2024 年江苏省研究生数学建模科研创新实践大赛

题 目　　人造革性能优化设计研究

摘　　要

人造革作为一种高分子材料，凭借其与真皮相似的质感和外观，以及优良的性能，被广泛应用于多个领域。随着人们对高品质和环保材料需求的不断增长，如何提升人造革的性能成为了生产者的重要研究方向。本文旨在研究人造革生产工艺参数对其性能的影响，并探讨如何通过优化工艺参数获得最优性能的人造革产品。

针对问题一，首先进行**数据预处理**，包括数据合并、探索性数据分析与数据标准化等。接着，通过绘制**散点图**评估产品性能与工艺参数之间、产品性能之间的初步关联性。鉴于样本数据的稀缺，采用**斯皮尔曼相关系数**量化相关性，并进一步利用**灰色关联分析**深入探究各因素间的关联度。在分析工艺参数对性能指标的交互作用时，结合**逐步回归**和**三因素方差分析**，细致研究不同工艺参数的组合对产品性能的影响，并评估这些交互作用的**显著性和影响程度**。结果表明，**树脂含量与透湿率**存在高度负相关（-0.8, $P < 0.01$）；**碱减量程度与柔软度、急弹角度**存在高度正相关（0.9, $P < 0.01$），**与撕裂强力**呈现高度负相关（-0.9, $P < 0.01$）；**固化温度对急弹角度**影响最大。同时，**属于一个大类的性能指标间存在最强的相互影响**。例如，力学性能的断裂强力、断裂伸长率和撕裂强力表现出显著联系；柔软性能的柔软度和急弹角度也显示出紧密的相关性；然而，**热湿舒适性能**的透气率和透湿率之间的关联性却异常低。在交互作用上，**树脂含量、固化温度以及碱减量程度**三者的交互影响十分显著（$F = 245.339$, $P = 0.000$）；其次是**固化温度和碱减量程度**的交互作用（$F = 188.271$, $P = 0.000$）；而**树脂含量和固化温度、碱减量程度**组合的交互作用较低。

针对问题二，考虑到工艺参数对性能指标存在复杂的交互作用，为了更好地抓住数据间的**非线性特征**，采用**机器学习算法**，选择逻辑回归、朴素贝叶斯、决策树、

随机森林、支持向量机、多层感知机、AdaBoost、XGBoost、CatBoost 和 LightGBM 等 10 种回归模型分别对不同的性能指标进行拟合。在模型训练时，按 7:3 的比例把数据随机划分为训练集和测试集，并使用**五折交叉验证**和 RMSE、MAE、R^2 等指标评估模型效果，进而确定最优模型。结果表明，**XGBoost 模型**在断裂强力、断裂伸长率、透湿率和柔软度预测方面表现最为出色（RMSE = 0.134，R^2 = 0.985），而**随机森林模型**在撕裂强力、透气率和折皱回复角的预测上达到了最佳效果（RMSE = 0.172，R^2 = 0.941）。考虑到**模型的非线性和搜索空间的复杂性**，以各回归模型为目标函数，使用**粒子群优化算法**进行求解，找出最优性能下各自的最佳工艺参数，结果见**第 18 页表 7**。

针对问题三，在问题二的基础上，针对多个性能指标进行**多目标优化**。考虑到直接求解多目标优化模型的复杂性，采用**熵权法**对标准化后的性能指标进行客观赋权，将**多个目标函数转化为单一目标函数**进行求解。对力学性能（断裂强力、断裂伸长率、撕裂强力）、热湿舒适性能（透气率、透湿率）、柔软性能（柔软度、折皱回复角）以及综合性能（所有七个指标）进行赋权，形成具有代表性的四个**新性能指标**。随后，重复问题二中的模型选择和评估步骤确定最优回归模型，并使用**粒子群优化算法**找出最优性能所需要的工艺参数，详见**表 8 至表 11**。结果表明，在预测力学性能和热湿舒适性能方面，**XGBoost 模型**表现卓越（RMSE = 0.103，R^2 = 0.951）；**随机森林模型**在柔软性能预测上表现最佳（RMSE = 0.148，R^2 = 0.912）；**LightGBM 模型**在综合性能预测上达到了最优效果（RMSE = 0.127，R^2 = 0.926）。比较发现，**树脂含量**波动在 15.76 wt%至 26.74 wt%，**固化温度**在 100.73 ℃至 128.34 ℃间变化，而**碱减量程度**变化幅度最大，从 0.01 至 0.29。专注于单一性能的优化模型虽然在**特定性能**上表现出色，但对**其他性能**有所牺牲。

针对问题四，考虑到性能指标存在优先级，采用**主客观相结合**的赋权方法。对于三大性能指标，利用**层次分析法**进行主观赋权；对于各子指标，采用**熵权法**客观赋权。综合权重，构建出**新的性能指标**，将问题**转化**为单目标优化模型，并重复问题二中的流程进行优化。求出最佳性能人造革的**工艺参数组合**为：树脂含量 21.46 wt%、固化温度 113.15 ℃、碱减量程度 0.04。各性能指标的最优值见**第 23 页**。

最后，通过**灵敏度分析**评估了层次分析法在确定人造革**热湿舒适性能权重**时的主观性对回归模型预测精度的影响。结果显示，即使权重在原始值的±10%波动，**大多数**性能指标预测稳定，仅透湿率的预测存在部分偏差。分析结果证明了使用层次分析法的**合理性**，表明由**主观性**引起的模型预测偏差较小。

关键词：人造革性能优化；斯皮尔曼相关系数；灰色关联分析；逐步回归；三因素方差分析；XGBoost；随机森林；LightGBM；熵权法；粒子群优化算法

目 录

一、问题重述

1.1 问题背景

人造革是一种由高分子材料（树脂）制成的纤维状物质，通过固化处理等工艺后具有类似皮革的质感和外观。这种材料不仅外观上仿真皮革，还具备高强度、高耐磨和柔软的特点，因而在服装、鞋类、家具等多个领域得到了广泛应用。随着人们对高品质和环保材料的需求增加，生产优质人造革产品成为了人造革生产者孜孜不倦的追求。

为了生产出性能优异的人造革，生产者们依据生产经验，进行了针对性的实验研究。在实验中，针刺非织造布（基布）经过定型、树脂含浸、碱减量等工序处理后，制备出不同类型的人造革。研究重点关注工艺参数对人造革性能的影响，这些参数包括实验室内可控的温度、压力、时间等条件。通过一系列的性能测试，对人造革的力学性能（如断裂强力、断裂伸长率、撕裂强力）、热湿舒适性能（如透气率、透湿率）以及柔软性能（如柔软度、折皱回复角）进行全面评估。对于力学性能而言，优质的人造革需要具备较高的抗断裂和抗撕裂能力，以确保其在使用过程中的耐久性和可靠性；柔软性能方面，要求织物手感柔软，折皱回复角大，以提高穿着的舒适度和美观度；热湿舒适性能则强调透气率和透湿率的高低，这些性能直接关系到穿着时的舒适体验，高透气率和透湿率能够使水蒸气快速排出，增强织物的热舒适性。通过对工艺参数的优化调整，生产者们可以开发出更加优质、舒适且耐用的人造革产品，以满足市场和消费者的需求。

1.2 题目重述

根据附件中提供的在不同工艺参数（树脂含量、固化温度和碱减量程度）下制备的人造革产品实验数据，参考有关文献并结合专业背景，建立相关数学模型解决以下四个问题：

1. 分析工艺参数与产品性能之间和不同产品性能之间可能存在的关联性，以及工艺参数之间可能对人造革的性能产生的交互作用；

2. 在问题 1 基础上，分别建立工艺参数与人造革 7 种性能之间的关系模型并阐明建模理由。通过关系模型，找出最优断裂强力、最优断裂伸长率、最优撕裂强力、最优透气率、最优透湿率、最优柔软度、最优折皱回复角各自的最佳工艺参数；

3. 建立数学模型，分别分析实现最优力学性能、最优热湿舒适性和最优柔软性能所需的工艺参数；同时建立数学模型，分析使得人造革 7 项指标综合性能最优所需的工艺参数；将以上两个模型的结果进行比较；

4. 基于“优先满足热湿舒适性，其次考虑力学性能和柔软性能，同时要求各项性能指标尽量接近问题 2 中的最佳水平”的要求，为某沙漠科考队设计最佳性能的人造革。

1.3 文献综述

人造革的生产工艺主要包括定型、树脂含浸、碱减量等步骤，每一步工艺都会对最终产品的性能产生重要影响。主要的性能指标包括力学性能（断裂强力、断裂伸长率、撕裂强力）、热湿舒适性能（透气率、透湿率）以及柔软性能（柔软度、折皱回复角）。近年来，许多研究致力于探索工艺参数对人造革性能的影响，并建立了相关数学模型。

树脂含量是影响人造革性能的一个关键参数，树脂的选择和用量决定了人造革的强度、柔软性和透气性[1]。研究表明，适中的树脂含量可以在保持良好力学性能的同时不影响其透气和柔软性，而过多或过少的树脂含量都会对人造革的综合性能产生负面影响[2]；固化温度影响树脂的交联程度，从而影响人造革的物理性能[3]。研究指出，适当的固化温度可以提高人造革的断裂强力和撕裂强力，过高的固化温度可能会导致树脂过度交联，使人造革变得过于硬脆[4]。碱减量处理通过溶解部分纤维使织物孔隙率增加，从而提高其透气和透湿性能[5]。研究发现，适度的碱减量可以显著改善人造革的热湿舒适性能，同时对力学性能和柔软性能也有一定的提升作用[6]。通过对现有文献的综述，可以看出树脂含量、固化温度和碱减量程度是影响人造革性能的关键工艺参数。

优质的人造革应具有较高的强度和耐磨性，同时保持柔软度和舒适性，以满足消费者的需求和市场的要求[7,8]。通过对工艺参数的优化调整，生产者可以制备出更加优质、耐用和舒适的人造革产品，从而在竞争激烈的市场中脱颖而出。

