属类别	2022 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2200949

基于支持向量机与插值回归模型的插层熔喷法参数调控机制

摘要

新冠疫情爆发以来，全国人民齐动员，上下一心，抗击新冠病毒。熔喷非织造材料是口罩生产的重要原材料，具有良好的过滤性能。研究插层熔喷法在口罩制造生产过程中的工艺参数、结构变量、产品性能之间的关系，有助于为产品调控机制的建立提供一定的理论基础，对于打赢这场抗击新冠病毒的全民战役有着重要的现实意义。

本文以工艺参数、结构变量、产品性能的实验数据为研究对象，在合理假设的基础上建立关于插层熔喷非纺造材料制备过程的参数理论模型。

针对问题一，从数据分析、数据预处理、模型理论解释、模型的建立与求解、数据的拟合等五个方面开展了探索性分析。我们建立了**基于支持向量机的回归模型**，分析不同插层率下结构变量、产品性能数据的变化规律，通过拟合出的曲线，得出结论：经过插层熔融得到的非纺造材料相较未插层材料的所有结构变量、过滤效率和透气性均有所增长，过滤阻力减小；随着插层率的增大，厚度逐渐增大，孔隙率、过滤阻力逐渐减小，对压缩回弹性率的影响先减小后增大。插层率对于过滤效率、透气性没有明显影响。

针对问题二，首先**基于贝叶斯数据融合**将同一工艺参数组合下的结构变量与产品性能数据进行预处理，然后建立了**二维三阶样条模型**进行插值拟合。随着接收距离、热风速度的增大，样品的厚度逐渐增大。当接收距离在 30-35cm 时，孔隙率上升平缓；当接收距离在 20-30cm、35-40cm 时，孔隙率上升明显。当热风速度在 1000-1100r/min 时，孔隙率上升平缓；当热风速度在 800-1000r/min、1100-1250r/min 时孔隙率上升明显。压缩回弹性率与工艺参数的关系图形似一个上凸的抛物面，经过计算得出，极值点为(27,990,88.5708)。

针对问题三，首先**基于主成分模型**和**相关性模型**进行**类外关联度分析**与**类内关联度分析**，然后运用四种插值方法，选取拟合程度最优的**线性插值**对两种主成分进行回归，得到结构变量与产品性能的关系：当结构变量主成分增大时，产品性能主成分先增大，后减小，呈现波动下滑的趋势。通过 SPSS 相关性分析得到结果：在结构变量内，厚度与孔隙率存在着严格强烈的正相关关系，且厚度与压缩回弹性率存在严格但并不强烈的负相关关系；在产品性能中，过滤效率与透气性存在严格强烈的负相关关系。同样通过二维三阶样条插值得到：理论上，当接收距离等于 20cm，热风速度等于 1200r/min，过滤效率可以达到最高。

针对问题四，根据实际生产中的各方面条件和要求，通过**模糊数学理论**，我们合理限制厚度、压缩回弹性率、透气性的范围，最终得到实际生产中过滤效率尽可能高，同时力求过滤阻力尽可能小的最优工艺参数组合：接收距离等于 23cm，热风速度等于 1090r/min。

关键词：支持向量回归 SVR 贝叶斯数据融合 二维三阶样条 主成分分析 相关性分析

一、问题重述

伴随着新冠疫情的爆发，寻找一种新方法制备过滤性能良好、成本低廉的新型口罩材料成为各家公司的研究目标。原本的熔喷非织造材料具有很好的过滤性能、生产工艺简单、成本低、质量轻等特点，但是由于熔喷非织造材料纤维细，经常因为压缩回弹性差而导致其性能得不到保障。于是，科学家研究出插层熔喷法,通过在熔喷制备聚丙烯过程中将涤纶短纤等纤维插入熔喷纤维流，制备出了“Z 型”结构的插层熔喷非织造材料。

但是插层熔喷法也带来了产品性能调控的麻烦，因为插层熔喷制备过程中工艺参数繁多，各种参数之间还存在着交互影响，引入插层气流这个因素之后更为复杂。

插层熔喷非织造材料的制备模型中，主要的影响因子有三个：

①工艺参数（接收距离和热空气速度）

②结构变量（厚度、孔隙率、压缩回弹性）

③产品性能（过滤阻力、过滤效率、透气性）

图 1

因此，在充分发挥该方法制备材料的性能控制研究附件的数据，建立工艺参数与结构变量、结构变量和产品性能之间的关系模型，为产品性能调控机制的建立提供一定的理论基础，解决以下问题：

问题一：研究插层后结构变量、产品性能的变化规律，分析插层率对于变化是否有影响。

问题二：研究工艺参数与结构变量之间关系，预测新所给工艺参数组合的结构变量数据。

问题三： ①研究结构变量与产品性能的关系。
②各个结构变量之间、产品性能之间的关系。
③研究当工艺参数为多少时，产品的过滤效率将会达到最高。

问题四：根据实际生产的要求，厚度尽量不要超过 3mm，压缩回弹性率尽量不要低于 85%. 研究当工艺参数为多少时，使得过滤效率尽量的高的同时力求过滤阻力尽量的小。

