.13 (79.61, 96.65)
b1 = 441.1 (410.5, 471.7)
c1 = 129.1 (97.92, 160.3)

Linear model Poly3:
f(x) = p1*x^3 + p2*x^2 + p3*x + p4
Coefficients (with 95% confidence bounds):
p1 = -1.862e-05 (-2.605e-05, -1.119e-05)
p2 = 0.01851 (0.01074, 0.02629)
p3 = -5.707 (-8.367, -3.048)
p4 = 565.6 (268.2, 863.1)

4.对于 200mg 0.5wt%Co/SiO2- 200mg HAP-乙醇浓度 1.68ml/min

（1）乙醇转化率与温度：因具有较强线性，采用一次多项式拟合，呈正相关

（2）C4 烯烃选择性与温度：采用三次多项式拟合

Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 0.5816 (0.5246, 0.6386)
p2 = -144.5 (-162.8, -126.3)

Linear model Poly3:
f(x) = p1*x^3 + p2*x^2 + p3*x + p4
Coefficients (with 95% confidence bounds):
p1 = -2.106e-05 (-3.789e-05, -4.232e-06)
p2 = 0.02175 (0.00533, 0.03817)
p3 = -7.136 (-12.4, -1.869)
p4 = 763.7 (208.2, 1319)

5.对于 200mg 2wt%Co/SiO2- 200mg HAP-乙醇浓度 0.3ml/min

（1）乙醇转化率与温度：采用二次多项式拟合

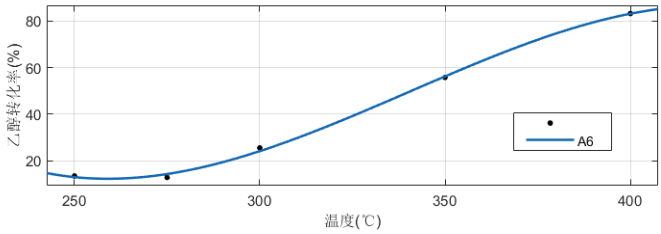
（2）C4 烯烃选择性与温度：采用高斯多项式拟合

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0032 (0.001894, 0.004506)
p2 = -1.672 (-2.523, -0.8214)
p3 = 232.4 (96.31, 368.5)

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 194 (-1232, 1620)
b1 = 668.7 (-381.7, 1719)
c1 = 210.6 (-140.2, 561.5)

6.对于 200mg 5wt%Co/SiO2- 200mg HAP-乙醇浓度 1.68ml/min

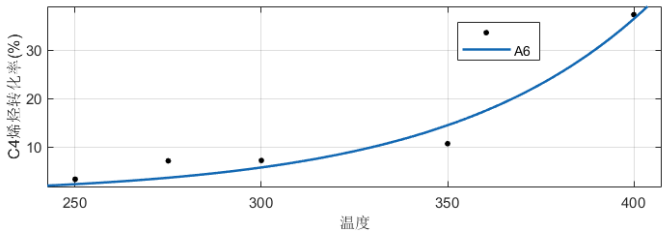
（1）乙醇转化率与温度：采用三次多项式拟合



图片 6

Linear model Poly3:
f(x) = p1*x^3 + p2*x^2 + p3*x + p4
Coefficients (with 95% confidence bounds):
p1 = -3.488e-05 (-0.00017, 0.0001003)
p2 = 0.03558 (-0.09602, 0.1672)
p3 = -11.41 (-53.55, 30.73)
p4 = 1187 (-3251, 5625)

（2）C4 烯烃选择性与温度：采用指数多项式拟合



图片 7

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.02233 (-0.06596, 0.1106)
b = 0.01849 (0.008373, 0.02862)

7.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 0.3ml/min（左为乙醇转化率与温度，右为 C4 烯烃选择性与温度，下文以此类推）

Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 0.3773 (0.3527, 0.4018)
p2 = -74.18 (-82.02, -66.34)

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.2268 (0.09008, 0.3635)
b = 0.01248 (0.0109, 0.01407)

8.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 0.9ml/min

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.001752 (0.0009778, 0.002527)
p2 = -0.8029 (-1.309, -0.297)
p3 = 97.16 (16.58, 177.7)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0007471 (0.0003593, 0.001135)
p2 = -0.2446 (-0.4979, 0.008599)
p3 = 19.82 (-20.52, 60.16)

9.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 2.1ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.006922 (0.002863, 0.01098)
b = 0.0217 (0.02021, 0.02319)

Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 0.2538 (0.2201, 0.2875)
p2 = -59.1 (-69.86, -48.33)

10.对于 50mg 5wt%Co/SiO2- 50mg HAP-乙醇浓度 2.1ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.001473 (-0.0008964, 0.003842)
b = 0.02469 (0.02063, 0.02876)

General model Exp2:
f(x) = a*exp(b*x) + c*exp(d*x)
Coefficients (with 95% confidence bounds):
a = 11.36 (-405.4, 428.1)
b = -0.007058 (-0.1527, 0.1386)
c = 0.0001352 (-0.005904, 0.006174)
d = 0.02793 (-0.08035, 0.1362)

11.对于 50mg 1wt%Co/SiO2+　90mg 石英砂-乙醇浓度 1.68ml/min，无 HAP

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.0003764 (-3.798e-05, 0.0007907)
b = 0.02843 (0.02565, 0.0312)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0001831 (9.296e-05, 0.0002733)
p2 = -0.06745 (-0.1263, -0.008569)
p3 = 5.586 (-3.794, 14.97)

12.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 1.68ml/min

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.001897 (0.001228, 0.002567)
p2 = -0.9506 (-1.388, -0.5135)
p3 = 120.9 (51.31, 190.6)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0009019 (0.0004882, 0.001316)
p2 = -0.3829 (-0.6531, -0.1127)
p3 = 45.32 (2.283, 88.37)

13.对于 67mg 1wt%Co/SiO2- 33mg HAP-乙醇浓度 1.68ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.007815 (0.0002393, 0.01539)
b = 0.02136 (0.0189, 0.02383)

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 27.98 (25.85, 30.11)
b1 = 396.6 (374.3, 418.8)
c1 = 109.2 (86.67, 131.7)

14.对于 33mg 1wt%Co/SiO2- 67mg HAP-乙醇浓度 1.68ml/min

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.002176 (0.0008034, 0.003549)
p2 = -1.083 (-1.979, -0.186)
p3 = 137.9 (-4.936, 280.7)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0009691 (0.0007268, 0.001211)
p2 = -0.494 (-0.6522, -0.3357)
p3 = 64.86 (39.66, 90.06)

15.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 1.68ml/min

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.001887 (0.001181, 0.002593)
p2 = -0.9503 (-1.411, -0.4894)
p3 = 121.5 (48.09, 194.9)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0009285 (-1.309e-05, 0.00187)
p2 = -0.3654 (-0.9804, 0.2495)
p3 = 39.07 (-58.89, 137)

16.对于 100mg 1wt%Co/SiO2- 100mg HAP-乙醇浓度 1.68ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.01647 (0.006438, 0.02651)
b = 0.01978 (0.01823, 0.02133)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0009926 (0.0001329, 0.001852)
p2 = -0.4028 (-0.9642, 0.1587)
p3 = 41.33 (-48.1, 130.8)

17.对于 10mg 1wt%Co/SiO2- 10mg HAP-乙醇浓度 1.68ml/min

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 344.7 (-2154, 2844)
b1 = 647.6 (74.92, 1220)
c1 = 148.2 (-2.569, 298.9)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0004777 (-0.0002604, 0.001216)
p2 = -0.1903 (-0.6714, 0.2907)
p3 = 20.97 (-55.97, 97.91)

18.对于 25mg 1wt%Co/SiO2- 25mg HAP-乙醇浓度 1.68ml/min

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 1.061e+08 (-1.668e+10, 1.689e+10)
b1 = 1666 (-1.135e+04, 1.468e+04)
c1 = 327.3 (-1307, 1962)

General model Fourier1:
f(x) = a0 + a1*cos(x*w) + b1*sin(x*w)
Coefficients (with 95% confidence bounds):
a0 = 14.87 (0.9395, 28.8)
a1 = -7.26 (-47.92, 33.4)
b1 = 6.258 (-57.85, 70.37)
w = 0.01976 (-0.006821, 0.04635)

19.对于 50mg 1wt%Co/SiO2- 50mg HAP-乙醇浓度 2.1ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.01256 (0.01021, 0.0149)
b = 0.02046 (0.01998, 0.02094)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0006497 (0.0004055, 0.000894)
p2 = -0.2761 (-0.4353, -0.117)
p3 = 32.4 (6.943, 57.86)

20.对于 75mg 1wt%Co/SiO2- 75mg HAP-乙醇浓度 1.68ml/min

General model Exp1:
f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
a = 0.0587 (0.01874, 0.09866)
b = 0.01746 (0.01569, 0.01922)

General model Gauss1:
f(x) = a1*exp(-((x-b1)/c1)^2)
Coefficients (with 95% confidence bounds):
a1 = 30.76 (25.44, 36.07)
b1 = 407.8 (365.2, 450.4)
c1 = 100.9 (61.85, 140)

21.对于 100mg 1wt%Co/SiO2- 100mg HAP-乙醇浓度 0.9ml/min

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.003278 (0.002265, 0.004291)
p2 = -1.711 (-2.371, -1.051)
p3 = 228.7 (123.1, 334.3)

Linear model Poly2:
f(x) = p1*x^2 + p2*x + p3
Coefficients (with 95% confidence bounds):
p1 = 0.0004518 (-3.461e-05, 0.0009382)
p2 = -0.06002 (-0.377, 0.257)
p3 = -9.858 (-60.56, 40.85)

5.1.6 总结

由于二次多项式形式简洁，且由二次多项式拟合出来的拟合函数已经基本能将数据中的SSE降的比较低，R^2尽可能靠近 1，因此，在对本题中乙醇转化率以及 C4 烯烃选择性与温度的关系中大部分使用的是二次多项式拟合，其次是指数多项式拟合，少部分催化剂组合使用了高斯多项式和傅里叶多项式进行拟合。

对于拟合得到的结果，我们不难看出，在 250℃-400℃这一区间中，当反应温度为 250℃时，乙醇转化率较低，基本在 2%-9%之间波动，催化剂的选择性使得乙醇转化为 $C_4C_{12}OH$，由碳数为 4-12 的脂肪醇在此时的选择性很高可以得到这一结论。

相关性

		温度	乙醇转化率(%)	乙烯选择性（%）	C4烯烃选择性(%)	乙醛选择性(%)	碳数为4-12脂肪醇选择性(%)	甲基苯甲醛和甲基苯甲醇选择性(%)	其他生成物的选择性(%)
温度	皮尔逊相关性	1	.912*	.918**	.978**	.771	-.997**	.615	.996**
	Sig.（双尾）		.011	.010	.001	.072	.000	.194	.000
	个案数	6	6	6	6	6	6	6	6

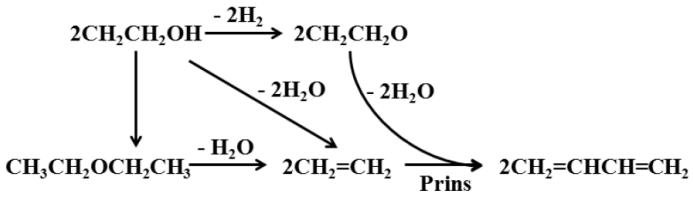
表 5

并且我们还可以总结得出，由皮尔逊相关系数和 *Sig* 的大小可知乙醇转化率与温度呈显著正相关，C4 烯烃选择性与温度呈显著正相关，且乙醇转化率相比 C4 烯烃选择性更加显著。

当反应温度在 350℃前，随着反应温度的升高，反应的产物仍然为碳数为 4-12 的脂肪醇（$C_4C_{12}OH$）；当反应温度逐渐超过 350℃，C_4-$C_{12}OH$ 的选择性开始不断下降，由 C4 烯烃的选择性不断增大可以看出，反应的产物转变为 C4 烯烃。