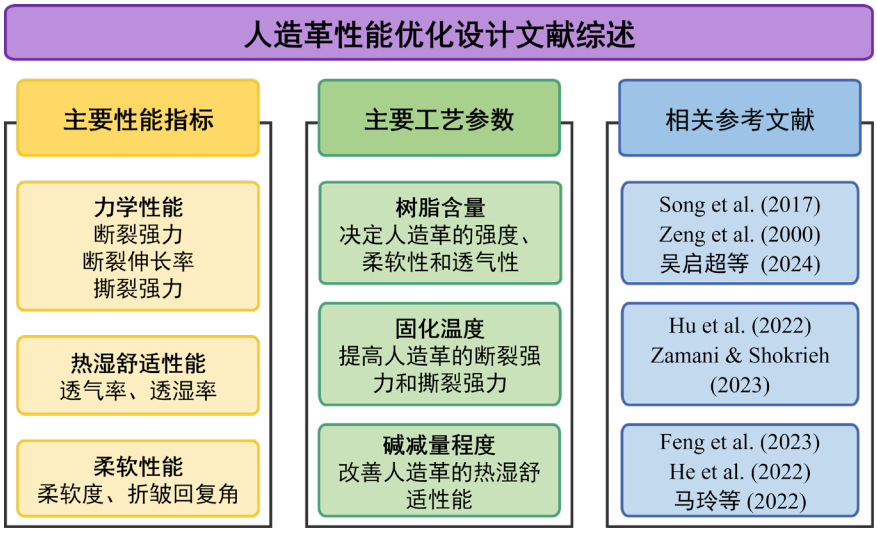

图 1：人造革性能优化设计文献综述

二、模型假设

在建模过程中，为了简化问题并排除无关因素的干扰，根据已有研究和文中提供的条件，提出以下几点假设：

假设 1：人造革生产条件的稳定性假设。

假设在生产过程中，除了研究中考虑的工艺参数外，其他生产条件（如环境温度、湿度、运行设备、人员操作等）保持稳定。这一假设简化建模过程，避免众多复杂且不必要因素的影响。

假设 2：实验数据的代表性和准确性假设。

假设实验数据能够代表实际生产过程中各个工艺参数对人造革性能的影响，即实验数据具有高度的代表性和准确性。这一假设保证了模型的预测结果在实际生产中具有较高的可靠性和应用性。

三、符号与变量说明

为方便描述，现将本文所用到的主要符号与变量说明如下：

符号与变量	相关说明与描述
ρ_s	斯皮尔曼相关系数
ξ	灰色关联系数
γ	灰色关联度
ρ	灰色关联分析分辨系数
MSE	均方误差
$RMSE$	均方根误差
MAE	平均绝对误差
$MAPE$	平均绝对百分比误差

四、数据预处理

考虑到附件所提供的数据存在信息重复的情况，同时各物理量参数的量纲和数量级存在差异，有必要对数据进行合并和标准化处理，以确保数据的一致性和可比性。首先，需要将不同表格中的数据进行合并，去除重复信息；接着，进行初步的探索性数据分析来发现潜在的模式和趋势，为后续的深入研究提供基础；最后，对各物理量参数进行标准化处理，以消除因量纲和数量级不同带来的影响。通过这些步骤，可以提高数据的质量和分析的准确性。

4.1 数据合并

通过观察附件数据与阅读数据相关说明可知，16 组实验均是在不同工艺参数（树脂含量、固化温度和碱减量程度）的四个固定水平下进行，且每组实验重复三次。因此，根据工艺参数将 7 个附件进行合并，结果保存在“*MergedData.csv*”文件中。同时，为了方便后续的分析并考虑到重复实验的特性，采用每组实验三次数据的算术平均值作为待分析数据，最终得到一个 16 行、11 列（3 列工艺参数、7 列人造革性能指标和 1 列实验序号）的数据矩阵，保存在“*ProcessedData.csv*”文件中。

4.2 探索性数据分析

探索性数据分析（Exploratory Data Analysis，EDA）是数据分析过程中一个重要的步骤，其目的是通过可视化和统计方法来了解数据的主要特征和结构[9]。EDA 的主要目标是发现数据中的异常值、关系和分布情况，为后续建模和分析提供依据。

表 1：变量统计描述结果

变量名	最大值	最小值	平均值	标准差	变异系数
树脂含量	30	15	22.5	5.77	0.11
固化温度	130	100	115	11.56	0.1
碱减量程度	0.3	0	0.15	0.12	0.14
断裂强力	1764.94	1146.95	1331.61	235.22	0.07
断裂伸长率	1.09	0.8	0.94	0.08	0.08
撕裂强力	176.27	91.02	117.27	26.15	0.42
透气率	328.35	88.28	183.52	74.16	0.13
透湿率	3505.3	2124.38	2626.77	375.74	0.37
柔软度	3.43	0.85	2.07	0.86	0.03
急弹角度	168.47	149.36	160.03	6.21	0.04

表 1 展示了描述性统计的结果，包括最大值、最小值、平均值、标准差和变异系数等统计量，用于研究定量数据的整体情况。可以看出，撕裂强力和透湿率的变异系数大于 0.15，数据中可能存在异常值，而其余数据均较为稳定。

图 2 展示了树脂含量、固化温度、碱减量程度、断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度、急弹角度频数分析和离散趋势分析的结果。离散趋势用极大值、极小值、25%分位数、中位数、75%分位数等统计指标对数据分布进行差异（稳定性）测量。可以看出大部分变量分布较为稳定，只有个别性能指标（撕裂强力和透湿率）出现了少量异常值，这与表 1 的结果一致。

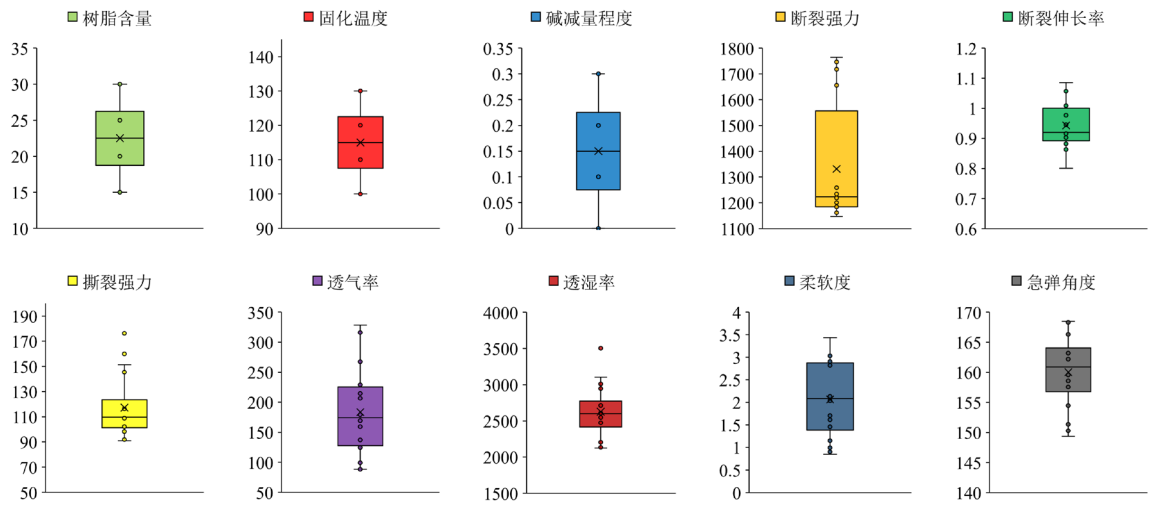

图 2：工艺参数和性能指标箱线图

4.3 数据标准化

为了尽可能减少或消除不同物理量在量纲和数量级上的差异，有必要对数据进行标准化处理。本研究采用 Z 标准化（Z-score Normalization），它是一种常用的数据预处理技术，用于将不同量纲的数据转换到同一量纲[10]。通过 Z 标准化处理后，数据将具有均值为 0、标准差为 1 的正态分布形式。该方法使用以下公式进行标准化：

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

其中，X 为原始数据；μ 为数据均值；σ 为数据标准差；Z 为标准化后的数据。

标准化处理的一个关键优势在于，它不受特征原始尺度的影响，因为这种转换基于每个特征自身的统计属性（均值和标准差）进行。无论原始特征值的大小，经过标准化后它们都具有相同的尺度，从而有助于提高相关模型的性能。标准化后的数据保存在“*NormalizedData.csv*”文件中。

五、问题一：工艺参数与产品性能关联分析模型

5.1 问题一分析

问题一要求分析产品性能间、产品性能与工艺参数的关联性，并探究不同工艺参数对产品性能的交互作用。首先，绘制散点图进行初步的关联性评估，随后考虑到样本数据的稀缺性，采用斯皮尔曼相关系数来量化产品性能间及产品性能与工艺参数间的相关性。进一步，运用灰色关联分析来深化对各因素间关联度的探究。在工艺参数的交互作用分析方面，结合逐步回归分析与三因素方差分析，细致研究不同工艺参数组合对产品性能的交互影响，并评估这些交互作用的显著性和对产品性能的具体影响程度。问题一思路流程如图 3 所示。

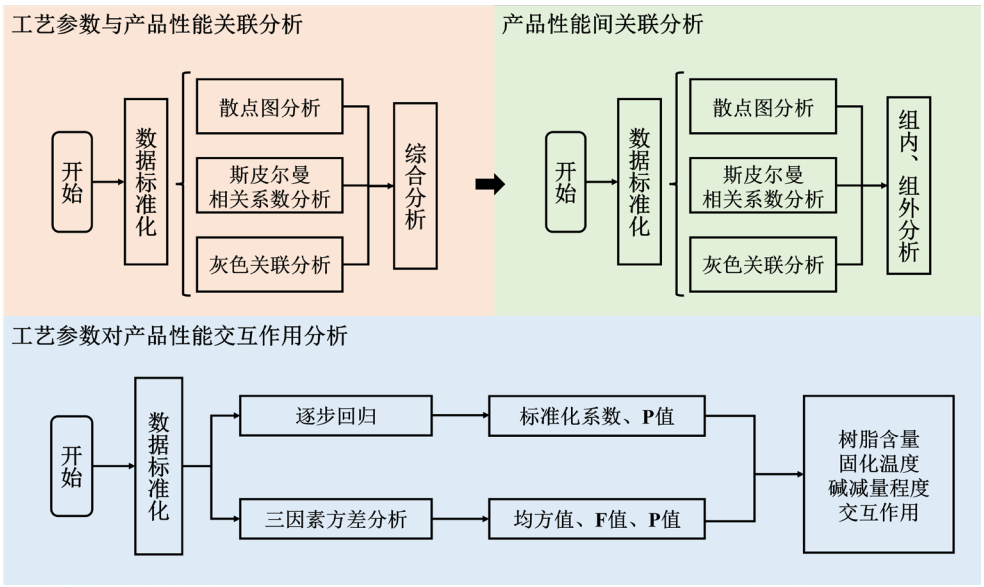

图 3：问题一思路流程

5.2 工艺参数与产品性能关联分析

5.2.1 散点图分析

为了直观方便地了解工艺参数与产品性能间的关系，分别绘制树脂含量、固化温度和碱减量程度与 7 个产品性能的散点关系如图 4。可以初步看出，透湿率（黑正方形）随树脂含量的升高明显下降（图 4a）；断裂强力（蓝三角）随碱减量程度的升高而明显下降（图 4c）；透气率（黄叉号）随树脂含量升高上升（图 4d），随固化温度与碱减量程度升高而降低（图 4e 和图 4f）；撕裂强力（紫星号）随碱减量程度升高而下降（图 4f）。

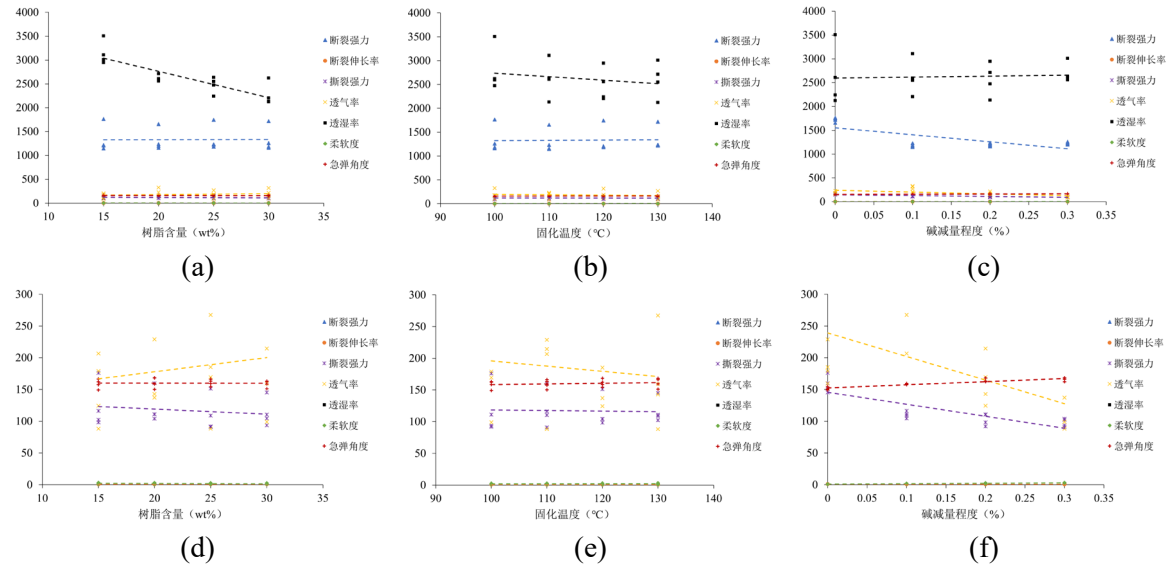

图 4：不同工艺参数，即树脂含量（a）、固化温度（b）、碱减量程度（c）与人造革产品性能散点关系图，（d–f）分别为（a–c）的局部放大。不同颜色虚线为各产品性能的线性趋势线。

5.2.2 斯皮尔曼相关系数分析

斯皮尔曼相关系数（Spearman Correlation Coefficient）是一种非参数统计量，用于衡量两个变量之间的单调关系强度和方向。与皮尔逊相关系数不同，斯皮尔曼相关系数不要求变量是正态分布的，而是基于变量的秩次进行计算[11]。考虑到样本的稀少，本文采用更为稳妥的斯皮尔曼相关系数进行关联性分析，其计算公式为：

$$\rho_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{2}$$

其中，ρ_s为斯皮尔曼相关系数；n为样本数；d_i为每对观测值的秩次差。

斯皮尔曼相关系数通过将每对观测值的秩次转换为秩次差的平方和来衡量两个变量的相关性，在数据中存在异常值或者数据不服从正态分布时特别有用，因为它不依赖于具体的数据分布。图 5 为工艺参数与产品性能的斯皮尔曼相关系数热图，其中的字母分别代表：树脂含量（X1）、固化温度（X2）、碱减量程度（X3）、断裂强力（Y1）、断裂伸长率（Y2）、撕裂强力（Y3）、透气率（Y4）、透湿率（Y5）、柔软度（Y6）和急弹角度（Y7）。

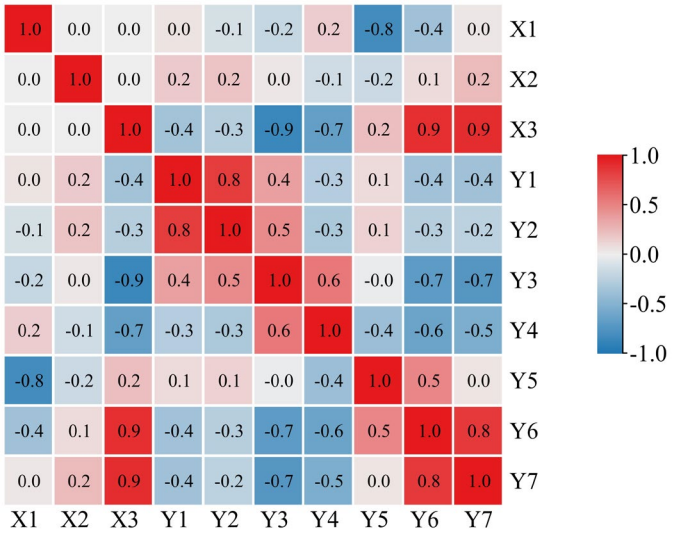

图 5：工艺参数与产品性能斯皮尔曼相关系数热图（已进行 99%水平显著性检验）

上述热图显示，树脂含量与透湿率存在高度负相关关系（-0.8, $P < 0.01$）；碱减量程度与柔软度、急弹角度存在高度正相关（0.9, $P < 0.01$），而与撕裂强力（-0.9, $P < 0.01$）、透气率（-0.7, $P = 0.005$）呈现高度负相关。这些斯皮尔曼相关系数均已通过99%水平的显著性检验，具有高度的统计显著性。此外，注意到固化温度与所有性能间的关联度都较低，因此仅凭斯皮尔曼相关系数可能无法得出全面而准确的结论。

5.2.3 灰色关联分析

灰色关联分析（Grey Relational Analysis, GRA）是灰色系统理论中的一种分析方法，主要用于多因素、多指标的复杂系统分析[12]。GRA 通过评估各因素间的关联度来揭示系统内部的关联结构和动态特性，尤其适用于数据不完备或信息不充分的情况。其具体操作流程如下：

首先，计算参考序列与各比较序列之间的关联系数：

$$\xi_{ij} = \frac{\Delta_{\min} + \rho \Delta_{\max}}{\Delta_{ij} + \rho \Delta_{\max}} \tag{3}$$

$$\Delta_{ij} = |x_0(j) - x_i(j)| \tag{4}$$

其中，Δ_{ij} 为参考序列与比较序列的差值；$\Delta_{\min}$ 和 $\Delta_{\max}$ 分别表示所有序列差值的最小值和最大值；ρ 为分辨系数，经验上一般取 0.5。

接着，对每个比较序列的关联系数求平均，得到灰色关联度：

$$\gamma_i = \frac{1}{n} \sum_{j=1}^{n} \xi_{ij} \tag{5}$$

其中，γ_i 为第 i 比较序列的灰色关联度；n 为指标个数。根据灰色关联度对各比较序列进行排序，关联度越大，说明该序列与参考序列的关系越密切。通过排序结果可以分析各因素对研究对象的影响程度。在本题中，依次取标准化后工艺参数（树脂含量、固化温度、碱减量程度）为参考列，取标准化后性能指标为比较序列。灰色关联分析结果见下表。

表 2：工艺参数与性能指标灰色关联分析结果

X1	关联度	排名	X2	关联度	排名	X3	关联度	排名
Y4	0.695	1	Y7	0.696	1	Y7	0.879	1
Y1	0.676	2	Y6	0.678	2	Y6	0.861	2
Y3	0.664	3	Y2	0.673	3	Y5	0.707	3
Y7	0.653	4	Y3	0.672	4	Y2	0.626	4
Y2	0.650	5	Y1	0.666	5	Y1	0.614	5
Y6	0.626	6	Y4	0.641	6	Y3	0.611	6
Y5	0.601	7	Y5	0.637	7	Y4	0.577	7

表 2 表明，树脂含量（X1）对透气率（Y4）的影响最大（0.695），其次是断裂强力（Y1）和撕裂强力（Y3）；固化温度（X2）对急弹角度（Y7）影响最大（0.696），柔软度（Y6）和断裂伸长率（Y2）其次；碱减量程度（X3）对急弹角度（Y7）和柔软度（Y6）影响很大（> 0.8），其次是透湿率（Y5）。灰色关联分析与斯皮尔曼相关系数的分析结果表现出较高的一致性，同时灰色关联分析在某些方面提供了更为全面和深入的视角。

5.3 产品性能间关联分析

在完成了对工艺参数与产品性能之间关系的深入分析之后，接下来将探讨产品性能之间的相互关联性。通过继续应用斯皮尔曼相关系数和灰色关联分析（GRA）这两种方法，进一步揭示产品性能指标之间的内在联系和相互影响，这对于理解产品性能的综合表现和优化产品的整体性能至关重要。

5.3.1 关联性分析

与 5.2.2 和 5.2.3 节类似，在这里仅考虑 7 个性能指标间的斯皮尔曼相关系数，结果仍见图 5。通过热图可以观察到，断裂强力（Y1）和断裂伸长率（Y2）存在高度正相关（0.8, $P < 0.01$）；撕裂强力（Y3）和急弹角度（Y7）之间呈现强负相关（-0.7, $P = 0.002$）；透气率（Y4）和撕裂强力（Y3）存在中等正相关（0.6, $P = 0.03$），和柔软度（Y6）为中等负相关（-0.6, $P = 0.01$）；柔软度（Y6）和急弹角度（Y7）有强正相关（0.8, $P < 0.001$），和撕裂强力（Y3）有强负相关（-0.7, $P = 0.003$）。透湿率（Y5）与其他六个性能指标之间的相关性均较低，表明它在这些性能指标中相对独立。

接着，采用灰色关联分析进一步探究各性能指标的内在联系。考虑到影响的相互性，表 3 只列出六个性能指标的分析结果。可以看出，断裂强力（Y1）、断裂伸长率（Y2）、撕裂强力（Y3）这三个性能指标存在显著的关联性；柔软度（Y6）和急弹角度（Y7）间的关联性同样很强；透气率（Y4）和撕裂强力（Y3）关联紧密；透湿率（Y5）和柔软度（Y6）的关联最为突出。

表 3：性能指标间灰色关联分析结果

Y1	**关联度**	**排名**	**Y2**	**关联度**	**排名**	**Y3**	**关联度**	**排名**
Y2	0.860	1	Y1	0.861	1	Y1	0.856	1
Y3	0.844	2	Y3	0.804	2	Y2	0.817	2
Y5	0.685	3	Y5	0.704	3	Y4	0.715	3
Y4	0.649	4	Y6	0.618	4	Y5	0.702	4
Y6	0.602	5	Y4	0.613	5	Y6	0.631	5
Y7	0.589	6	Y7	0.605	6	Y7	0.621	6
Y4	**关联度**	**排名**	**Y5**	**关联度**	**排名**	**Y6**	**关联度**	**排名**
Y3	0.723	1	Y6	0.764	1	Y7	0.818	1
Y1	0.674	2	Y7	0.740	2	Y5	0.730	2
Y2	0.634	3	Y2	0.735	3	Y2	0.612	3
Y5	0.620	4	Y1	0.719	4	Y3	0.607	4
Y7	0.594	5	Y3	0.716	5	Y1	0.598	5
Y6	0.583	6	Y4	0.634	6	Y4	0.555	6