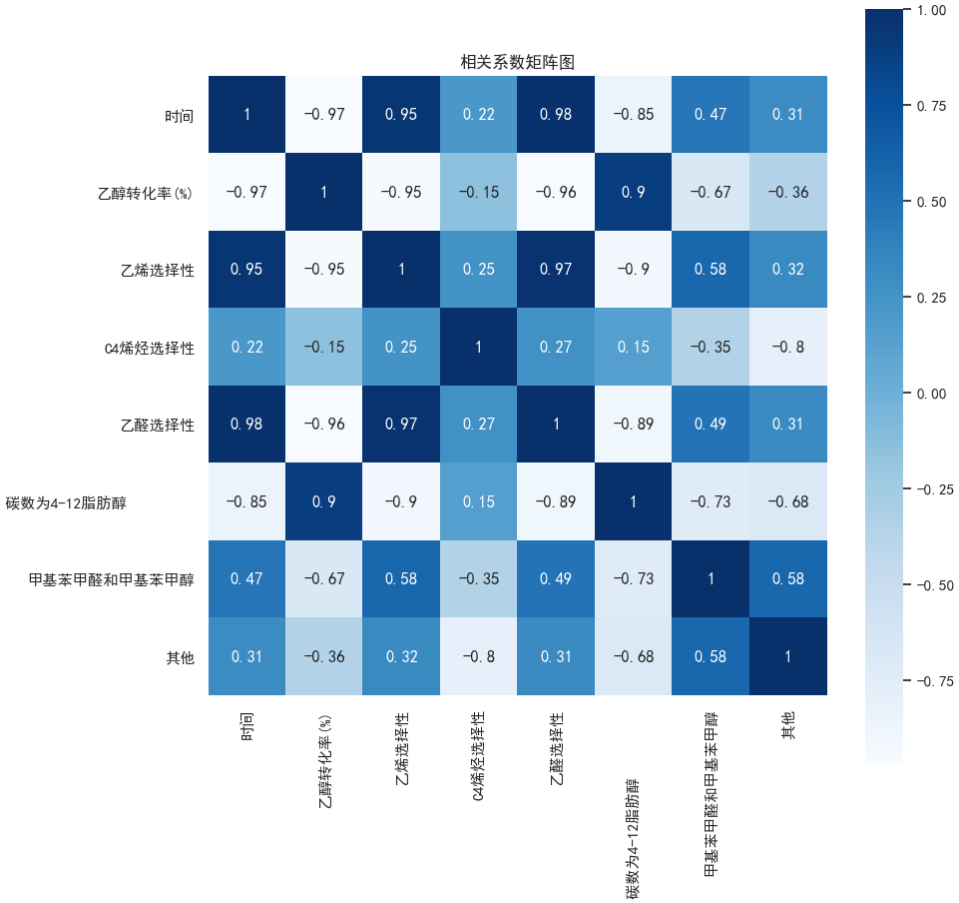
随着反应温度的提高，乙醇的转化率和 C4 烯烃的选择性基本也呈现上升的趋势，C4-C12OH 的选择性开始降低，C4 烯烃的选择性不断增加，乙醇的转化率渐渐上升。同时，我们不难发现，在 400℃前，乙醇的转化率是保持上升的，C4 烯烃的选择性也在不断增加。因此，不难判断出，对于 250℃-400℃这一温度区间，400℃是催化剂反应的适宜温度。

最后，我们发现附件中各产物以及中间产物的关系符合普林斯机理：



图片 8

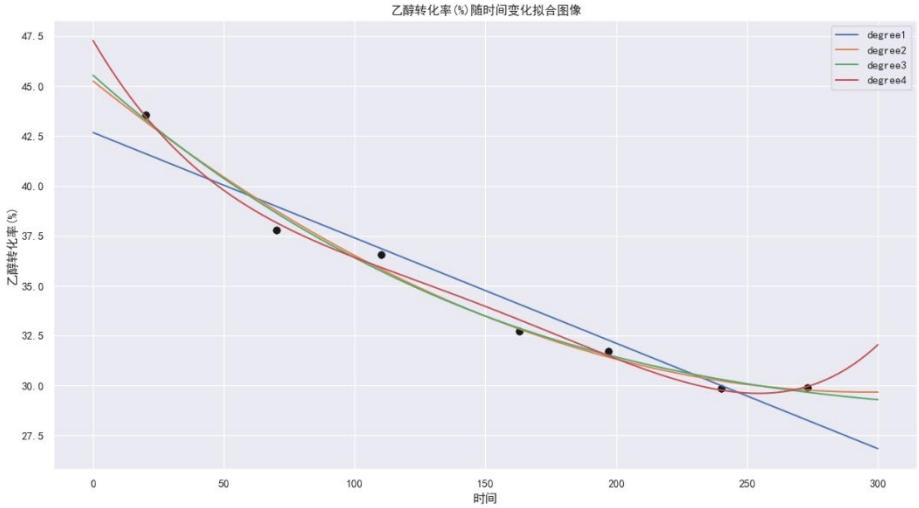
5.1.7 350℃给定催化剂组合在不同时间测试结果分析



图片 9

通过计算不同变量之间的相关系数，得到了相关系数矩阵，观察几种实验产物与时间的相关系数可以得到结论：乙醇转化率以及碳数为 4-12 的脂肪醇与时间呈负相关，乙烯转化率和乙醛选择性与时间呈正相关。因此可以得出结论，乙醇与碳数为 4-12 的脂肪醇在一定催化剂的条件下，在 350 摄氏度时较为不稳定，乙烯和乙醛十分稳定。C4 烯烃以及甲基苯甲醛和甲基苯甲醇与时间的相关系数绝对值并不高，因此与时间并没有具有较为显著的关系，因此也可以认定为稳定。

运用线性回归和多项式回归的方法，分别将乙醇，碳数为 4-12 的脂肪醇，乙烯和乙醛作为因变量，将时间作为自变量进行回归分析。回归的模型次数从一次到四次，根据曲线的趋势判断反应产物的稳定性，选择最优的模型。

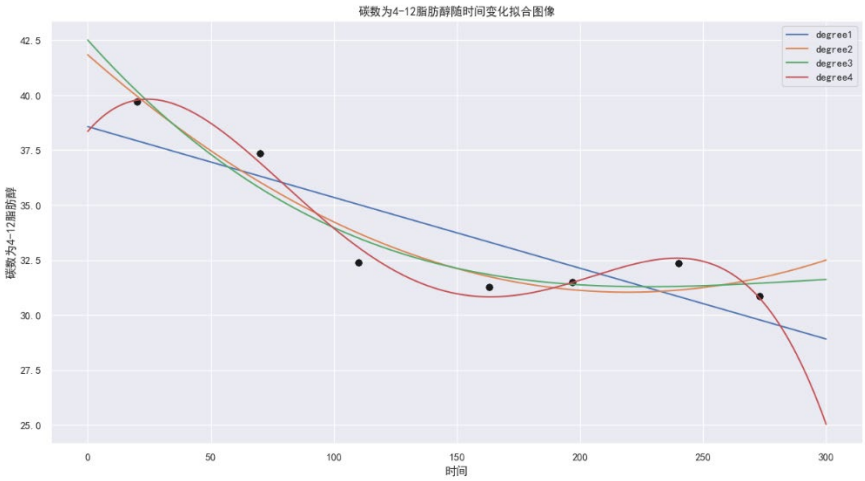


图片 10

可以看出拟合的曲线和直线在时间 0~300 中是呈现下降的趋势，其中四次多项式的拟合效果最好，但是存在过拟合的现象，因此本模型选择三次的多项式回归模型。

将时间x作为自变量，乙醇转化率(%)y作为因变量，得到以下的公式：

$$y = -2.21 * 10^{-7} * x^3 + 2.75 * 10^{-4} * x^2 - 1.16 * 10^{-1} * x^1 \quad (16)$$

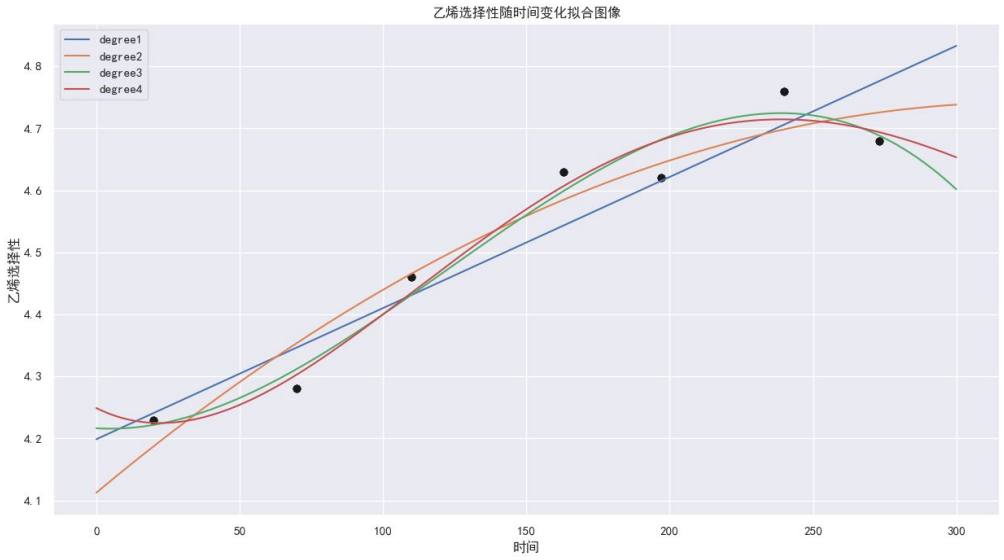


图片 11

可以看出拟合的曲线和直线在时间 0~300 中是呈现下降的趋势，其中四次多项式的拟合效果最好，但是存在过拟合的现象，因此本模型选择三次的多项式回归模型。

将时间x作为自变量，碳数为 4-12 脂肪醇y作为因变量，得到以下的公式：

$$y = -5.14 * 10^{-7} * x^3 + 4.508 * 10^{-4} * x^2 - 1.25 * 10^{-1} * x^1 \quad (17)$$

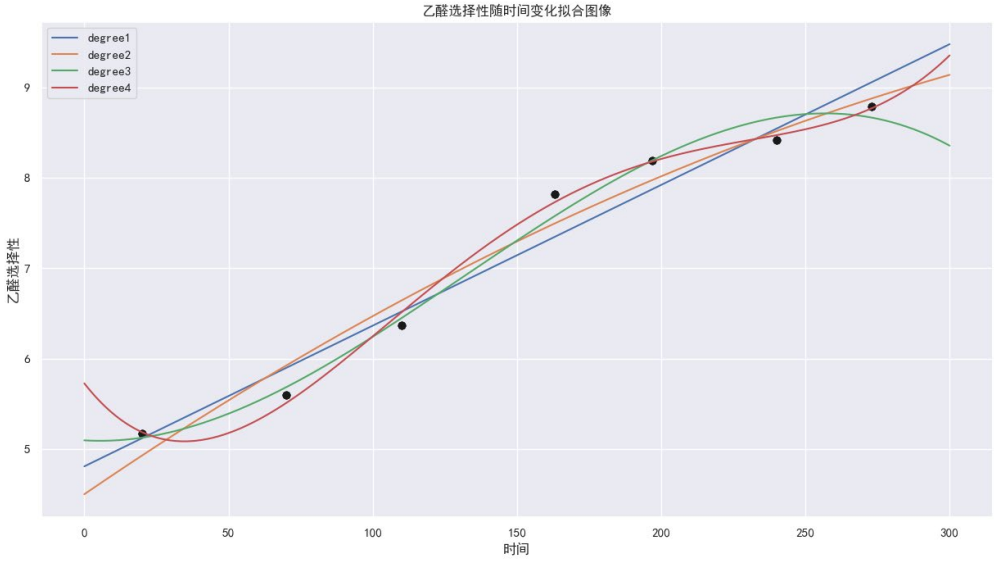


图片 12

可以看出拟合的曲线和直线在时间 0~300 中是呈现上升的趋势，这说明乙烯是能够在 350 摄氏度以及催化剂的条件下存在的，体现出乙烯的稳定性是较高的。多项式回归的拟合效果最好，但是存在过拟合的现象，因此本模型选择三次的多项式回归模型。

将时间x作为自变量，乙烯选择性y作为因变量，得到以下的公式：

$$y = -7.9515 * 10^{-8} * x^3 + 2.905 * 10^{-5} * x^2 - 2.7579 * 10^{-4} x \quad (18)$$



图片 13

可以看出拟合的曲线和直线在时间 0~300 中是呈现上升的趋势，这说明乙醛是能够在 350 摄氏度以及催化剂的条件下存在的，体现出乙醛的稳定性是较高的。多项式回归的拟合效果最好，但是存在过拟合的现象，因此本模型选择三次的多项式回归模型。

将时间x作为自变量，乙醛选择性y作为因变量，得到以下的公式：

$$y = -4.562 * 10^{-7} * x^3 + 1.7954 * 10^{-4} * x^2 - 1.932 * 10^{-3} x \quad (19)$$

5.2 问题二模型的建立与求解

模型的建立

5.2.1 梯度提升决策树(GBDT)算法解释

首先初始化函数$f_0(x)=0$，对$m=1,2\ldots M$计算每一次实验的$r_{mi}=y_i-f_{m-1}(x),i=1,2,3,\ldots N$，拟合残差$r_{mi}$参数学习回归树得到$h_m(x)$，并且更新原始函数$f_m(x)=f_{m-1}(x)+h_m(x)$，最终得到回归问题的提升树 $f_M(x)=\sum_{m=1}^{M}h_m(x)$。

计算损失函数并没有考虑平方损失和指数损失函数时，因为对于一般损失函数而言，往往每一步优化起来不那么容易。针对这一问题，Friedman 提出了梯度提升树算法，这是利用最速下降的近似方法，其关键是利用损失函数的负梯度作为提升树算法中的残差的近似值。其中第 t 轮的第 i 个样本的损失函数的负梯度为：

$$-[\frac{\partial L\left(y,\mathrm{f}(x_i)\right)}{\partial\left(\mathrm{f}(x_i)\right)}]_{\mathrm{f}(x)=f_{t-1}(x)} \tag{20}$$

如果使用的损失函数是平方损失：

$$L\left(y,\mathrm{f}(x_i)\right)=\frac{1}{2}(y-\mathrm{f}(x_i))^2 \tag{21}$$

负梯度为：

$$-[\frac{\partial L\left(y,\mathrm{f}(x_i)\right)}{\partial\left(\mathrm{f}(x_i)\right)}]_{\mathrm{f}(x)=f_{t-1}(x)}=\ y-\mathrm{f}(x_i) \tag{22}$$

此时的 GBDT 的负梯度为残差，因此在损失函数的选择上，GBDT 将第 t 轮的第 i 个样本的损失函数的负梯度作为损失函数，对叶子区域$j=1,2,\ldots J$计算最佳的拟合值

$$\gamma_{jm}=\arg\min\sum L(y_i,f_{m-1}(x_i)+\gamma) \tag{23}$$

并且对强学习器进行更新，此时的函数为：

$$f_m(x_i)=\ f_{m-1}(x_i)+\sum_{j=1}^{J}\gamma_{jm}I\left(x\in R_{jm}\right) \tag{24}$$

最终得到了学习器

$$f(x)=\ f_M(x)\ =f_0(x)+\sum_{m=1}^{M}\sum_{j=1}^{J}\gamma_{jm}I\left(x\in R_{jm}\right) \tag{25}$$

5.2.2 XGBoost 算法基于 GBDT 的优化

XGBoost在回归与分类方面与GBDT相比都有不小的提升，首先XFBoost在损失函数在GBDT的基础上增加了正则化的控制变量，正则化项如下：

$$Lt=\frac{\lambda}{2}\sum_{j=1}^{J}w_{tj}^2+\gamma J \tag{26}$$

因此 XGBoost 的损失函数可以表达为：

$$L_t = \sum_{i=1}^{m} L(y_i, f(x_i) + h_t(x_i)) + \frac{\lambda}{2}\sum_{j=1}^{J} w_{tj}^2 + \gamma J \tag{27}$$

XGBoost 运用损失函数的二阶泰勒展开式进行求解，将损失函数进行泰勒展开，并且将每个叶子节点区域样本的一阶和二阶导数和单独表示如下，最后得到 XGBoost 的损失函数形式：

$$L_{t=} \sum_{j=1}^{J}[G_{tj}w_{tj} + \frac{1}{2}(H_{tj} + \lambda)w_{tj}^2] + \gamma J \tag{28}$$

与 GBDT 算法相比 XGBoost 算法拥有更好的并行性。因此，对不同的特征划分点，XGBoost 分别在不同的线程中并行选择分裂的最大增益。另外，XGBoost 算法在损失函数的基础上增加了正则项，这样可以用于控制模型复杂度，这对于本题提供的相对有限的数据量是有帮助的，因此在本题中考虑使用 XGBoost 算法。

5.2.3 lightGBM 算法基于 GBDT 的优化

lightGBM 算法在 GBDT 算法的基础上进行了这样一些优化。首先 lightGBM 采用了基于 Hisogram 的决策树算法，这对于本题提供的连续性并不强的温度与催化剂组合数据以及离散化的装配模式具便利性。

其次，lightGBM 算法带有深度限制的带深度限制的 Leaf-wise 的叶子生长策略：大多数 GBDT 工具使用低效的按层生长的决策树生长策略，因为它不加区分的对待同一层的叶子，带来了很多没必要的开销。实际上很多叶子的分裂增益较低，没必要进行搜索与分裂。LightGBM 使用了带有深度限制的按叶子生长算法。

最后，lightGBM 直接支持类别特征，并且支持高效的并行。因此，本题打算对于催化剂和温度对于乙醇转化率和 C4 烯烃选择性分别运用 XGBoost 算法和 lightGBM 算法构建模型，最后通过R^2以及 MSE 对于模型进行评估，选择更加适合本题数据的模型。

模型的求解

5.2.4 XGBoost 参数调优思路

首先采用的是 XGBoost 默认的学习速率 0.1 进行模型的训练，并不进行学习速率的下降。

接着是对于给定的学习速率和决策树数量，进行决策树特定参数调优（max_depth，min_child_weight，gamma，subsample,colsample_bytree）在，这样调参过程确定的是 XGBoost 的树的结构.

其次，选择不同的参数，对于 Xgboost 的正则化参数的调优（lambda，alpha）。这些参数可以降低模型的复杂度，从而提高模型的表现。尤其是针对本题的数据量较小的情况，防止模型过拟合的情况发生尤其重要。

最后对于 XGBoost 的学习速率进行调整,适当降低学习速率确定最优的 XGBoost 模型。

5.2.5 XGBoost 参数调优具体过程

本题中 XGBoost 的模型建立将主要依靠陈天奇的 XGBoost 类库，参数的调优主要基于 python sklearn 类库的网格搜索方法选择最优的超参数。

首先对这个值为树的最大深度以及最小叶子节点样本权重和这个组合进行调整。最大深度控制了树的结构，最小叶子节点样本权重这个参数用于避免过拟合。当它的值较大时，可以避免模型学习到局部的特殊样本。但是如果这个值过高，会导致欠拟合。

param：

param_test1 = {'max_depth':range(3,10,2),'min_child_weight':range(2,7,2)}

再对参数 gamma 进行调整。在 XGBoost 节点分裂时，只有分裂后损失函数的值下降了，才会分裂这个节点。gamma 指定了节点分裂所需的最小损失函数下降值。这个参数的大小决定了模型的保守程度。参数越高，模型越不保守。

param：

param_test2 = {'gamma':[i/100.0 for i in range(0,100)]}

再对参数 subsample 和 colsample_bytree 进行调整。subsample 控制对于每棵树的随机采样的比例。减小这个参数的值，算法会更加保守，避免过拟合。但是，如果这个值设置得过小，它可能会导致欠拟合。colsample_bytree 用来控制每棵随机采样的列数的占比(每一列是一个特征)。

param：

param_test3 = { 'subsample':[i/10.0 for i in range(1,10)],

'colsample_bytree':[i/10.0 for i in range(1,10)]}

接着再对模型的 gamma 参数进行调整，控制模型的正则项，防止出现过拟合的现象。

param：

param_test4 = { 'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05]}

最后进行学习速率的调整，选择最优的学习速率最终确定适合的模型。

param：

param_test5 = {'learning_rate':[0, 0.001, 0.005, 0.01, 0.05,0.1,0.5,1]}

5. 2. 6 lightGBM 参数调优思路

lightGBM 的调参思路与 XGBoost 相仿，首先选择默认的'boosting_type‘为 gbdt', 由于本题解决的问题是回归问题，因此 objective 的参数选择为 regression。

再选择较高的学习率，大概 0.1，这样是为了加快收敛的速度。加快模型的训练速度

接着对决策树基本参数调参，包括但不限于参数（num_leaves，max_depth, subsamplecolsample_bytree, feature_fraction）进行调整。

再对正则化参数调参，控制模型的正则化程度，防止过拟合。

最后，对学习率进行参数整，得到更好地模型参数。

5. 2. 7 lightGBM 参数调优具体过程

本题中 lightGBM 的模型建立将主要依靠 sklearn 自带的 lightGBM 类库，参数的调优主要基于 python sklearn 类库的网格搜索方法选择最优的超参数。

首先对 max_depth 和 num_leaves 这两个参数进行调整。max_depth 设置树深度，深度越大可能过拟合。因为 LightGBM 使用的是 leaf-wise 的算法，因此在调节树的复杂程度时，使用的是 num_leaves。大致换算关系：num_leaves =$2^{\max\_depth}$，但是它的值的设置应该小于或等于$2^{\max_depth}$，否则模型可能会出现过拟合现象。

```
param：
                                    parameters = {
                                'max_depth': [i for i in range(1,10)],
                                    'num_leaves': range(1,30),}
```

接着对也叫 min_child_samples，它的值取决于训练数据的样本个数和 num_leaves。将其设置的较大可以避免生成一个过深的树，但有可能导致欠拟合。同时进行调整的参数是 min_child_weight，使一个结点分裂的最小海森值之和。这两个参数共同控制树的深度。

```
param：
                                    parameters = {
                                'min_child_samples': range(1,20),
                                'min_child_weight':     [i/1000 for i in range(10)]}
```

再对参数 feature_fraction 进行调整。这个参数用来进行特征的子抽样。同时可以用来防止过拟合及提高训练速度。

```
param：
                                    parameters = {
                                'feature_fraction': [i/100 for i in range(1,100)],}
```

对参数 bagging_fraction 以及 bagging_freq 进行调整。这一组参数同时进行调整，可以选择更加适合模型的拟合程度和训练速度。

```
param：
                                    parameters = {
                                'bagging_fraction': [i/10 for i in range(1,11)],
                                    'bagging_freq': [i for i in range(1,11)],}
```

再对参数 cat_smooth 进行调整。通过这个浮点数的调整，可以让用于 category 特征的概率平滑。默认值为 10。它可以降低噪声在 category 特征中的影响，尤其是对于数据很少的类。调整这个参数这对于本题较少的数据量也是比较有效的。

```
param：
                                parameters = {'cat_smooth': [i for i in range(1,20)],}
```

最后组合调整参数 learning_rate 以及 num_iterations。学习速率和迭代次数控制模型的训练时间，之前使用较高的学习速率是因为可以让收敛更快，但是准确度肯定没有细水长流来的好。最后对于这个参数组合进行调整获得最优的超参数。

5. 2. 8 模型具体参数以及模型的选择

1.运用 XGBoost 算法对催化剂和温度对于乙醇转化率的影响的最优模型：

param：

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=0.7, gamma=0.59, gpu_id=-1,
importance_type='gain', interaction_constraints='',
learning_rate=0.1, max_delta_step=0, max_depth=5,
min_child_weight=4, missing=nan, monotone_constraints='()',
n_estimators=140, n_jobs=4, nthread=4, num_parallel_tree=1,
objective='reg:linear', random_state=27, reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, seed=27, subsample=0.8, tree_method='exact',
validate_parameters=1, verbosity=None)

result:

R^2：0.8918192850404874　　*MSE* : 36.455938363183144

importance [0.05487424 0.12949349 0.04828398 0.12543833 0.11332276 0.52858716]

2.运用 XGBoost 算法对催化剂和温度对于 C4 烯烃选择性的影响的最优模型：

param：

XGBRegressor(alpha=0.01, base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=0.7, gamma=0.96, gpu_id=-1,
importance_type='gain', interaction_constraints='',
learning_rate=0.1, max_delta_step=0, max_depth=5,
min_child_weight=2, missing=nan, monotone_constraints='()',
n_estimators=140, n_jobs=4, nthread=4, num_parallel_tree=1,
objective='reg:linear', random_state=27, reg_alpha=0.00999999978,
reg_lambda=1, scale_pos_weight=1, seed=27, subsample=0.5,
tree_method='exact', validate_parameters=1, verbosity=None)

result:

R^2：0.5088127635296485　　*MSE* : 45.154976360800454

importance [0.07210191 0.15641111 0.17877726 0.11485448 0.10866112 0.3691941]

3.运用 lightGBM 算法对催化剂和温度对于乙醇转化率的影响的最优模型：

param：

LGBMRegressor(bagging_fraction=0.9, bagging_freq=11, cat_smooth=1,
feature_fraction=0.59, is_unbalance=True, max_depth=2,
min_child_samples=2, min_child_weight=0.0, num_iterations=500,
num_leaves=3, objective='regression', reg_alpha=0.001,
reg_lambda=8)

result:

R^2：0.8523788920215144　　*MSE* : 49.74699987500174

importance [0.046, 0.225, 0.197, 0.106, 0.187, 0.239]

4.运用 lightGBM 算法对催化剂和温度对于 C4 烯烃选择性的影响的最优模型：

param：

LGBMRegressor(bagging_fraction=0.7, bagging_freq=8, cat_smooth=1,
feature_fraction=0.25, is_unbalance=True, learning_rate=0.8,
max_depth=1, min_child_samples=1, min_child_weight=0.0,
num_iterations=200, num_leaves=2, objective='regression',
reg_alpha=0.001, reg_lambda=8)

result:

R^2：0.5606752589062286　　*MSE* : 40.38724304270771

importance [0.055, 0.27, 0.175, 0.15, 0.145, 0.205]

本文将数据按照 9：1 的比例划分成训练集和测试集，运用训练集训练之后，对测试集的数据和预测数据进行R^2以及 *MSE* 的计算。通过这两个参数的计算对于模型进行评估，选择更加适合本题数据的模型。

催化剂和温度对于乙醇转化率(%)的影响的模型比较：

模型	R^2	*MSE*
XGBoost	0.891	36.45
lightGBM	0.852	49.74

表 6

可以看出 XGBoost 无论是在模型的解释度方面还是对于预测数据的拟合效果方面都是要优于 lightGBM 模型的，因此在催化剂和温度对于乙醇转化率(%)的影响的模型选择上选择 XGBoost 模型。

催化剂和温度对于 C4 烯烃选择性(%)的影响的模型比较：

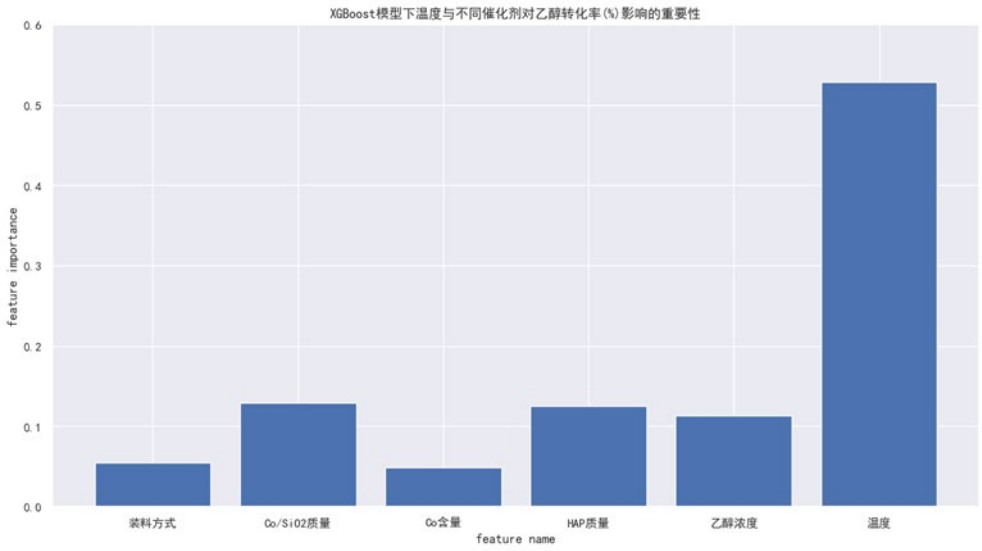
模型	R^2	*MSE*
XGBoost	0.508	45.15
lightGBM	0.561	40.38

表 7

可以看出 lightGBM 无论是在模型的解释度方面还是对于预测数据的拟合效果方面都是要优于 XGBoost 模型的，因此在催化剂和温度对于 C4 烯烃选择性(%)的影响的模型选择上选择 lightGBM 模型。

5.2.9 结论

绘制 XGBoost 模型的 feature_importance 图像，比对不同因变量对于乙醇转化率的影响程度。



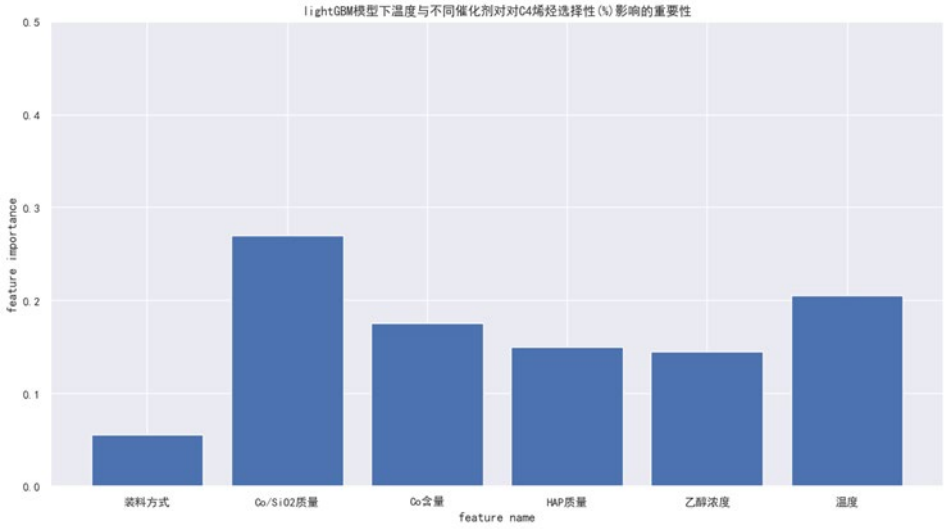
图片 14

这几个参数的 feature_imprtance 分别是

[0.05487424 0.12949349 0.04828398 0.12543833 0.11332276 0.52858716]

因此可以得出这样的结论：对于乙醇转化率，温度这个参数具有决定性的作用，且要远远高于其他的催化剂变量。在催化剂中，Co/SiO2 的质量以及 HAP 质量和乙醇浓度这三个变量对于乙醇转化率的影响相仿，但是 Co/SiO2 的影响要大于 HAP 质量大于乙醇浓度。Co 质量对于乙醇转化率的影响并不大，装料方式的影响同样不大。

绘制 lightGBM 模型的 feature_importance 图像，比对不同因变量对于 C4 烯烃选择性的影响程度。



图片 15

这几个参数的 feature_imprtance 分别是

[0.055, 0.27, 0.175, 0.15, 0.145, 0.205]

因此可以得出这样的结论：对于 C4 烯烃选择性(%)，温度这个参数并不具有决定性的作用，但是也有一定的影响。Co/SiO2 的质量对于 C4 烯烃选择性(%)具有显著的影响，其次是 Co 含量。HAP 质量和乙醇浓度这三个变量对于乙醇转化率的影响相仿。装料方式的影响同样不大。

5.3 问题三模型的建立与求解

模型的建立

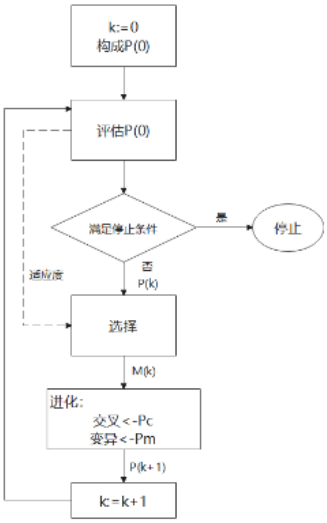
5.3.1 遗传算法解释

遗传算法思想源于生物科学的进化理论和遗传变异理论。

假定需要求解如下形式的最优化问题:

$$\text{maximize} \quad f(x)$$
$$\text{subject to} \quad x \in \Omega$$

使用遗传算法求解这一问题，第一步从可行集Ω中选定一组初始点，用 P(0)表示，代表初始种群，并计算 P(0)中每个点对应的目标函数值，基于计算结果通过交叉和变异操作，产生一组新的点 P(1)。循环进行这一过程，产生种群序列 P(2) ,P(3)…，直到达到停止条件。交叉和变异操作的目的在于创建一个新种群，使新种群目标函数的平均值大于上一代。



图片 16

这里首先介绍算法相关术语和定义。

染色体和表达模式：遗传算法不是针对约束集Ω中的点进行操作的，而是对这些点进行编码后再进行操作。首先需要将Ω映射为一个字符串的集合，这些字符串全等长，称为染色体，记 L 为字符串长度。每个染色体(记x)都对应的目标函数值，称为染色体的适应度(记$f(x)$)，为方便描述设目标函数非负。字符串的长度、字符表和编码方式称为表达模式。

选择步骤：在选择步骤中，利用选择操作产生一个新的种群$M(k)$，并使其个体的数量与$P(k)$相等。种群中个体的数量称为种群容量，用N表示。$M(k)$称为配对池，是在$P(k)$ 的基础上进行随机处理后得到的，即$M(k)$ 中每个个体$m(k)$以概率

$$\frac{f(x^{(k)})}{F(k)} \tag{29}$$

等于$P(k)$中的$x^{(k)}$。$F(k) = \sum f(x_i^{(k)})$，即染色体选入配对池的概率与其适应度成正比。这一选择模式被称为轮盘赌模式，此外还有锦标赛等不同模式。

进化步骤：这一步骤主要进行交叉与变异。交叉指从配对池中选择两条染色体，将他们的字符串相互交换一段子字符串，产生一对新的子代染色体。交叉完成后利用子代替代父代便完成了更新。

变异操作：对配对池中的每一条染色体以变异概率Pm改变其中的字符。

通过对配对池$M(k)$展开交叉和变异操作，得到新的种群$P(k+1)$。计算这一种群的适应度，进行选择进化，循环往复，直到得到最优解。遗传算法步骤可归纳为：

1. 令k:=0 ， 产生初始种群$P(0)$ 。
2. 计算$P(k)$ 中每个个体的适应度。
3. 如果满足停止规则，就停止迭代。
4. 从$P(k)$ 中选择$M(k)$ 。
5. 进化$M(k)$的，构成新种群$P(k+1)$ 。
6. 令k:=k+l ， 回到第 2 步。

模型的求解

5.3.2 模型带入

在第二问中，我们建立了基于 XGBoost 的乙醇转化率以及 C4 烯烃选择性预测模型。这里我们基于该模型，通过遗传算法，寻找最佳的催化剂组合及温度，使相同实验条件下 C4 烯烃收率尽可能高。

优化问题建立：我们将这一问题抽象为如下优化问题：

Maximize	C4 烯烃收率=乙醇转化率×C4 烯烃的选择性
Subject to	装料方式=1 或 2
	0≤Co/SiO2 质量≤250(mg)
	0≤Co 含量≤6.25(%)
	0≤HAP 质量≤250(mg)
	0≤乙醇浓度≤2.625(ml/min)
	0≤温度≤500(度)