5.3.2 结果与讨论

结合表 3 和图 5，关联性分析结果揭示了一种显著趋势：同属于一个大类的性能

指标间存在最强的相互影响。例如，在力学性能上，断裂强力（Y1）、断裂伸长率（Y2）和撕裂强力（Y3）表现出显著的内在联系；同样，在柔软性能上，柔软度（Y6）和急弹角度（Y7）也显示出紧密的相关性。这种关联性是符合逻辑的，因为这些指标通常受到相似的物理和化学因素的影响。然而，在热湿舒适性能上，透气率（Y4）和透湿率（Y5）之间的关联性却异常低。这一发现与预期相悖，这种低关联性可能是由于样本量较小，导致统计分析的可信度降低；或是由于其不同的物理特性所致。因此，为了更准确地理解这些指标之间的相互作用，可能需要进一步增加实验样本，并深入探究指标背后的物理特性。

5.4 工艺参数对产品性能交互作用分析

分析不同工艺参数之间以及它们与产品性能之间的交互作用，有助于理解工艺参数如何共同影响产品性能，以及在优化产品性能时应如何调整这些参数。本文采用逐步回归和三因素方差分析相结合的方法，分析工艺参数对性能的交互作用。

5.4.1 逐步回归

逐步回归（Stepwise Regression, SR）是一种用于选择回归模型中变量的统计方法，它通过逐步添加或删除预测变量，找到最佳的回归模型[13]。该方法可以减少多重共线性的问题，使模型更易于解释和理解。为了找到工艺参数的交互作用，本文构建树脂含量（X1）×固化温度（X2），树脂含量（X1）×碱减量程度（X3），固化温度（X2）×碱减量程度（X3），树脂含量（X1）×固化温度（X2）×碱减量程度（X3）四个交叉项，建立逐步回归模型如下：

$$\begin{aligned} Y = {} & \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 (X_1 X_2) \\ & + \beta_5 (X_1 X_3) + \beta_6 (X_2 X_3) + \beta_7 (X_1 X_2 X_3) + \varepsilon \end{aligned} \tag{6}$$

其中，β为各项系数；ε为误差项。采用向前选择和向后剔除结合的方法，分别对七个性能指标进行逐步回归，结果如表 4 所示。其中，断裂伸长率（Y2）没有合适的模型；撕裂强力（Y3）、透气率（Y4）和柔软度（Y6）不受交互项影响。

表 4：逐步回归分析结果

Y1	标准化系数	P	Y5	标准化系数	P	Y7	标准化系数	P
X3	-0.718	0.001	X1	-0.847	0.000	X2	0.202	0.011
X1X2X3	-0.402	0.024	X1X3	0.455	0.000	X3	0.254	0.000
—	—	—	X1X2X3	-0.424	0.002	X1X2X3	0.917	0.003

可以看出，交互项在性能指标上确实发挥了显著的作用。尤其对于急弹角度（Y7），标准化系数高达 0.917，表明树脂含量（X1）、固化温度（X2）以及碱减量程度（X3）三组的交互作用对急弹角度产生了显著的正面影响。此外，该组合对断

裂强力（Y1）和透湿率（Y5）产生了中等程度的负面影响。树脂含量（X1）与碱减量程度（X3）两组的交互作用对透湿率（Y5）带来了中等程度的正面影响。

5.4.2 三因素方差分析

三因素方差分析（Three-way ANOVA）是一种多因素方差分析，用于分析三个因素的不同水平是否对结果有显著影响，以及三因素之间的交互效应[14]。本文建立三因素 ANOVA 模型如下：

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + (\alpha\beta)_{ij} + (\alpha\gamma)_{ik} + (\beta\gamma)_{jk} + (\alpha\beta\gamma)_{ijk} + \varepsilon_{ijk} \quad (7)$$

其中，μ 为总体均值；α_i, β_j, γ_k 为三因素主效应；$(\alpha\beta)_{ij}$, $(\alpha\gamma)_{ik}$, $(\beta\gamma)_{jk}$ 为两者交互效应；$(\alpha\beta\gamma)_{ijk}$ 为三者交互效应；ε_{ijk} 为误差项。

方差分析结果如表 5 所示（仅以断裂强力 Y1 为例，其余见附录 A.1）。重点关注交互项的影响，发现树脂含量（X1）、固化温度（X2）以及碱减量程度（X3）三组的交互影响十分显著（F = 245.339, P = 0.000）；其次是固化温度和碱减量程度的交互作用（F = 188.271, P = 0.000）；而树脂含量和另外两者的组合交互作用较低。对于其他六个性能指标的分析过程与此类似，因此不再赘述。

表 5：三因素方差分析结果（断裂强力 Y1）

项	均方	F	P	R^2	调整 R^2
X1	0.237	216.299	0.000*		
X2	0.146	133.463	0.000*		
X3	0.185	168.342	0.000*		
X1X2	0.064	58.247	0.000*	0.881	0.835
X1X3	0.065	59.182	0.000*		
X2X3	0.097	188.271	0.000*		
X1X2X3	2.686	245.339	0.000*		

六、问题二：产品性能独立优化模型

6.1 问题二分析

问题二要求在问题一的基础上，分别建立工艺参数与人造革 7 种性能之间的关系模型，并通过关系模型找到不同性能指标各自的最佳工艺参数。问题一发现，工艺参数对性能指标存在复杂的交互作用。为了更好地抓住数据间的非线性特征，采用机器学习算法，选择逻辑回归、朴素贝叶斯、决策树、随机森林、支持向量机、多层感知机、AdaBoost、XGBoost、CatBoost 和 LightGBM 等 10 种回归模型分别对不同的性能指标进行拟合。在模型训练时，按 7:3 的比例把数据随机划分为训练集和测试

集，并使用五折交叉验证和 RMSE、MAE、R^2 等指标评估模型效果，进而确定最优模型。考虑到模型的非线性和搜索空间的复杂性，使用粒子群算法进行求解，找出最优性能下各自的最佳工艺参数。问题二思路流程如图 6 所示。

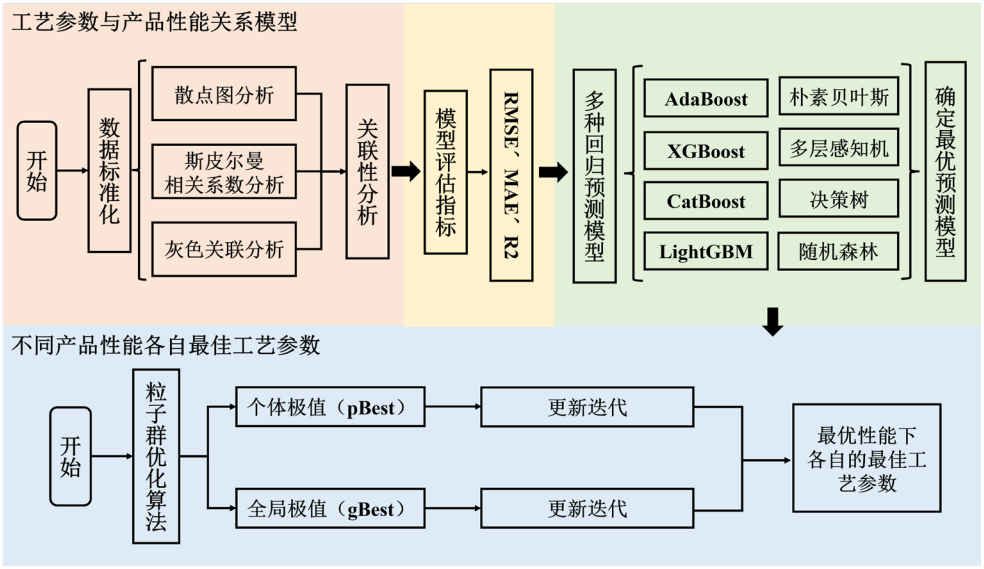

图 6：问题二思路流程

6.2 工艺参数与产品性能关系模型

6.2.1 机器学习回归模型构建

本节选择十种常见的机器学习回归模型，分别为：线性回归、朴素贝叶斯、多层感知机、支持向量机、决策树、随机森林、AdaBoost、XGBoost、CatBoost 和 LightGBM。在模型训练时，按 7:3 比例把数据随机划分为训练集和测试集，并使用五折交叉验证和 RMSE、MAE、R^2 等指标评估模型效果，确定最优的回归模型。

6.2.2 评价指标与模型评估

为了衡量不同模型在测试集上的表现，本文使用以下指标对模型进行评估：

（1）均方误差（MSE）：衡量预测值与实际值之间差的平方的平均值。

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8}$$

（2）均方根误差（RMSE）：均方误差的平方根。

$$RMSE = \sqrt{MSE} \tag{9}$$

（3）平均绝对误差（MAE）：衡量预测值与实际值之间差的绝对值的平均值。

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{10}$$

（4）平均绝对百分比误差（MAPE）：通常用于衡量预测的准确性。

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{11}$$

（5）R^2：变异性与总变异性的比例，值越接近 1 模型解释力越强。

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{12}$$

其中，y_i, $\hat{y}_i$, $\bar{y}_i$ 分别为样本观测值、预测值和样本平均值。

考虑到篇幅限制，本小节以断裂强力（Y1）为例，评估其在十个机器学习回归模型上的性能，结果如表 6 所示。其余产品性能回归模型评估结果见附录 A.2。可以看出，XGBoost 与其他模型相比具有较高的性能和泛化能力。因此，使用该模型作为断裂强力性能指标的回归模型。

表 6：十种机器学习回归模型在验证集上的表现（断裂强力 Y1）

模型名称	MSE	RMSE	MAE	MAPE	R^2
线性回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	0.061	0.246	0.190	24.228	0.944
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	**0.018**	**0.134**	**0.122**	**15.597**	**0.985**
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

6.3 不同产品性能最佳工艺参数

6.3.1 粒子群优化算法

考虑到模型的非线性和搜索空间的复杂性，使用粒子群优化算法进行求解，找出不同最优性能各自的最佳工艺参数。粒子群优化算法（Particle Swarm Optimization, PSO）是一种基于群体的智能优化算法，广泛应用于求解非线性优化问题[15-17]。

在 PSO 中，每个解被视为搜索空间中的一个粒子，每个粒子在搜索过程中会记录自己所找到的最优解，称为个体极值（*pBest*）。同时，整个粒子群会记录所有粒子找到的最优解，称为全局极值（*gBest*）。比较各粒子当前适应度值与个体极值，若当