确认方式如下：对于装料方式，设装料方式 I 为 1，装料方式 II 为 2。对于其他变量，由于附件以给出的数据范围有限，为了保证预测模型的准确性，仅在原有范围的基础上增加 25%。例如，附件 1 中的约束条件温度范围为 0 到 400 度，这里将约束定为 0 到 500 度。

遗传算法实现：这里我们使用了遗传算法包 Geatpy。具体参数设置如下：

参数	值
种群规模	20
最大进化代数	400
差分进化中的参数 F	0.7
重组概率	0.7

表 8

对于问题二的第一个问题，由于没有限制温度范围，我们选取附件 1 中使用不同装药方式的 C4 烯烃收率最高的样本：

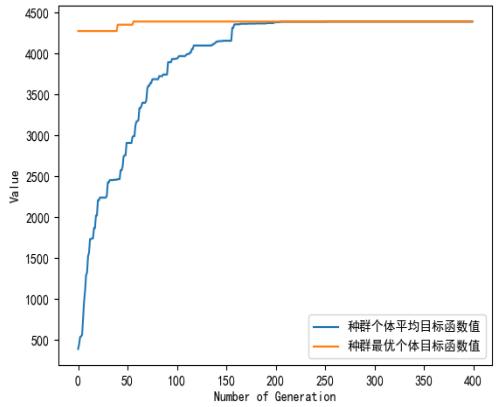
装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度(ml/min)	温度(度)	C4 烯烃收率（%）	C4 烯烃收率预测值（%）
I	200	1	200	0.9	400	44.73	42.76
II	100	1	100	0.9	400	26.49	24.33

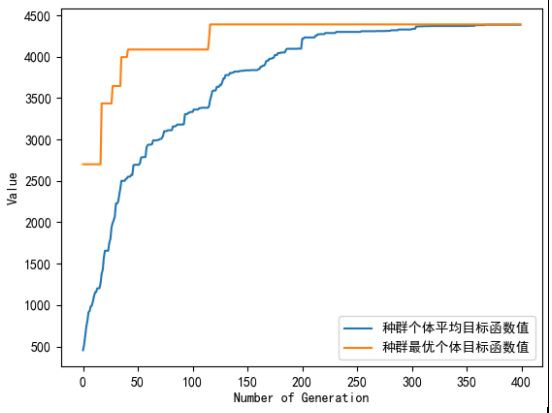
表 9

作为先验信息分别输入模型，在其附近寻找最优点。

迭代过程中种群最优个体目标函数值与种群个体平均目标函数值变化分别如下：

装药方式 I	装药方式 II
组图 1	





最终获得的最优点分别为

先验装料方式	现装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度 (ml/min)	温度 (度)	C4 烯烃收率预测值（%）
I	I	196.03	1.06	236.47	1.42	424.57	43.91
II	I	228.43	0.86	212.04	1.63	445.26	43.91

表 10

生成的两个最优点的 C4 烯烃收率预测值（%）均高于附件 1 种最优点的 C4 烯烃收率预测值（%），证明其相较附件中的最优点更优，实际使用时这两种组合均有较大可能有较好的效果。同时两最优点的装料方式均为 I，进一步佐证了装药方式 I 更优。

对于问题二的第二个问题，由于限制温度不高于 350 度，因此我们将优化问题改为

Maximize	C4 烯烃收率=乙醇转化率×C4 烯烃的选择性
Subject to	装料方式=1 或 2
	0≤Co/SiO2 质量≤250(mg)
	0≤Co 含量≤6.25(%)
	0≤HAP 质量≤250(mg)
	0≤乙醇浓度≤2.625(ml/min)
	0≤温度≤350(度)

我们选取附件 1 中温度小于等于 350 度，使用不同装药方式的 C4 烯烃收率最高的样本：

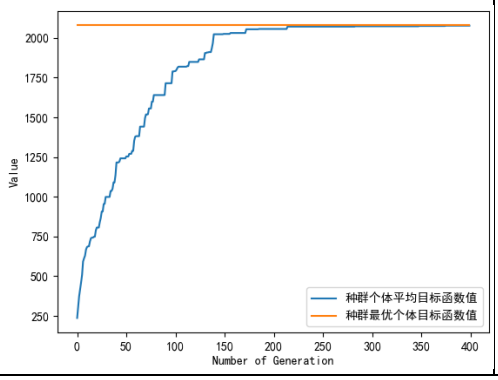
装料方式		Co/SiO2 质量(mg)	Co 含量 (%)	HAP 质量(mg)	乙醇浓度 (ml/min)	温度(度)	C4 烯烃收率（%）	C4 烯烃收率预测值（%）
I		200	2	200	1.68	350	26.54	20.80
II		100	1	100	0.9	350	7.57	7.45

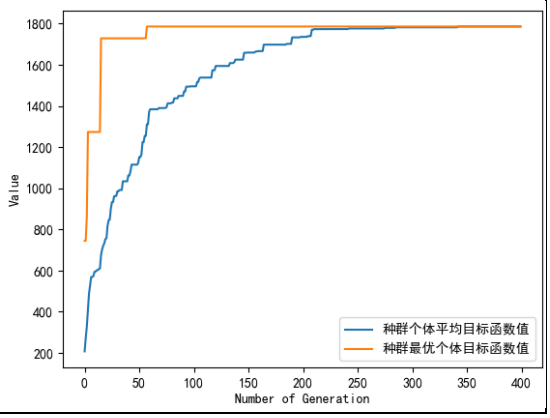
表 11

作为先验信息分别输入模型，在其附近寻找最优点。

迭代过程中种群最优个体目标函数值与种群个体平均目标函数值变化分别如下：

装药方式 I	装药方式 II
组图 2	





最终获得的最优点分别为

先验装料方式	现装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度 (ml/min)	温度 (度)	C4 烯烃收率预测值（%）
I	I	250	2.96	222.72	1.84	343.53	20.80
II	II	197.72	2.39	238.09	1.39	340.30	17.86

表 12

可见，装料方式 I 在 350 度以下时，仍优于装料方式 II。生成的实用装药方式 I 的最优点的 C4 烯烃收率几乎与附件 1 中的温度低于 350 度时的 C4 烯烃收率相同，因此这两点均为最优点，实际使用时这两种组合均能有较好的效果。

5. 4 问题四模型的建立与求解

实验一&实验二：验证问题 3.生成的最优催化剂组合与温度

我们在问题 3.生成的最优催化剂组合与温度如下：

温度限制	装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度 (ml/min)	温度 (度)	C4 烯烃收率预测值（%）
小于 350 度	I	250	2.96	222.72	1.84	343.53	20.80
无限制	I	228.43	0.86	212.04	1.63	445.26	43.91

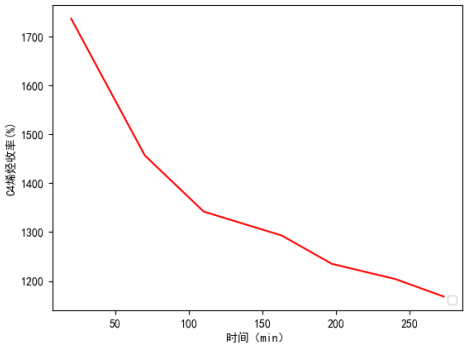
表 13

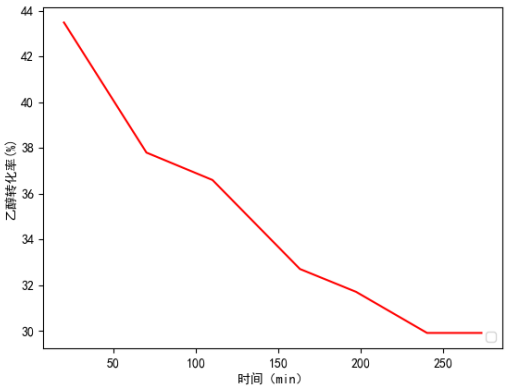
由于生成的最优催化剂组合与温度中的 Co/SiO2 质量(mg)、HAP 质量(mg)已经在附件 1 所给出的数据范围之外，并不能保证预测模型有足够的准确性，因此需要进行两次实验，验证其是否能够达到较高的 C4 烯烃收率。

实验三：验证附件 2 中 350 度时给定的催化剂组合实验最终 C4 烯烃收率是否稳定。

绘制附件 2 中 350 度时给定的催化剂组合实验乙醇转化率、C4 烯烃收率随时间变化的曲线可见，乙醇转化率随时间逐渐降低并趋于稳定，但 C4 烯烃收率在该段时间内还并未趋于稳定。

组图 3





因此，需要设计附件 2 中 350 度时给定的催化剂组合在 350min 的测试结果进行分析,检测 C4 烯烃收率是否最终趋于稳定，若 350min 时的 C4 烯烃收率与 273min 时相近，则说明反应已经趋于稳定；若 350min 时的 C4 烯烃收率显著低于 273min 时，则说明反应仍未稳定，需要继续进行试验。

实验四：探究更高温度对乙醇转化率、C4 烯烃的选择性的影响。

在第一问中，有相关性检验以及拟合曲线可知，对于大多数催化剂组合，温度与乙醇转化率、C4 烯烃的选择性成正相关。而在第三问，为了寻找最优温度，将温度的变化范围由附件 1 中的 0-450 度扩大为了 0-500 度。为了探究更高温度对乙醇转化率、C4 烯烃的选择性的影响，同时验证第三问获得的最优催化剂组合与温度，我们选取该最优催化剂组合

装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度 (ml/min)
I	228.43	0.86	212.04	1.63

表 14

并将温度改为 500 度，在进行一次实验，并对结果进行分析。

实验五：研究反应时间对于乙烯和乙醛的选择性影响。

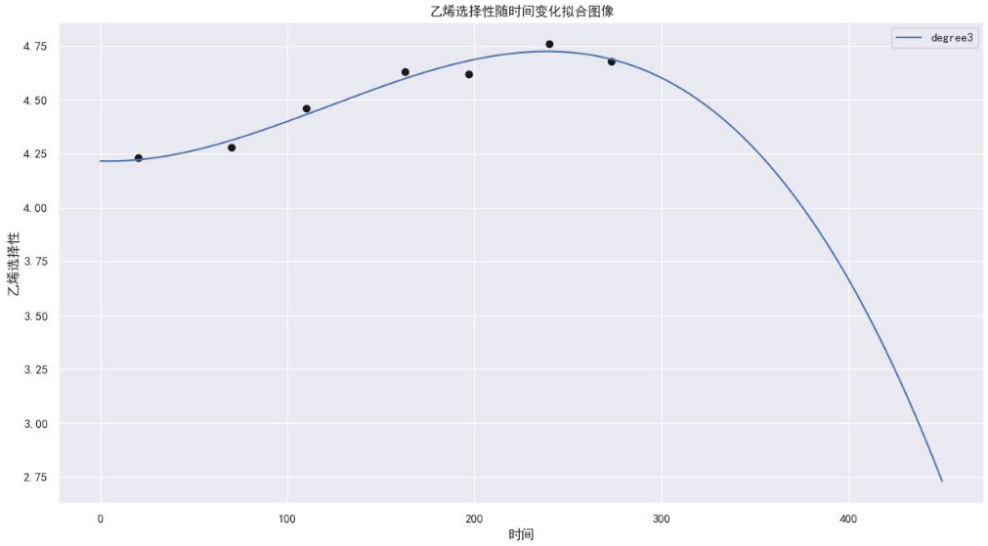
题目附件二提供的数据包含时间范围是 20min 到 273min。在这个时间段内乙烯选择性和乙醛选择性都是随时间上升的，如果考虑延长时间，是否出现乙烯与乙醛的选择性下降，转化为 C4 烯烃选择性，C4 烯烃选择性上升导致 C4 烯烃收率上升的情况并不可知，因此进行反应时间的延长是有研究价值的。

利用第一问的第二小问获得的乙烯与反应时间的二次函数的拟合多项式以及乙醛与反应时间的二次函数的拟合多项式，延长时间的范围，绘制相关的函数图像。

函数表达式如下：

将时间x作为自变量，乙烯选择性y作为因变量，得到以下的公式：

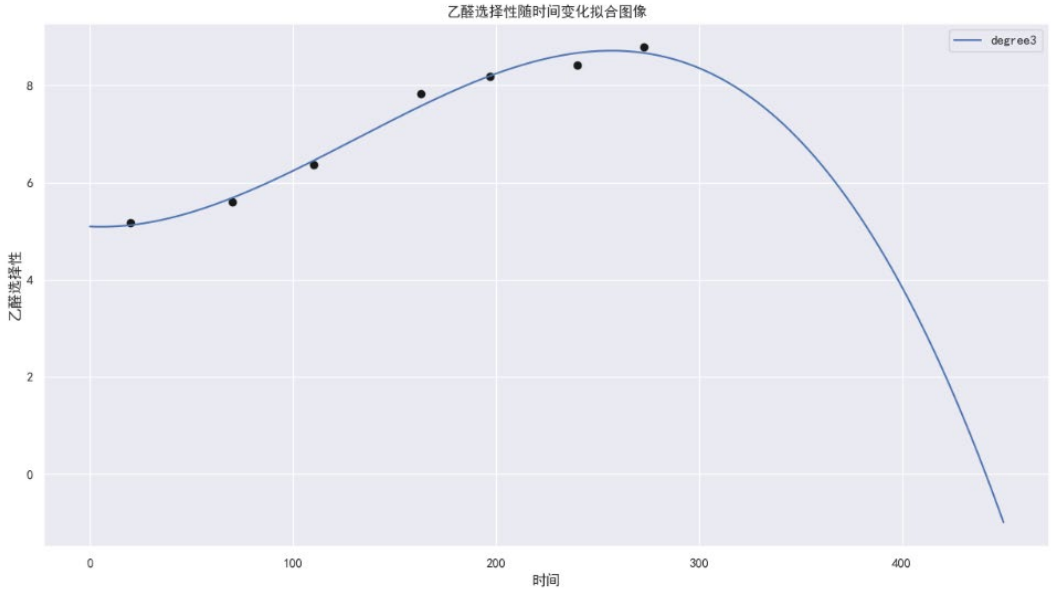
$$y = -7.9515 * 10^{-8} * x^3 + 2.905 * 10^{-5} * x^2 - 2.7579 * 10^{-4} x \quad (30)$$



图片 17

将时间x作为自变量，乙醛选择性y作为因变量，得到以下的公式：

$$y = -4.562 * 10^{-7} * x^3 + 1.7954 * 10^{-4} * x^2 - 1.932 * 10^{-3}x \quad (31)$$



图片 18

可以看到在时间调整到 400min 时，三次函数的预测猜测模型预测乙烯选择性以及乙醛选择性都会有显著的下降。因此设计实验，在 350 摄氏度某催化剂条件下延长实验时间到 400min，并对在 273 分钟以及 400 分钟乙烯选择性，乙醛选择性，C4 烯烃选择性和 C4 烯烃收率进行测量。观察时间的延长，乙醇，乙醛的选择性是否会下降，下降的那部分乙醇和乙醛是否会部分转化为 C4 烯烃，转化结果是否会对 C4 烯烃收率进行影响。

六、 模型的分析与检验

这里我们找到了一篇乙醇偶合制备 C4 烯烃相关的学术论文[1]。该文献通过实验确定，当 Co/SiO2 与 HAP 混料比为 1:1、反应温度为 400 ℃、Co 负载量为 1 wt%时，催化剂性能最优，C4 烯烃收率最高。这与我们找到的最优点之一较为接近，证明算法生成的最优方案具有较高的可靠性。

先验装料方式	现装料方式	Co/SiO2 质量(mg)	Co 含量(%)	HAP 质量(mg)	乙醇浓度(ml/min)	温度(度)	C4 烯烃收率预测值（%）
I	I	196.03	1.06	236.47	1.42	424.57	43.91

表 15

七、 模型的评价、改进与推广

7.1 模型的优点

在问题 2.中使用的 XGBoost 算法优点主要在于其在 GBDT 算法的基础上改变了目标函数，目标函数使用了损失函数关于待求函数的二阶导数。并且使用了许多方法去避免过拟合的现象发生，比如引入参数 reg_alpha 与 subsample 参数。另外 XGBoost 支持并行化同级层节点可并行化，多线程并行加快训练速度，是模型训练更加高效。

在问题 2.中使用的 lightGBM 算法优点在于使用 Histogram 算法，大大降低了算法的时间开销以及对于内存的开销，并且运用了带深度限制的 Leaf-wise 的叶子生长策略，大大减少了生成树的没有必要的搜索与分裂。

7.2 模型的缺点

在问题 1.的拟合过程中，由于数据过少，存在过拟合现象。问题 2.中 lightGBM 即使使用有深度限制的 Leaf-wise 算法，也有可能会长出比较深的决策树，也会产生过拟合的现象。

在问题 2.中 XGBoost 算法的主要缺点在于叶子节点的分裂增益较低，没必要进行跟进一步的分裂，这就带来了不必要的开销。并且由于 XGBoost 在迭代前都会对样本进行与排序，在数据量大时，贪心算法的时间开销较大。lightGBM 算法的主要缺点在于可能会生成较为深的决策树，同时由于基于偏差的算法，会对噪声点较为敏感。

7.3 模型的改进

对于问题 1，采取收集更多测试数据集，设置训练集与测试集，判断拟合得到的结果是否过拟合的策略。

对于问题 2. 中装料方式 A 与装料方式 B 的数据并不平均，运用 XGBoost 与 lightGBM 算法回归拟合可能会对于装料方式 B 的信息提取没有装料方式 A 的多。基于这个考虑，首先在数据层面，通过条件生成式对抗网络 CGAN，学习少数类样本的分布信息，训练生成器生成少数类补充样本，再利用 XGBoost 进行特征组合生成新的特征，再通过最大相关最小冗余算法筛选出更适合不平衡数据分类的特征子集；最后在算法层面，引入针对不平衡数据分类改进后的 XGBoost 通过新的数据集训练得到最终的模型。这样得到的模型对于不平衡数据集的信息提取会比原先的模型效果更好。

对于问题 3.遗传算法存在容易过早收敛的缺点，后续考虑可以使用粒子群算法等其他全局优化算法进行问题求解，与遗传算法生成的结果对照，优化结果。

八、 参考文献

[1] 吕绍沛. 乙醇偶合制备丁醇及 C4 烯烃[D].大连理工大学,2018.
[2] Chen, TQ (Chen, Tianqi) XGBoost: A Scalable Tree Boosting System Univ Washington 2016
[3] Ke, GL (Ke, Guolin) LightGBM: A Highly Efficient Gradient Boosting Decision Tree Microsoft Res 2017
[4]宋玲玲 王时绘 杨超 盛潇 改进的 XGBoost 在不平衡数据处理中的应用研究 湖北大学 2020
[5]王晓晖 张亮 李俊清 孙玉翠 田捷 韩睿毅 基于遗传算法与随机森林的 XGBoost 改进方法研究 山东农业大学 2020

附录

附录 1

介绍：问题 3.代码

```
import geatpy as ea
import numpy as np
import pandas as pd
import pickle
print(ea.__version__)
pkl_file_yc = open('cuihuaji_yichun_2.dat', 'rb')
reg_yc = pickle.load(pkl_file_yc)
pkl_file_c4 = open('cuihuaji_c4_2.dat', 'rb')
reg_c4 = pickle.load(pkl_file_c4)
def yc(x0,x1,x2,x3,x4,x5):
    a_yc = np.array([[x0,x1,x2,x3,x4,x5]])
    df_X_yc = pd.DataFrame(a_yc)
    df_X_yc.columns =['装料方式','Co/SiO2 质量', 'Co 含量', 'HAP 质量', '乙醇浓度
', '温度']
    a_yc = reg_yc.predict(df_X_yc)
    return a_yc[0]
def c4(x0,x1,x2,x3,x4,x5):
    a_c4 = np.array([[x0,x1,x2,x3,x4,x5]])
    df_X_c4 = pd.DataFrame(a_c4)
    df_X_c4.columns =['装料方式','Co/SiO2 质量', 'Co 含量', 'HAP 质量', '乙醇浓度
', '温度']
    a_c4 = reg_c4.predict(df_X_c4)
    return a_c4[0]
class MyProblem(ea.Problem):
    def __init__(self):
        '''Subject to   装料方式=1 或 2
                    0≤Co/SiO2 质量≤250(mg)
                    0≤Co 含量≤6.25(%)
                    0≤HAP 质量≤250(mg)
                    0≤乙醇浓度≤2.625(ml/min)
                    0≤温度≤500(度)'''
        name = 'MyProblem'
        M = 1
        maxormins = [-1]
        Dim = 6
        varTypes =[1]+[0] * (Dim-1)
        print(varTypes)
        lb = [1,0,0,0,0,0] # 下界
        ub = [2,250,6.25,250,2.625,350] # 上界
        lbin = [1,1,1,1,1,1]
```

```
            ubin = [1,1,1,1,1,1]
            ea.Problem.__init__(self, name, M, maxormins, Dim, varTypes, lb, ub, lbin, ub
in)

        def aimFunc(self, pop): # 目标函数 Maximize C4 烯烃收率=乙醇转化率×C4
烯烃的选择性
            Vars = pop.Phen
            x1 = Vars[:, [0]]
            x2 = Vars[:, [1]]
            x3 = Vars[:, [2]]
            x4 = Vars[:, [3]]
            x5 = Vars[:, [4]]
            x6 = Vars[:, [5]]
            tttt=x1*x2
            temptt=yc(x1[0][0],x2[0][0],x3[0][0],x4[0][0],x5[0][0],x6[0][0])*c4(x1[0][0],x
2[0][0],x3[0][0],x4[0][0],x5[0][0],x6[0][0]) # 计算目标函数值，赋值给 pop 种群对象
的 ObjV 属性
            #yty=[np.array([temptt])]
            tttt[0][0]=temptt
            print(temptt)
            pop.ObjV = tttt

        def calReferObjV(self):
            referenceObjV = np.array([[100*100]])
            return referenceObjV
    if __name__ == '__main__'
```