前适应度值更优，则更新个体极值。然后比较所有粒子的个体极值，找到最优的个体极值并更新全局极值。速度和位置更新公式分别为：

$$v_i(t+1)=w\cdot v_i(t)+c_1\cdot r_1\cdot(pBest_i-x_i(t))+c_2\cdot r_2\cdot(gBest_i-x_i(t)) \tag{13}$$

$$x_i(t+1)=x_i(t)+v_i(t+1) \tag{14}$$

其中，$v_i(t)$, $x_i(t)$分别是粒子i在t时刻的速度和位置；w为惯性权重；c_1, c_2为学习因子；r_1, r_2是介于 0 和 1 的随机数。

6.3.2 工艺参数优化求解

在本节中，待优化的目标函数为 6.2 节筛选出的 7 个机器学习回归模型，三个工艺参数的约束条件分别设为实验中各参数四个水平的上下界（取闭区间）。通过粒子群优化算法得到七个性能指标的最优值和各自相应的最佳参数组合如下。

表 7：粒子群优化算法结果

性能指标	性能指标最优值	最佳树脂含量	最佳固化温度	最佳碱减量程度
断裂强力	1773.42	23.98	104.56	0.01
断裂伸长率	1.13	24.83	107.37	0.03
撕裂强力	185.94	24.08	115.76	0.14
透气率	336.17	18.53	100.41	0.12
透湿率	3577.39	15.01	125.19	0.15
柔软度	3.58	16.39	129.13	0.30
折皱回复角	175.44	20.65	128.94	0.29

七、问题三：产品性能组合优化模型

7.1 问题三分析

问题三要求分别分析实现最优力学性能、最优热湿舒适性和最优柔软性能所需的工艺参数，同时分析使人造革 7 项指标综合性能最优所需的工艺参数，并将两个结果进行比较。该问是在问题二的基础上，对多个性能进行多目标优化。考虑到直接求解多目标优化模型较为复杂，本问采用熵权法，对标准化后的各性能指标进行客观赋权，将多个目标函数转化为单一的目标函数进行求解。对力学性能（断裂强力 Y1、断裂伸长率 Y2、撕裂强力 Y3）、热湿舒适性能（透气率 Y4、透湿率 Y5）、柔软性能（柔软度 Y6、折皱回复角 Y7）以及综合性能（7 个指标）进行赋权，得到性能指标 Y8、Y9、Y10 和 Y11。最后，重复问题二中的步骤，分别找出最优的回归模型并利用粒子群优化算法进行求解。问题三思路流程如图 7 所示。

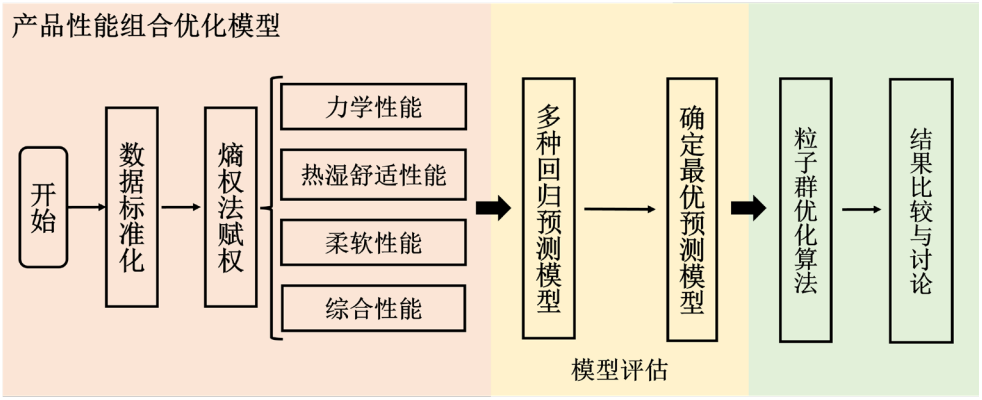

图 7：问题三思路流程

7.2 熵权法赋权

熵权法（Entropy Weight Method, EWM）是一种客观赋权方法，通过计算各指标的熵值来确定权重，反映指标的离散程度[18,19]。熵值越大表示该指标的不确定性越高，对综合评价的贡献越小；反之，熵值越小对综合评价的贡献越大。熵权法的优点是不需要主观赋权，可以客观反映各指标的重要性，其基本步骤如下：

首先，根据标准化后的数据计算性能指标的熵值：

$$e_j = -\frac{1}{\ln m}\sum_{i=1}^{m} r_{ij}\cdot \ln r_{ij} \tag{15}$$

其中，e_j 是第 j 个性能指标的熵值；m 为实验组数。接着计算指标的差异系数，差异系数越大，指标的有效信息量越高：

$$g_j = 1 - e_j \tag{16}$$

最后，确定各性能指标的权重如下。其中，n 为指标个数。

$$w_j = \frac{g_j}{\sum_{j=1}^{n} g_j} \tag{17}$$

在本节中，对力学性能（断裂强力 Y1、断裂伸长率 Y2、撕裂强力 Y3）、热湿舒适性能（透气率 Y4、透湿率 Y5）、柔软性能（柔软度 Y6、折皱回复角 Y7）以及综合性能（7 个指标）分别进行熵权法赋权，得到性能指标 Y8、Y9、Y10 和 Y11。

7.3 最佳力学性能优化模型

使用 7.2 节中的方法，对力学性能指标（断裂强力 Y1、断裂伸长率 Y2、撕裂强力 Y3）进行赋权，得到指标 Y8 如下：

$$Y_8 = 0.52Y_1 + 0.13Y_2 + 0.35Y_3 \tag{18}$$

重复问题二中的寻优步骤，得到最优机器学习模型、最佳力学性能和相应的工艺参数组合如表 8 所示。

表 8：力学性能（Y8）优化结果

最优模型	最佳力学性能（Y1, Y2, Y3）	树脂含量	固化温度	碱减量程度
XGBoost	(1742.01, 1.12, 173.22)	15.76	100.73	0.01

7.4 最佳热湿舒适性能优化模型

对热湿舒性能指标（透气率 Y4、透湿率 Y5）进行赋权，得到指标 Y9 如下：

$$Y_9 = 0.52Y_4 + 0.48Y_5 \tag{19}$$

重复问题二中的寻优步骤，得到最优机器学习模型、最佳热湿舒性能和相应的工艺参数组合如表 9 所示。

表 9：热湿舒性能（Y9）优化结果

最优模型	最佳热湿舒性能（Y4, Y5）	树脂含量	固化温度	碱减量程度
XGBoost	(319.43, 3477.85)	19.18	101.51	0.13

7.5 最佳柔软性能优化模型

对柔软性能指标（柔软度 Y6、折皱回复角 Y7）进行赋权，得到指标 Y10 如下：

$$Y_{10} = 0.58Y_6 + 0.42Y_7 \tag{20}$$

重复问题二中的寻优步骤，得到最优机器学习模型、最佳柔软性能和相应的工艺参数组合如表 10 所示。

表 10：柔软性能（Y10）优化结果

最优模型	最佳柔软性能（Y6, Y7）	树脂含量	固化温度	碱减量程度
随机森林	(3.29, 169.92)	16.22	128.34	0.29

7.6 最佳综合性能优化模型

对综合性能指标（断裂强力 Y1、断裂伸长率 Y2、撕裂强力 Y3、透气率 Y4、透湿率 Y5、柔软度 Y6、折皱回复角 Y7）进行赋权，得到指标 Y11 如下：

$$Y_{11} = 0.28Y_1 + 0.07Y_2 + 0.16Y_3 + 0.15Y_4 + 0.13Y_5 + 0.12Y_6 + 0.09Y_7 \tag{21}$$

重复问题二中的寻优步骤，得到最优机器学习模型和相应的工艺参数组合如表 11 所示。综合性能七项指标分别为：1742.36、1.17、159.21、326.72、3419.08、3.44 和 171.25。

表 11：综合性能（Y11）优化结果

最优模型	树脂含量	固化温度	碱减量程度
LightGBM	26.74	119.62	0.02

7.7 结果比较与讨论

从回归模型的选择角度审视，XGBoost 模型在力学性能和热湿舒适性能的优化中表现最为出色，随机森林模型则在柔软性能的优化中占据优势，综合性能的优化则以 LightGBM 模型为最佳选择。在工艺参数的最优值方面，树脂含量的波动范围从 15.76 wt%至 26.74 wt%，突显了不同性能优化对树脂含量需求的显著差异。固化温度的变化从 100.73 ℃至 128.34 ℃，进一步揭示了不同性能对温度敏感度的多样性。值得注意的是，碱减量程度的变化幅度最大（0.01 至 0.29），这可能对产品性能产生显著影响。整体而言，专注于单一性能的优化模型在特定性能上表现卓越，但同时也在其他性能上做出了妥协。工艺参数的精心选择对最终产品的性能至关重要，必须基于实际需求进行细致权衡。

八、问题四：热湿舒适性优先的产品性能优化模型

8.1 问题四分析

问题四要求基于“优先满足热湿舒适性，其次考虑力学性能和柔软性能，同时要求各项性能指标尽量接近问题 2 中的最佳水平”的要求，为某沙漠科考队设计最佳性能的人造革。该问同样是一个多目标优化问题，考虑到性能指标的优先级，采用主客观赋值相结合的方法。对于三大性能指标，利用层次分析法进行主观赋权；对于各子指标，依旧采用熵权法客观赋权。最终构造新出指标Y12，将多目标转化为单目标优化模型。同样重复问题二的步骤，找出最优回归模型并利用粒子群优化算法进行求解。问题四思路流程如图 8 所示。

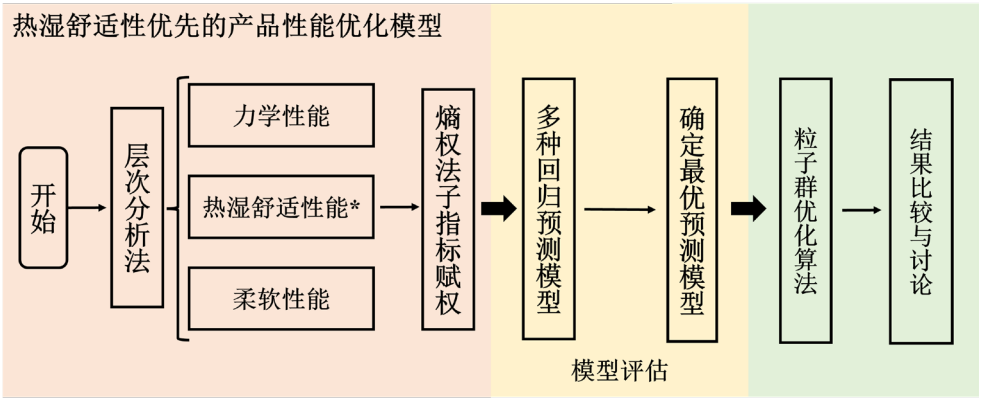

图 8：问题四思路流程

8.2 沙漠科考队人造革性能优化模型

8.2.1 层次分析-熵权法赋权

层次分析法（Analytic Hierarchy Process, AHP）是一种常用的决策分析方法，它通过建立层次结构模型，将复杂问题分解为多个组成因素，并在各因素之间进行两两比较、打分，以确定各因素的相对重要性，进而进行综合评价和决策[20-22]。对于沙漠科考队提出的人造革需求，可应用层次分析法对三大性能指标赋权，而对于各子性能指标，仍采用熵权法进行客观赋权。