附录 2

介绍：问题 1.第二问代码

```
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import numpy as np
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号



import seaborn as sns
import pandas as pd
sns.set(font_scale = 1)
```

```
import matplotlib.pyplot as plt
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
plt.figure(figsize=(10, 10), dpi=100)
data1 = pd.read_excel(r'C:\Users\86139\Desktop\B\1.2.xlsx')
a=data1.iloc[:,0:].corr()
sns.heatmap(a, annot=True, vmax=1, square=True, cmap="Blues")
plt.title("相关系数矩阵图")
plt.show()



data = pd.read_excel(r'C:\Users\86139\Desktop\B\1.2.xlsx')
#生成多项式回归的函数，默认参数为 2
def PolynomialRegression(degree=2,**kwarges):
    return make_pipeline(PolynomialFeatures(degree),LinearRegression())


from sklearn.preprocessing import PolynomialFeatures as PF
from sklearn.linear_model import LinearRegression
time = data['时间']
al = data['乙醇转化率(%)']
fig = plt.figure(figsize=(15, 8), dpi=100)
score =[]
coef = []
for i in range(0,4):
    i  =i+1
    poly = PF(degree=i)
    X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
    # 训练数据的拟合
    LinearR_ = LinearRegression().fit(X_, al)
    X_curve = np.linspace(0,300,500).reshape(-1,1)
    X_curve1 = PF(degree=i).fit_transform(X_curve)
    plt.scatter(time,al,color='k')
    y_curve = LinearR_.predict(X_curve1)
    plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
    plt.title("乙醇转化率(%)随时间变化拟合图像")
    plt.xlabel("时间")
    plt.ylabel("乙醇转化率(%)")
    coef.append(LinearR_.coef_)
    score.append(LinearR_.score(X_,al))
    plt.legend()
    plt.show()
```

```
a = range(1,10)
print([*zip(a,coef,score)])


from sklearn.preprocessing import PolynomialFeatures as PF
from sklearn.linear_model import LinearRegression
fig = plt.figure(figsize=(15, 8), dpi=100)
al = data['乙烯选择性']
score =[]
coef = []
for i in range(0,4):
    i   =i+1
    poly = PF(degree=i)
    X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
    # 训练数据的拟合
    LinearR_ = LinearRegression().fit(X_, al)
    X_curve = np.linspace(0,300,500).reshape(-1,1)
    X_curve1 = PF(degree=i).fit_transform(X_curve)
    plt.scatter(time,al,color='k')
    y_curve = LinearR_.predict(X_curve1)
    plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
    coef.append(LinearR_.coef_)
    score.append(LinearR_.score(X_,al))
    plt.title("乙烯选择性随时间变化拟合图像")
    plt.xlabel("时间")
    plt.ylabel("乙烯选择性")
    plt.legend()
    plt.show()
a = range(1,10)
print([*zip(a,coef,score)])



fig = plt.figure(figsize=(15, 8), dpi=100)
al = data.iloc[:,5]
score =[]
coef = []
for i in range(0,4):
    i   =i+1
    poly = PF(degree=i)
    X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
    # 训练数据的拟合
    LinearR_ = LinearRegression().fit(X_, al)
    X_curve = np.linspace(0,300,500).reshape(-1,1)
```

```
        X_curve1 = PF(degree=i).fit_transform(X_curve)
        plt.scatter(time,al,color='k')
        y_curve = LinearR_.predict(X_curve1)
        plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
        coef.append(LinearR_.coef_)
        score.append(LinearR_.score(X_,al))
        plt.title("碳数为 4-12 脂肪醇随时间变化拟合图像")
        plt.xlabel("时间")
        plt.ylabel("碳数为 4-12 脂肪醇")
        plt.legend()
        plt.show()
a = range(1,10)
print([*zip(a,coef,score)])


fig = plt.figure(figsize=(15, 8), dpi=100)
al = data['乙醛选择性']
score =[]
coef = []
for i in range(0,4):
        i    =i+1
        poly = PF(degree=i)
        X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
        #  训练数据的拟合
        LinearR_ = LinearRegression().fit(X_, al)
        X_curve = np.linspace(0,300,500).reshape(-1,1)
        X_curve1 = PF(degree=i).fit_transform(X_curve)
        plt.scatter(time,al,color='k')
        y_curve = LinearR_.predict(X_curve1)
        plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
        coef.append(LinearR_.coef_)
        score.append(LinearR_.score(X_,al))
        plt.title("乙醛选择性随时间变化拟合图像")
        plt.xlabel("时间")
        plt.ylabel("乙醛选择性")
        plt.legend()
        plt.show()
a = range(1,10)
print([*zip(a,coef,score)])



from sklearn.preprocessing import PolynomialFeatures as PF
from sklearn.linear_model import LinearRegression
```

```
fig = plt.figure(figsize=(15, 8), dpi=100)
al = data['乙烯选择性']
i = 3
poly = PF(degree=i)
X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
# 训练数据的拟合
LinearR_ = LinearRegression().fit(X_, al)
X_curve = np.linspace(0,450,5000).reshape(-1,1)
X_curve1 = PF(degree=i).fit_transform(X_curve)
plt.scatter(time,al,color='k')
y_curve = LinearR_.predict(X_curve1)
plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
print(LinearR_.coef_)
print(LinearR_.score(X_,al))
plt.title("乙烯选择性随时间变化拟合图像")
plt.xlabel("时间")
plt.ylabel("乙烯选择性")
plt.legend()
plt.show()

from sklearn.preprocessing import PolynomialFeatures as PF
from sklearn.linear_model import LinearRegression
fig = plt.figure(figsize=(15, 8), dpi=100)
al = data['乙醛选择性']
i = 3
poly = PF(degree=i)
X_ = poly.fit_transform(np.array(time).reshape(-1, 1))
# 训练数据的拟合
LinearR_ = LinearRegression().fit(X_, al)
X_curve = np.linspace(0,450,5000).reshape(-1,1)
X_curve1 = PF(degree=i).fit_transform(X_curve)
plt.scatter(time,al,color='k')
y_curve = LinearR_.predict(X_curve1)
plt.plot(X_curve,y_curve,label='degree{a}'.format(a = i))
print(LinearR_.coef_)
print(LinearR_.score(X_,al))
plt.title("乙醛选择性随时间变化拟合图像")
plt.xlabel("时间")
plt.ylabel("乙醛选择性")
plt.legend()
plt.show()
```

附录 3

介绍：问题 2.lightgbm 训练代码

```
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score as CVS, train_test_split as
TTS
import matplotlib.pyplot as plt
from lightgbm import LGBMRegressor as LGBR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error as MSE
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
data = pd.read_excel(r'C:\Users\86139\Desktop\B\原始数据.xlsx')


X = data.iloc[:,1:7]
Y = data.iloc[:,7]
Xtrain,Xtest,Ytrain,Ytest = TTS(X,Y,test_size=0.1,random_state=420)



#Step 1 调整 max_depth 和 num_leaves
parameters = {
    'max_depth': [2,4,6,8],
    'num_leaves': range(11,30),
}
gbm =           LGBR(objective = 'regression',
                     is_unbalance = True,
                     max_depth = 6,
                     num_leaves = 40,
                     learning_rate = 0.1,
                     feature_fraction = 0.7,
                     min_child_samples=21,
                     min_child_weight=0.001,
                     bagging_fraction = 1,
                     bagging_freq = 2,
                     reg_alpha = 0.001,
                     reg_lambda = 8,
                     cat_smooth = 0,
                     num_iterations = 200,
                    )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
```

```
print('最佳模型得分:{0}'.format(gsearch.best_score_))

#Step2  调整 min_data_in_leaf  和  min_sum_hessian_in_leaf
parameters = {
'min_child_samples': range(15,20),
'min_child_weight':      [i/1000 for i in range(10)]
}

gbm =                LGBR(objective = 'regression',
                                is_unbalance = True,
                                max_depth = 2,
                                num_leaves = 11,
                                learning_rate = 0.1,
                                feature_fraction = 0.7,
                                min_child_samples=21,
                                min_child_weight=0.001,
                                bagging_fraction = 1,
                                bagging_freq = 2,
                                reg_alpha = 0.001,
                                reg_lambda = 8,
                                cat_smooth = 0,
                                num_iterations = 200,
                               )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))

#Step2  调整 min_data_in_leaf  和  min_sum_hessian_in_leaf
parameters = {
      'feature_fraction': [i/10 for i in range(1,10)],
}
gbm =                LGBR(objective = 'regression',
                                is_unbalance = True,
                                max_depth = 2,
                                num_leaves = 11,
                                learning_rate = 0.1,
                                feature_fraction = 0.7,
                                min_child_samples=15,
                                min_child_weight=0.0,
                                bagging_fraction = 1,
                                bagging_freq = 2,
                                reg_alpha = 0.001,
                                reg_lambda = 8,
```

```
                                    cat_smooth = 0,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))

#Step2  调整 min_data_in_leaf  和  min_sum_hessian_in_leaf
parameters = {
        'bagging_fraction': [i/10 for i in range(1,11)],
        'bagging_freq': [i for i in range(1,11)],
}
gbm =                LGBR(objective = 'regression',
                                    is_unbalance = True,
                                    max_depth = 2,
                                    num_leaves = 11,
                                    learning_rate = 0.1,
                                    feature_fraction = 0.7,
                                    min_child_samples=15,
                                    min_child_weight=0.0,
                                    bagging_fraction = 1,
                                    bagging_freq = 2,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 0,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))

#Step2  调整 min_data_in_leaf  和  min_sum_hessian_in_leaf
parameters = {
        'cat_smooth': [i*5 for i in range(1,10)],
}
gbm =                LGBR(objective = 'regression',
                                    is_unbalance = True,
                                    max_depth = 2,
                                    num_leaves = 11,
                                    learning_rate = 0.1,
                                    feature_fraction = 0.7,
                                    min_child_samples=15,
```

```
                                    min_child_weight=0.0,
                                    bagging_fraction = 0.9,
                                    bagging_freq = 5,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 0,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))

#Step 7 最后，本人会适当调小 learning_rate 的值以及调整 num_iterations 的大小。
parameters = {
        'learning_rate': [i/10 for i in range(10)],
      'num_iterations': [(i+1)*100 for i in range(10) ]
}
gbm =                LGBR(objective = 'regression',
                                    is_unbalance = True,
                                    max_depth = 2,
                                    num_leaves = 11,
                                    learning_rate = 0.1,
                                    feature_fraction = 0.7,
                                    min_child_samples=15,
                                    min_child_weight=0.0,
                                    bagging_fraction = 0.9,
                                    bagging_freq = 5,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 5,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))


reg = gsearch.best_estimator_
plt.figure(figsize=(20, 10), dpi=100)
plt.bar(Xtrain.columns, reg.feature_importances_)
plt.xticks(range(Xtrain.shape[1]))
plt.ylim(0,0.6)
```

```
plt.xlabel("feature name")
plt.ylabel("feature importance")
plt.title("温度与不同催化剂对对 C4 烯烃选择性(%)影响的重要性")
plt.rcParams['figure.figsize'] = (20.0, 8.0)
print(reg)
print("R^2",reg.score(Xtest,Ytest))#R^2 评估指标
print("MSE",MSE(Ytest,reg.predict(Xtest)))
print("importance",reg.feature_importances_)

import pandas as pd
from sklearn.model_selection import KFold, cross_val_score as CVS, train_test_split as
TTS
import matplotlib.pyplot as plt
from lightgbm import LGBMRegressor as LGBR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error as MSE
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
data = pd.read_excel(r'C:\Users\86139\Desktop\B\原始数据.xlsx')



X = data.iloc[:,1:7]
Y = data.iloc[:,9]
Xtrain,Xtest,Ytrain,Ytest = TTS(X,Y,test_size=0.1,random_state=420)



#因为 LightGBM 使用的是 leaf-wise 的算法，因此在调节树的复杂程度时，使用的
是 num_leaves 而不是 max_depth。大致换算关系：num_leaves = 2^(max_depth)，但
是它的值的设置应该小于 2^(max_depth)
parameters = {
    'max_depth': [i for i in range(1,5)],
    'num_leaves': range(1,20),
}
gbm =           LGBR(objective = 'regression',
                     is_unbalance = True,
                     max_depth = 6,
                     num_leaves = 40,
                     learning_rate = 0.1,
                     feature_fraction = 0.7,
                     min_child_samples=21,
                     min_child_weight=0.001,
```

```
                                    bagging_fraction = 1,
                                    bagging_freq = 2,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 0,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))



#Step2  调整 min_data_in_leaf  和  min_sum_hessian_in_leaf
#一个叶子上的最小数据量。
#一个叶子上的最小 hessian 和。默认设置为 0.001，一般设置为 1。不建议调整，增大数值会得到较浅的树深
parameters = {
'min_child_samples': range(1,20),
'min_child_weight':      [i/1000 for i in range(10)]
}
gbm =                LGBR(objective = 'regression',
                                    is_unbalance = True,
                                    max_depth = 1,
                                    num_leaves = 2,
                                    learning_rate = 0.1,
                                    feature_fraction = 0.7,
                                    min_child_samples=21,
                                    min_child_weight=0.001,
                                    bagging_fraction = 1,
                                    bagging_freq = 2,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 0,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))
```

```
parameters = {
      'feature_fraction': [i/10 for i in range(1,10)],
}
gbm =                LGBR(objective = 'regression',
                                  is_unbalance = True,
                                  max_depth = 1,
                                  num_leaves = 2,
                                  learning_rate = 0.1,
                                  feature_fraction = 0.7,
                                  min_child_samples=10,
                                  min_child_weight=0.0,
                                  bagging_fraction = 1,
                                  bagging_freq = 2,
                                  reg_alpha = 0.001,
                                  reg_lambda = 8,
                                  cat_smooth = 0,
                                  num_iterations = 200,
                                 )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))



parameters = {
       'bagging_fraction': [i/10 for i in range(1,11)],
       'bagging_freq': [i for i in range(1,11)],
}
gbm =                LGBR(objective = 'regression',
                                  is_unbalance = True,
                                  max_depth = 1,
                                  num_leaves = 2,
                                  learning_rate = 0.1,
                                  feature_fraction = 0.3,
                                  min_child_samples=10,
                                  min_child_weight=0.0,
                                  bagging_fraction = 1,
                                  bagging_freq = 2,
                                  reg_alpha = 0.001,
                                  reg_lambda = 8,
                                  cat_smooth = 0,
                                  num_iterations = 200,
                                 )
```

```
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))



parameters = {
        'cat_smooth': [i*5 for i in range(1,10)],
}
gbm =                   LGBR(objective = 'regression',
                                        is_unbalance = True,
                                        max_depth = 1,
                                        num_leaves = 2,
                                        learning_rate = 0.1,
                                        feature_fraction = 0.3,
                                        min_child_samples=10,
                                        min_child_weight=0.0,
                                        bagging_fraction = 0.9,
                                        bagging_freq = 9,
                                        reg_alpha = 0.001,
                                        reg_lambda = 8,
                                        cat_smooth = 0,
                                        num_iterations = 200,
                                       )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))



parameters = {
        'learning_rate': [i/10 for i in range(10)],
      'num_iterations': [(i+1)*100 for i in range(5) ]
}
gbm =                   LGBR(objective = 'regression',
                                        is_unbalance = True,
                                        max_depth = 1,
                                        num_leaves = 2,
                                        learning_rate = 0.1,
                                        feature_fraction = 0.3,
                                        min_child_samples=10,
                                        min_child_weight=0.0,
```

```
                                    bagging_fraction = 0.9,
                                    bagging_freq = 9,
                                    reg_alpha = 0.001,
                                    reg_lambda = 8,
                                    cat_smooth = 5,
                                    num_iterations = 200,
                                   )
gsearch = GridSearchCV(gbm, param_grid=parameters, scoring='r2', cv=5)
gsearch.fit(Xtrain, Ytrain)
print('参数的最佳取值:{0}'.format(gsearch.best_params_))
print('最佳模型得分:{0}'.format(gsearch.best_score_))



reg = gsearch.best_estimator_
plt.figure(figsize=(20, 10), dpi=100)
plt.bar(Xtrain.columns, reg.feature_importances_)
plt.xticks(range(Xtrain.shape[1]))
plt.ylim(0,0.6)
plt.xlabel("feature name")
plt.ylabel("feature importance")
plt.title("温度与不同催化剂对对 C4 烯烃选择性(%)影响的重要性")
plt.rcParams['figure.figsize'] = (20.0, 8.0)
print(reg)
print("R^2",reg.score(Xtest,Ytest))#R^2 评估指标
print("MSE",MSE(Ytest,reg.predict(Xtest)))
print("importance",reg.feature_importances_)
```