8.2.2 结果与讨论

在本节中，层次分析法的目标层为寻找最优人造革；准则层为三大指标，即力学性能、热湿舒适性能和柔软性能；指标层为七个小指标，即断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度和折皱回复角。考虑到热湿舒适性能对沙漠科考队人造革的优先级，构造准则层的判断矩阵如下。其中，1 表示同等重要；2 表示稍微重要；5 表示明显重要。

表 12：层次分析准侧层判断矩阵

	力学性能	**热湿舒适性能**	柔软性能
力学性能	1	0.2	0.5
热湿舒适性能	5	1	5
柔软性能	2	0.5	1

使用“方根法”计算权重，得到力学性能的权重为 11.25%，热湿舒适性能的权重为 70.89%，柔软性能权重为 17.85%，且结果通过一致性检验（CI = 0.027, CR = 0.051 < 0.1）。对于指标层，利用熵权法对各子指标进行赋权，得到断裂强力、断裂伸长率、撕裂强力、透气率、透湿率、柔软度和折皱回复角的权重分别为：0.278、0.068、0.185、0.135、0.125、0.121 和 0.088。将其与层次分析法得到的准则层权重结合，可算出各指标的最终权重，结果见图 9。

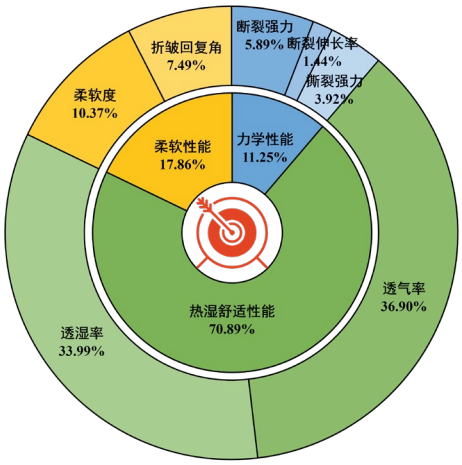

图 9：层次分析-熵权法指标赋权结果

因此，构建新指标 Y12 来表示这 7 个指标的组合：

$$Y_{12} = 0.0589Y_1 + 0.0144Y_2 + 0.0392Y_3 + 0.369Y_4 + 0.3399Y_5 + 0.1037Y_6 + 0.0749Y_7 \tag{22}$$

重复问题二中的寻优步骤，得到最优回归模型和相应工艺参数如表 13 所示。此时热湿舒适性能指标为：透气率 319.85 L/m^2/s、透湿率 3489.73 g/(m^2·24h)。力学性能和柔软性能分别为：断裂强力 1574.25 N、断裂伸长率 0.93、撕裂强力 150.91 N；柔软度 2.89 mm、折皱回复角 167.33 度。比较发现，该模型较好优化了沙漠科考队人造革的热湿舒适性能，同时其余各指标也较为接近问题二的最佳水平。

表 13：沙漠科考队人造革性能优化结果

最优模型	树脂含量	固化温度	碱减量程度
XGBoost	21.46	113.15	0.04

九、灵敏度分析

考虑到问题四中应用层次分析法对三大性能指标进行权重分配的主观性，有必要进行敏感性分析，以评估这种主观性对最优回归模型预测精度的潜在影响。在问题四中，通过层次分析法确定的热湿舒适性能权重为 70.89%，鉴于这一性能指标对沙漠科考队使用的人造革材料具有较高的优先级，本节将重点对该权重进行敏感性分析。令热湿舒适性能的权重在原始值上下 10%的范围内变动，并评估这种变动对最优回归模型预测性能的影响。由于计算资源的限制，本研究将对自变量进行离散化处理，仅探讨权重变动±5%和±10%时对模型预测效果的影响。

图 10 为层次分析法指标赋权的灵敏度分析结果。可以看出，即使热湿舒适性能的权重在±10%的范围发生波动，大多数性能的预测结果都保持稳定，仅在透湿率的预测上出现了较为显著的偏差。这证明了层次分析法的合理性，表明由主观性引起的预测偏差是相对较小的。因此，在为沙漠科考队选择合适的人造革时，即使对热湿舒适性能的重要性有所调整，也不会对最终的优化结果产生显著影响。

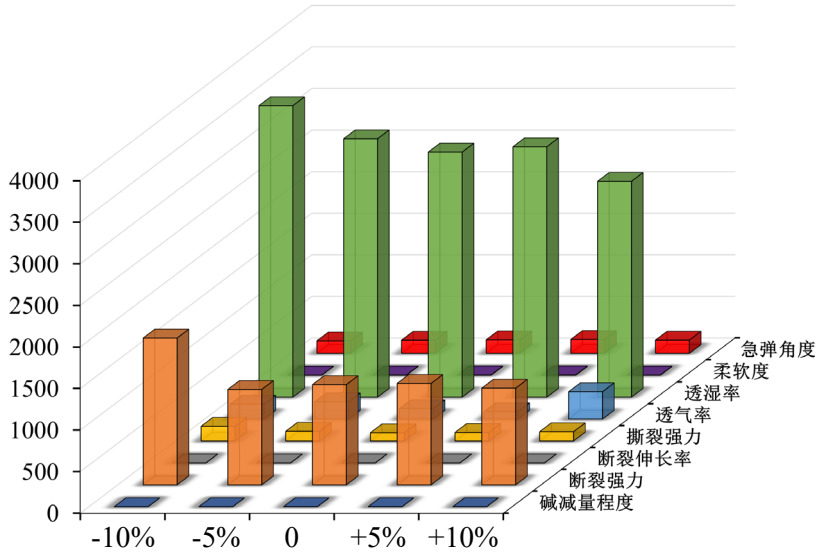

图 10：层次分析法指标赋权灵敏度分析

十、模型评价

10.1 模型优点

- 模型综合考虑了力学性能、热湿舒适性能、柔软性能等多个维度，能够全面评估人造革的性能。通过机器学习和统计方法，模型能够基于实验数据做出更加精确的预测和优化；
- 使用了包括 XGBoost、随机森林、LightGBM 在内的多种算法，提高模型的泛化能力和预测准确性。结合客观和主观的赋权方法，使得处理多目标优化中的权重分配问题更为合理；
- 粒子群优化算法的引入有效解决了多参数优化问题，有利于寻找最佳工艺参数组合。考虑模型评估和灵敏度分析，增强结果的可信度和鲁棒性。

10.2 模型缺点

- 模型涉及多种统计算法和优化技术，计算过程可能相对复杂。在机器学习模型训练过程中存在过拟合风险，特别是在数据量有限的情况下；
- 模型在理论上能够得到优化，但实际生产过程中受到设备、材料批次等因素影响。粒子群优化算法存在一些局限性，如可能陷入局部最优解、参数选择对算法性能影响较大等。

十一、参考文献

[1] Zeng, J., Schlup, J. R., & Fan, L. T. (2000). Synthesis and mechanical properties of leather–epoxy interpenetrating polymer networks. *Journal of Applied Polymer Science*, *78*(6), 1224–1232.

[2] Song, Y., Zeng, Y., Xiao, K., Wu, H., & Shi, B. (2017). Effect of molecular weight of acrylic resin retanning agent on properties of leather. *Journal of the American Leather Chemists Association*, *112*(04), 128–134.

[3] Zamani, M., & Shokrieh, M. M. (2023). Investigation of curing time and temperature effects on mechanical properties and thermal degradation of artificial leather made of PVC. *Journal of Science and Technology of Composites*.

[4] Hu, Y., Liu, J., Han, G., Li, X., Zhang, Z., Zheng, X., ... & Tang, K. (2022). Artificial deterioration of vegetable-tanned leather under synergistic effect of temperature and humidity. *Journal of Cultural Heritage*, *53*, 118–126.

[5] He, Z., Huang, Z., Chen, J., Chen, A., Ai, J., Song, L., & Liu, B. (2023). Preparation of modified CO2-based polyurethane for wet-type artificial leather with excellent alkali resistance. *Journal of Applied Polymer Science*, *140*(46), e54671.

[6] Feng, L., Wang, W., Song, B., Zhu, X., Wang, L., Shao, R., ... & Xu, Z. (2023). Synthesis of P, N and Si-containing waterborne polyurethane with excellent flame retardant, alkali resistance and flexibility via one-step synthetic approach. *Progress in Organic Coatings*, *174*, 107286.

[7] 吴启超, 范新晖, 陈慧雪. 高耐磨 PVC 人造革的制备、改进及应用研究进展 [J]. 塑料包装, 2024, 34(2): 18–21+54.

[8] 马玲, 贾满兰, 王艳庚. 人造革产品耐折牢度和耐碱液水解牢度的影响因素分析 [J]. 纺织检测与标准, 2022, 8(3): 31–33.

[9] Data, M. C., Komorowski, M., Marshall, D. C., Salciccioli, J. D., & Crutain, Y. (2016). Exploratory data analysis. *Secondary Analysis of Electronic Health Records*, 185–203.

[10] Wu, Y., & Li, L. (2016). Sample normalization methods in quantitative metabolomics. *Journal of Chromatography A*, *1430*, 80–95.

[11] 徐维超. 相关系数研究综述[J]. 广东工业大学学报, 2012, 29(3): 12–17.

[12] Patel, M. T. (2015). Multi optimization of process parameters by using grey relation analysis-a review. *International Journal of Advanced Research in IT and Engineering*,

4(6), 1–15.

[13] 游士兵, 严研. 逐步回归分析法及其应用[J]. 统计与决策, 2017, (14): 31–35.

[14] 郭萍. 三因素方差分析的原理及应用[J]. 沈阳大学学报(自然科学版), 2015, 27(1): 40–43.

[15] Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing*, *22*(2), 387–408.

[16] 冯茜, 李擎, 全威, 等. 多目标粒子群优化算法研究综述[J]. 工程科学学报, 2021, 43 (6): 745–753.

[17] 胡旺, 李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报, 2007, (4): 861–868.

[18] Zhu, Y., Tian, D., & Yan, F. (2020). Effectiveness of entropy weight method in decision-making. *Mathematical Problems in Engineering*, *2020*(1), 3564835.

[19] 程启月. 评测指标权重确定的结构熵权法[J]. 系统工程理论与实践, 2010, 30(7): 1225–1228.

[20] 冯长根, 李杰, 李生才. 层次分析法在中国安全科学研究中的应用[J]. 安全与环境学报, 2018, 18(6): 2126–2130.

[21] Palcic, I., & Lalic, B. (2009). Analytical Hierarchy Process as a tool for selecting and evaluating projects. *International Journal of Simulation Modelling*, *8*(1).

[22] Vaidya, O. S., & Kumar, S. (2006). Analytic hierarchy process: An overview of applications. *European Journal of Operational Research*, *169*(1), 1–29.

附录 A 论文扩展结果

A.1 问题一扩展结果

表 A.1-1：三因素方差分析结果（断裂伸长率 Y2）

项	均方	F	P	R^2	调整 R^2
X1	0.153	164.866	0.000*		
X2	0.180	135.486	0.000*		
X3	0.187	145.971	0.000*		
X1X2	0.077	124.068	0.000*	0.971	0.957
X1X3	0.057	91.376	0.000*		
X2X3	0.099	102.812	0.000*		
X1X2X3	2.758	235.449	0.000*		

表 A.1-2：三因素方差分析结果（撕裂强力 Y3）

项	均方	F	P	R^2	调整 R^2
X1	0.164	81.388	0.000*		
X2	0.191	62.998	0.000*		
X3	0.115	110.447	0.000*		
X1X2	0.089	109.615	0.000*	0.817	0.792
X1X3	0.066	155.758	0.000*		
X2X3	0.037	262.884	0.000*		
X1X2X3	5.788	467.53	0.000*		

表 A.1-3：三因素方差分析结果（透气率 Y4）

项	均方	F	P	R^2	调整 R^2
X1	0.274	80.013	0.000*		
X2	0.301	87.547	0.000*		
X3	0.159	192.749	0.000*		
X1X2	0.112	238.823	0.000*	0.832	0.807
X1X3	0.078	50.379	0.000*		
X2X3	0.096	37.932	0.000*		
X1X2X3	6.158	367.53	0.000*		

表 A.1-4：三因素方差分析结果（透湿率 Y5）

项	均方	F	P	R^2	调整 R^2
X1	0.104	18.024	0.000*		
X2	0.141	33.258	0.000*		
X3	0.129	23.67	0.000*		
X1X2	0.112	16.155	0.000*	0.915	0.903
X1X3	0.099	19.811	0.000*		
X2X3	0.078	38.122	0.000*		
X1X2X3	5.974	437.048	0.000*		

表 A.1-5：三因素方差分析结果（柔软度 Y6）

项	均方	F	P	R^2	调整 R^2
X1	2.904	52.265	0.000*		
X2	0.324	59.051	0.000*		
X3	6.195	112.937	0.000*		
X1X2	2.444	145.367	0.000*	0.952	0.919
X1X3	0.850	154.938	0.000*		
X2X3	1.485	270.636	0.000*		
X1X2X3	14.857	270.337	0.000*		

表 A.1-6：三因素方差分析结果（急弹角度 Y7）

项	均方	F	P	R^2	调整 R^2
X1	0.465	46.028	0.000*		
X2	0.154	39.527	0.000*		
X3	0.092	35.251	0.000*		
X1X2	1.526	126.693	0.000*	0.845	0.837
X1X3	1.732	136.948	0.000*		
X2X3	2.411	160.844	0.000*		
X1X2X3	7.245	336.721	0.000*		

A.2 问题二扩展结果

表 A.2-1：十种机器学习回归模型在验证集上的表现（断裂伸长率 Y2）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	0.061	0.246	0.190	24.228	0.944
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	**0.018**	**0.134**	**0.122**	**15.597**	**0.985**
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

表 A.2-2：十种机器学习回归模型在验证集上的表现（撕裂强力 Y3）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	**0.061**	**0.246**	**0.190**	**24.228**	**0.944**
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	0.018	0.134	0.122	15.597	0.985
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

表 A.2-3：十种机器学习回归模型在验证集上的表现（透气率 Y4）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946

随机森林	**0.061**	**0.246**	**0.190**	**24.228**	**0.944**
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	0.018	0.134	0.122	15.597	0.985
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

表 A.2-4：十种机器学习回归模型在验证集上的表现（透湿率 Y5）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	0.061	0.246	0.190	24.228	0.944
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	**0.018**	**0.134**	**0.122**	**15.597**	**0.985**
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

表 A.2-5：十种机器学习回归模型在验证集上的表现（柔软度 Y6）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	0.061	0.246	0.190	24.228	0.944
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	**0.018**	**0.134**	**0.122**	**15.597**	**0.985**
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

表 A.2-6：十种机器学习回归模型在验证集上的表现（急弹角度 Y7）

模型名称	MSE	RMSE	MAE	MAPE	R^2
逻辑回归	0.301	0.549	0.788	41.245	0.559
朴素贝叶斯	0.063	0.250	0.298	31.671	0.893
支持向量机	0.043	0.207	0.147	27.588	0.911
多层感知机	0.075	0.274	0.245	28.533	0.908
决策树	0.066	0.256	0.230	24.392	0.946
随机森林	**0.055**	**0.194**	**0.105**	**14.817**	**0.994**
AdaBoost	0.069	0.264	0.230	29.507	0.895
XGBoost	0.018	0.134	0.122	15.597	0.985
CatBoost	0.140	0.374	0.267	34.479	0.881
LightGBM	0.292	0.445	0.346	45.568	0.643

附录 B 论文源代码

B.1 问题一源代码

Prob1.py

```
import pandas as pd
import matplotlib.pyplot as plt

# Descriptive statistics
stats_table = merged_data.describe()

# Boxplot for process parameters and performance indicators
process_params = ['ResinContent', 'CuringTemperature', 'AlkaliReduction']
performance_indicators = ['TensileStrength', 'Elongation', 'TearStrength',
'Permeability', 'MoistureTransmission', 'Softness', 'Flexibility']

plt.figure(figsize=(12, 8))
merged_data[process_params].boxplot()
plt.xticks(rotation=45)
plt.title('Boxplot of Process Parameters')
plt.savefig('ProcessParameters_Boxplot.png')
plt.show()

plt.figure(figsize=(12, 8))
merged_data[performance_indicators].boxplot()
plt.xticks(rotation=45)
plt.title('Boxplot of Performance Indicators')
plt.savefig('PerformanceIndicators_Boxplot.png')
plt.show()


from sklearn.preprocessing import StandardScaler
```

```
# Select columns to standardize
columns_to_standardize = performance_indicators

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the data
normalized_data = merged_data.copy()
normalized_data[columns_to_standardize] =
scaler.fit_transform(merged_data[columns_to_standardize])

# Save normalized data to CSV
normalized_data.to_csv("NormalizedData.csv", index=False)


import seaborn as sns

# Plot scatter plots for process parameters vs performance indicators
fig, axes = plt.subplots(2, 3, figsize=(18, 12))

sns.scatterplot(ax=axes[0, 0], x='ResinContent', y='MoistureTransmission',
data=merged_data)
axes[0, 0].set_title('Resin Content vs Moisture Transmission')

sns.scatterplot(ax=axes[0, 1], x='CuringTemperature', y='TensileStrength',
data=merged_data)
axes[0, 1].set_title('Curing Temperature vs Tensile Strength')

sns.scatterplot(ax=axes[0, 2], x='AlkaliReduction', y='TensileStrength',
data=merged_data)
axes[0, 2].set_title('Alkali Reduction vs Tensile Strength')

sns.scatterplot(ax=axes[1, 0], x='ResinContent', y='Permeability',
data=merged_data)
axes[1, 0].set_title('Resin Content vs Permeability')

sns.scatterplot(ax=axes[1, 1], x='CuringTemperature', y='Permeability',
data=merged_data)
axes[1, 1].set_title('Curing Temperature vs Permeability')

sns.scatterplot(ax=axes[1, 2], x='AlkaliReduction', y='TearStrength',
data=merged_data)
axes[1, 2].set_title('Alkali Reduction vs Tear Strength')

plt.tight_layout()
plt.savefig('ScatterPlots_ProcessParameters_vs_PerformanceIndicators.png')
plt.show()


import numpy as np
import seaborn as sns

# Calculate Spearman correlation coefficients
correlation_matrix = merged_data.corr(method='spearman')

# Plot heatmap of Spearman correlation coefficients
```

```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1,
vmax=1)
plt.title('Spearman Correlation Coefficient Heatmap')
plt.savefig('SpearmanCorrelationHeatmap.png')
plt.show()


from sklearn.preprocessing import MinMaxScaler

# Normalize the data using Min-Max scaling
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(merged_data[performance_indicators
+ process_params])

# Reference sequence: mean of each performance indicator
reference_sequence = np.mean(normalized_data, axis=0)

def grey_relation_coefficient(reference, comparison,
resolution_coefficient=0.5):
    diff = np.abs(reference - comparison)
    min_diff = np.min(diff)
    max_diff = np.max(diff)
    return (min_diff + resolution_coefficient * max_diff) / (diff +
resolution_coefficient * max_diff)

# Calculate Grey Relational Grades
grey_relation_grades = np.zeros((normalized_data.shape[0],
len(performance_indicators)))

for i, indicator in enumerate(performance_indicators):
    for j in range(normalized_data.shape[0]):
        grey_relation_grades[j, i] =
grey_relation_coefficient(reference_sequence[i], normalized_data[j, i])

# Calculate mean Grey Relational Grade for each performance indicator
mean_grades = np.mean(grey_relation_grades, axis=0)

# Create a DataFrame to display results
gra_results = pd.DataFrame({
    'Performance Indicator': performance_indicators,
    'Grey Relational Grade': mean_grades
})

# Sort the results by Grey Relational Grade
gra_results = gra_results.sort_values(by='Grey Relational Grade',
ascending=False)

# Save the results to a CSV file
gra_results.to_csv("GreyRelationalAnalysisResults.csv", index=False)

# Calculate Spearman correlation coefficients for performance indicators
performance_correlation_matrix =
merged_data[performance_indicators].corr(method='spearman')
```

```
# Plot heatmap of Spearman correlation coefficients for performance
indicators
plt.figure(figsize=(10, 6))
sns.heatmap(performance_correlation_matrix, annot=True, cmap='coolwarm',
vmin=-1, vmax=1)
plt.title('Spearman Correlation Coefficient Heatmap (Performance
Indicators)')
plt.savefig('SpearmanCorrelationHeatmap_PerformanceIndicators.png')
plt.show()

# Normalize the performance indicator data using Min-Max scaling
normalized_performance_data =
scaler.fit_transform(merged_data[performance_indicators])

# Reference sequence: mean of each performance indicator
reference_sequence_performance = np.mean(normalized_performance_data,
axis=0)

# Calculate Grey Relational Grades for performance indicators
grey_relation_grades_performance =
np.zeros((normalized_performance_data.shape[0],
len(performance_indicators)))

for i, indicator in enumerate(performance_indicators):
    for j in range(normalized_performance_data.shape[0]):
        grey_relation_grades_performance[j, i] =
grey_relation_coefficient(reference_sequence_performance[i],
normalized_performance_data[j, i])

# Calculate mean Grey Relational Grade for each performance indicator
mean_grades_performance = np.mean(grey_relation_grades_performance,
axis=0)

# Create a DataFrame to display results
gra_performance_results = pd.DataFrame({
    'Performance Indicator': performance_indicators,
    'Grey Relational Grade': mean_grades_performance
})

# Sort the results by Grey Relational Grade
gra_performance_results = gra_performance_results.sort_values(by='Grey
Relational Grade', ascending=False)

# Save the results to a CSV file
gra_performance_results.to_csv("GreyRelationalAnalysisResults_PerformanceI
ndicators.csv", index=False)


import statsmodels.api as sm
from statsmodels.formula.api import ols

# Function to perform stepwise regression
def stepwise_regression(X, y):
    initial_list = []
    included = list(X.columns)
    while True:
```

```
        changed = False
        # Forward step
        excluded = list(set(X.columns) - set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included +
[new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < 0.05:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed = True
        # Backward step
        model = sm.OLS(y,
sm.add_constant(pd.DataFrame(X[included]))).fit()
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max()
        if worst_pval > 0.10:
            changed = True
            worst_feature = pvalues.idxmax()
            included.remove(worst_feature)
        if not changed:
            break
    return included

# Prepare the data
X = merged_data[process_params]
y = merged_data['TensileStrength']  # Change the target variable as needed

# Add interaction terms
X['X1_X2'] = X['ResinContent'] * X['CuringTemperature']
X['X1_X3'] = X['ResinContent'] * X['AlkaliReduction']
X['X2_X3'] = X['CuringTemperature'] * X['AlkaliReduction']
X['X1_X2_X3'] = X['ResinContent'] * X['CuringTemperature'] *
X['AlkaliReduction']

# Perform stepwise regression
included_features = stepwise_regression(X, y)
print("Selected features:", included_features)

# Fit the final model
final_model = sm.OLS(y, sm.add_constant(X[included_features])).fit()
print(final_model.summary())

# Define the model formula
formula = 'TensileStrength ~ C(ResinContent) * C(CuringTemperature) *
C(AlkaliReduction)'

# Fit the model
anova_model = ols(formula, data=merged_data).fit()

# Perform ANOVA
anova_results = sm.stats.anova_lm(anova_model, typ=2)
print(anova_results)
```