附录 4

介绍：问题 2. xgboost 训练代码

```
from xgboost import XGBRegressor as XGBR
from sklearn.model_selection import KFold, cross_val_score as CVS, train_test_split as
TTS
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV
import pandas as pd
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
from numpy import nan as NA
import pickle
```

```
data = pd.read_excel(r'C:\Users\86139\Desktop\B\原始数据.xlsx')
X = data.iloc[:,1:7]
Y = data.iloc[:,7]
Xtrain,Xtest,Ytrain,Ytest = TTS(X,Y,test_size=0.1,random_state=420)



#第二步：调整树结构
#和 GBM 中的参数相同，这个值为树的最大深度。
#决定最小叶子节点样本权重和，这个参数用于避免过拟合。当它的值较大时，可以避免模型学习到局部的特殊样本。但是如果这个值过高，会导致欠拟合。这个参数需要使用 CV 来调整。
param_test1 = {
 'max_depth':range(3,10,2),
 'min_child_weight':range(2,7,2)
}
gsearch1 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140, max_depth=5,
 min_child_weight=1, gamma=0, subsample=0.8, colsample_bytree=0.8, objective= 'reg:linear',
 nthread=4, scale_pos_weight=1, seed=27),
 param_grid = param_test1, scoring='r2',n_jobs=4, cv=5)
gsearch1.fit(Xtrain,Ytrain)
gsearch1.best_params_, gsearch1.best_score_



#在节点分裂时，只有分裂后损失函数的值下降了，才会分裂这个节点。Gamma 指定了节点分裂所需的最小损失函数下降值。
#这个参数的值越大，算法越保守。这个参数的值和损失函数息息相关，所以是需要调整的
param_test3 = {
 'gamma':[i/100.0 for i in range(0,100)]
}
gsearch1 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8, objective= 'reg:linear',
 nthread=4, scale_pos_weight=1, seed=27),
 param_grid = param_test3, scoring='r2',n_jobs=4, cv=5)
gsearch1.fit(Xtrain,Ytrain)
gsearch1.best_params_, gsearch1.best_score_
```

```
#这个参数控制对于每棵树，随机采样的比例。减小这个参数的值，算法会更加保守，避免过拟合。但是，如果这个值设置得过小，它可能会导致欠拟合。
#colsample_bytree[默认 1]和 GBM 里面的 max_features 参数类似。用来控制每棵随机采样的列数的占比(每一列是一个特征)。
param_test4 = {
 'subsample':[i/10.0 for i in range(1,10)],
 'colsample_bytree':[i/10.0 for i in range(1,10)]
}
gsearch4 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140, max_depth=3,
 min_child_weight=2, gamma=0.8, subsample=0.8, colsample_bytree=0.8, objective= 'reg:linear',
 nthread=4, scale_pos_weight=1, seed=27),
 param_grid = param_test4, scoring='r2',n_jobs=4, cv=5)
gsearch4.fit(Xtrain,Ytrain)
gsearch4.best_params_, gsearch4.best_score_


#第五步 调整正则项
param_test5 = {
 'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05]
}
gsearch5 = GridSearchCV(estimator = XGBR( learning_rate =0.1, n_estimators=140, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.6, colsample_bytree=0.8,
 objective= 'reg:linear', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test5, scoring='r2',n_jobs=4, cv=5)
gsearch5.fit(Xtrain,Ytrain)
gsearch5.best_params_, gsearch5.best_score_


#第六步 调整学习速率
param_test6 = {
 'learning_rate':[0, 0.001, 0.005, 0.01, 0.05,0.1,0.5,1]
}
gsearch5 = GridSearchCV(estimator = XGBR( learning_rate =0.1, n_estimators=140, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.6, colsample_bytree=0.8,
 objective= 'reg:linear', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test6, scoring='r2',n_jobs=4, cv=5)
gsearch5.fit(Xtrain,Ytrain)
gsearch5.best_params_, gsearch5.best_score_
```

```
reg = gsearch5.best_estimator_
plt.figure(figsize=(20, 10), dpi=100)
plt.bar(Xtrain.columns, reg.feature_importances_)
plt.xticks(range(Xtrain.shape[1]))
plt.ylim(0,0.6)
plt.xlabel("feature name")
plt.ylabel("feature importance")
plt.title("温度与不同催化剂对乙醇转化率(%)影响的重要性")
plt.rcParams['figure.figsize'] = (20.0, 8.0)
print(reg)
print("R^2",reg.score(Xtest,Ytest))#R^2 评估指标
print("MSE",MSE(Ytest,reg.predict(Xtest)))
print("importance",reg.feature_importances_)


from xgboost import XGBRegressor as XGBR
from sklearn.model_selection import KFold, cross_val_score as CVS, train_test_split as
TTS
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV
import pandas as pd
# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
from numpy import nan as NA
import pickle


data = pd.read_excel(r'C:\Users\86139\Desktop\B\原始数据.xlsx')
X = data.iloc[:,1:7]
Y = data.iloc[:,9]
Xtrain,Xtest,Ytrain,Ytest = TTS(X,Y,test_size=0.1,random_state=420)
#第二步：调整树结构


param_test1 = {
 'max_depth':range(3,10,2),
 'min_child_weight':range(2,7,2)
}
gsearch1 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140,
max_depth=5,
 min_child_weight=1, gamma=0.29, subsample=0.8, colsample_bytree=0.8, objective=
```

```
'reg:linear',
  nthread=4, scale_pos_weight=1, seed=27),
  param_grid = param_test1, scoring='r2',n_jobs=4, cv=5)
gsearch1.fit(Xtrain,Ytrain)
gsearch1.best_params_, gsearch1.best_score_

param_test3 = {
  'gamma':[i/100.0 for i in range(0,100)]
}
gsearch1 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140,
max_depth=3,
  min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.6, objective=
'reg:linear',
  nthread=4, scale_pos_weight=1, seed=27),
  param_grid = param_test3, scoring='r2',n_jobs=4, cv=5)
gsearch1.fit(Xtrain,Ytrain)
gsearch1.best_params_, gsearch1.best_score_

param_test4 = {
  'subsample':[i/10.0 for i in range(6,10)],
  'colsample_bytree':[i/10.0 for i in range(6,10)]
}
gsearch4 = GridSearchCV(estimator =XGBR( learning_rate =0.1, n_estimators=140,
max_depth=3,
  min_child_weight=2, gamma=0.8, subsample=0.8, colsample_bytree=0.8, objective=
'reg:linear',
  nthread=4, scale_pos_weight=1, seed=27),
  param_grid = param_test4, scoring='r2',n_jobs=4, cv=5)
gsearch4.fit(Xtrain,Ytrain)
gsearch4.best_params_, gsearch4.best_score_

#第五步 调整正则项
param_test5 = {
  'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05]
}
gsearch5 = GridSearchCV(estimator = XGBR( learning_rate =0.1, n_estimators=140,
max_depth=3,
  min_child_weight=2, gamma=0, subsample=0.6, colsample_bytree=0.8,
  objective= 'reg:linear', nthread=4, scale_pos_weight=1,seed=27),
  param_grid = param_test5, scoring='r2',n_jobs=4, cv=5)
gsearch5.fit(Xtrain,Ytrain)
gsearch5.best_params_, gsearch5.best_score_

#第六步 调整学习速率
```

```
param_test6 = {
 'learning_rate':[0, 0.001, 0.005, 0.01, 0.05,0.1,0.5,1]
}
gsearch5 = GridSearchCV(estimator = XGBR( learning_rate =0.1, n_estimators=140, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.6, colsample_bytree=0.8,
 objective= 'reg:linear', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test6, scoring='r2',n_jobs=4, cv=5)
gsearch5.fit(Xtrain,Ytrain)

reg = gsearch5.best_estimator_
plt.figure(figsize=(20, 10), dpi=100)
plt.bar(Xtrain.columns, reg.feature_importances_)
plt.xticks(range(Xtrain.shape[1]))
plt.ylim(0,0.6)
plt.xlabel("feature name")
plt.ylabel("feature importance")
plt.title("温度与不同催化剂对乙醇转化率(%)影响的重要性")
plt.rcParams['figure.figsize'] = (20.0, 8.0)
print(reg)
print("R^2",reg.score(Xtest,Ytest))#R^2 评估指标
print("MSE",MSE(Ytest,reg.predict(Xtest)))
print("importance",reg.feature_importances_)
```

附录 5

介绍：支撑材料列表

1. 附件-问题 1
   - 拟合结果：问题 1.中得到的拟合结果的图形
   - 拟合参数：问题 1.中得到的拟合结果的参数
   - 问题 1 第二问.py：问题 1.第二问用于线性回归的以 python 编写的代码
2. 附件-问题 2
   - lightgbm 训练 1.py、lightgbm 训练 2.py：问题 2.用于 lightGBM 模型的以 python 编写的代码
   - xgboost 训练 1.py、xgboost 训练 2.py：问题 2.用于 XGBoost 模型的以 python 编写的代码
   - cuihuaji_yichun_2.dat：XGBoost 预测乙醇转化率的模型
   - cuihuaji_c4_gbm_2.dat：lightGBM 预测 C4 烯烃选择性的模型
3. 吕绍沛-乙醇偶合制备丁醇及 C4 烯烃.pdf
   - 参考文献[1]中提到的本论文查阅参考的论文资料