B.2 问题二源代码

Prob2.py

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
from pyswarm import pso  # PSO implementation library

# Load your dataset
data = pd.read_csv('data.csv')

# Separate features (process parameters) and targets (performance
indicators)
X = data[['x1', 'x2', 'x3']]  # Replace with actual column names
performance_indicators = ['y1', 'y2', 'y3', 'y4', 'y5', 'y6', 'y7']

# Split data into train, validation, and test sets (7:3 ratio)
X_train, X_valtest, y_train_all, y_valtest_all = train_test_split(X,
data[performance_indicators], test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_valtest, y_valtest_all,
test_size=0.33, random_state=42)

# Define regression models to be evaluated
models = {
    'Linear Regression': LinearRegression(),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVR(),
    'Multi-layer Perceptron': MLPRegressor(random_state=42),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'AdaBoost': AdaBoostRegressor(random_state=42),
    'XGBoost': XGBRegressor(random_state=42),
    'CatBoost': CatBoostRegressor(random_state=42, verbose=0),
    'LightGBM': LGBMRegressor(random_state=42)
}

# Define evaluation function
def evaluate_model(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)
    mse = mean_squared_error(y_val, y_pred)
```

```
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_val, y_pred)
    r2 = r2_score(y_val, y_pred)
    return mse, rmse, mae, r2

# Evaluate all models for each performance indicator
results = {}
for pi in performance_indicators:
    y_train = y_train_all[pi]
    y_val_ = y_val[pi]
    results[pi] = {}
    for name, model in models.items():
        mse, rmse, mae, r2 = evaluate_model(model, X_train, y_train,
X_val, y_val_)
        results[pi][name] = {'MSE': mse, 'RMSE': rmse, 'MAE': mae, 'R2':
r2}
        print(f"Model: {name}, Performance Indicator: {pi}, MSE: {mse},
RMSE: {rmse}, MAE: {mae}, R2: {r2}")

# Determine the best model for each performance indicator based on R2
best_models = {pi: max(results[pi], key=lambda x: results[pi][x]['R2'])
for pi in performance_indicators}

print("Best models for each performance indicator:", best_models)


# Example PSO function for optimizing parameters for a given performance
indicator using the best model
def objective_function(params, model, scaler):
    params = np.array(params).reshape(1, -1)
    params = scaler.transform(params)  # Ensure the parameters are scaled
if using scaling
    predicted_y = model.predict(params)
    return -predicted_y  # Negative because PSO minimizes, and we want to
maximize the performance indicator

# Define parameter bounds based on your process ranges
# Replace min1, max1, min2, max2, etc., with actual parameter bounds
bounds = [(min1, max1), (min2, max2), (min3, max3)]

# Run PSO optimization for each performance indicator
optimized_parameters = {}
for pi in performance_indicators:
    best_model_name = best_models[pi]
    best_model = models[best_model_name]
    best_model.fit(X_train, y_train_all[pi])  # Ensure the model is
trained with the best model
    scaler = StandardScaler().fit(X_train)  # Fit scaler on training data
if needed

    best_params, _ = pso(objective_function, lb=[bound[0] for bound in
bounds], ub=[bound[1] for bound in bounds],
                         args=(best_model, scaler))
    optimized_parameters[pi] = best_params
```

```
    print("Optimized Parameters for each performance indicator:",
optimized_parameters)
```

B.3 问题三源代码

Prob3.py

```
import numpy as np

# Calculate entropy values
def entropy(data):
    eps = np.finfo(float).eps
    data_normalized = data / np.sum(data, axis=1, keepdims=True)
    entropy_values = -np.sum(data_normalized * np.log(data_normalized +
eps), axis=1)
    return entropy_values

# Calculate coefficient of variation
def coefficient_of_variation(data):
    std_dev = np.std(data, axis=1)
    mean = np.mean(data, axis=1)
    return std_dev / mean

# Calculate weights using entropy weight method
def entropy_weight(data):
    ent = entropy(data)
    cv = coefficient_of_variation(data)
    weights = (1 - ent) * (1 - cv)
    weights /= np.sum(weights)
    return weights

# Calculate weights
weights = entropy_weight(normalized_data)
print("Weights:", weights)

from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor

# Features and target values
X = data[['resin_content', 'curing_temp', 'alkali_reduction']]
y_mechanical = data[['Y1', 'Y2', 'Y3']]
y_thermal_comfort = data[['Y4', 'Y5']]
y_softness = data[['Y6', 'Y7']]

# Mechanical performance optimization model
model_mechanical = XGBRegressor()
params_mechanical = {'n_estimators': [100, 200], 'learning_rate': [0.01,
0.1]}
grid_mechanical = GridSearchCV(model_mechanical, params_mechanical, cv=3)
grid_mechanical.fit(X, y_mechanical)

# Thermal comfort performance optimization model
```

```
model_thermal_comfort = XGBRegressor()
params_thermal_comfort = {'n_estimators': [100, 200], 'learning_rate':
[0.01, 0.1]}
grid_thermal_comfort = GridSearchCV(model_thermal_comfort,
params_thermal_comfort, cv=3)
grid_thermal_comfort.fit(X, y_thermal_comfort)

# Softness performance optimization model
model_softness = RandomForestRegressor()
params_softness = {'n_estimators': [100, 200], 'max_depth': [10, 20]}
grid_softness = GridSearchCV(model_softness, params_softness, cv=3)
grid_softness.fit(X, y_softness)

# Get best parameter combinations
best_mechanical_params = grid_mechanical.best_params_
best_thermal_comfort_params = grid_thermal_comfort.best_params_
best_softness_params = grid_softness.best_params_

print("Best mechanical performance parameters:", best_mechanical_params)
print("Best thermal comfort performance parameters:",
best_thermal_comfort_params)
print("Best softness performance parameters:", best_softness_params)

# Best parameter combinations stored in the dictionary below
optimal_params = {
    'mechanical': {
        'model': 'XGBoost',
        'performance': (1742.01, 1.12, 173.22),
        'resin_content': 15.76,
        'curing_temp': 100.73,
        'alkali_reduction': 0.01
    },
    'thermal_comfort': {
        'model': 'XGBoost',
        'performance': (319.43, 3477.85),
        'resin_content': 19.18,
        'curing_temp': 101.51,
        'alkali_reduction': 0.13
    },
    'softness': {
        'model': 'RandomForest',
        'performance': (3.29, 169.92),
        'resin_content': 16.22,
        'curing_temp': 128.34,
        'alkali_reduction': 0.29
    },
    'comprehensive': {
        'model': 'LightGBM',
        'resin_content': 26.74,
        'curing_temp': 119.62,
        'alkali_reduction': 0.02,
        'performance': (1742.36, 1.17, 159.21, 326.72, 3419.08, 3.44,
171.25)
    }
}
```

```
# Output results in table format
for key, value in optimal_params.items():
    print(f"Optimization type: {key}")
    print(f"Best model: {value['model']}")
    print(f"Performance: {value['performance']}")
    print(f"Resin content: {value['resin_content']}")
    print(f"Curing temperature: {value['curing_temp']}")
    print(f"Alkali reduction: {value['alkali_reduction']}")
        print("\n")
```

B.4 问题四源代码

Prob4.py

```
import numpy as np
from scipy.optimize import minimize

# Define weights from AHP and entropy method
weights_criteria = np.array([0.1125, 0.7089, 0.1785])  # Mechanical,
Thermal Comfort, Flexibility

weights_subcriteria = np.array([0.278, 0.068, 0.185, 0.135, 0.125, 0.121,
0.088])  # Subcriteria weights

# Function to compute Y12 as a combination of subcriteria
def calculate_Y12(parameters):
    # Assuming parameters include resin content, curing temperature,
alkali reduction level
    resin_content, curing_temp, alkali_reduction = parameters

    # Example formula for Y12 (adjust based on specific formula derived)
    Y12 = resin_content * weights_subcriteria[0] + curing_temp *
weights_subcriteria[1] + alkali_reduction * weights_subcriteria[2]
    + ...  # Include other subcriteria terms similarly

    return Y12

# Example objective function to minimize Y12
def objective_function(parameters):
    Y12 = calculate_Y12(parameters)
    return Y12

# Initial guess for parameters (resin content, curing temperature, alkali
reduction)
initial_guess = np.array([21.46, 113.15, 0.04])

# Constraints (if any) can be added here based on specific problem
requirements

# Perform optimization using Particle Swarm Optimization (PSO) or other
suitable method
result = minimize(objective_function, initial_guess, method='Nelder-Mead')
```