二、问题分析

问题的研究对象是熔喷非织造材料，研究内容为插层熔喷法制备过程中的工艺参数、结构变量、产品性能，分别建立两两之间的关系模型，为产品性能调控机制的建立提供一定的理论基础。

问题一：此题探究的是插层前后，结构变量与产品性能的变化规律，并且探究插层率的影响。要研究数据变化趋势，需要使用适当的回归方法，根据数据具有数据量小、非线性、局部极小点的特点，我们使用支持向量回归的方法进行求解。

问题二：我们观察 data3 数据，发现是在插层率固定的条件下的，使用不同组合的工艺参数，每个组合重复实验三次，得到结构变量数据和产品性能数据。使用贝叶斯数据融合进行数据处理。我们需要从离散化的数据中，预测出连续的变化趋势，从而获取各种不同工艺参数组合下的理论值，这里采用二维三阶样条插值模型进行求解。

问题三：数据成分维度多，我们使用主成分分析进行降维，使用四种插值算法拟合曲线，得到两种主成分的关系；接着运用相关性模型分析结构变量之间、产品性能之间的关系；同样运用三阶样条插值模型探索工艺参数的最优化方案。

问题四：为满足实际生产的需要，运用线性规划、模糊数学理论，合理限制理论条件，最终得到最佳的实际工艺参数。

三、符号说明

符号	意义
N_{ij}	未插层的组别第 i 组第 j 个属性
I_{ij}	插层后的组别第 i 组第 j 个属性
C_i	对应组别的插层率
D_{ij}	N_{ij}与I_{ij}的比值
μ_{ij} σ_{ij}	同一组数据的第 i 个和第 j 个的均值与方差
μ_0 σ_0	同一组中所有的数据的均值与方差
U_{ij}	第 i 组数据的第 j 个属性的贝叶斯均值
L_i V_i	接收距离与热风速度
$F1$ $F2$	结构变量与产品性能的主成分
E_{ij}	第 i 个属性与第 j 个属性的相关性

四、模型的建立及求解

4.1 问题一的分析与求解

4.1.1 不同插层率下结构变量、产品性能数据的探索性分析

（1）数据分析

在第一题中我们要研究的是插层前后结构变量、产品性能的变化规律，并且探究不同的插层率对这些变化的影响。

对结构变量和产品性能进行探索性数据分析，从中提取有用信息，从而根据样本去研究变化规律，从而去推断插层率对这些变化是否有影响。

首先观察所给 C 题数据.xlsx 中的 data1 数据，发现是为了探究接受距离、热风速度的最佳工艺配置，设置的二因素五水平实验，并且每组平行实验，分别设置 1#表示未插层材料，2#表示插层熔喷材料的对比试验。

（2）数据预处理

为了避免多组数据的分组比较，我们采用了合理的数据预处理方式：
将 25 组平行实验拆分为 25 组未插层材料试验数据，25 组插层熔喷材料试验数据（存放于未插层.xlsx、插层.xlsx）。

设未插层的组别第 i 组第 j 个属性记为$\boldsymbol{N_{ij}}$,插层后的组别第 i 组第 j 个属性记为

$\boldsymbol{I_{ij}}$设对应组别的插层率为$\boldsymbol{C_i}$

（规定属性顺序为：厚度、孔隙率、压缩回弹性率、过滤阻力、过滤效率、透气性）

4.1.2 模型建立：支持向量机的回归模型

（1）模型理论解释

由于本题所给数据量少，且规律并不明显，我们这里采用支持向量机进行回归拟合（简称支持向量回归）

支持向量回归适用于：数据量小，非线性，局部极小点

支持向量机（SVM）是建立在统计学习理论基础上的一种数据挖掘方法。SVM 的机理是寻找一个满足分类要求的最优分类超平面，使得该超平面在保证分类精度的同时，能够使超平面两侧的空白区域最大化。理论上，支持向量机能够实现对线性可分数据的最优分类[1]。

（2）模型的建立与求解

使用同一组别中的结构变量、产品性能数据编号 1#与编号 2#相除，得到 25 组未插层数据与插层数据的比值（存放于比率数据.xlsx）。我们将所需要的变量比值记作

$\boldsymbol{D_{ij}}$

$$D_{ij} = \frac{N_{ij}}{I_{ij}}$$

SVR 回归是找到一个回归曲线，使得数据点到回归直线的距离之和最小，并且使得最远处的样本点与回归曲线的距离最近。

在本题中，我们在使用支持向量回归前，先**筛除了部分异常值**（提高支持向量回归得到的回归曲线的可信度）

在支持向量回归中，我们使用的距离是默认的欧式距离，记作 d

$$d = \sqrt{(C_{i1} - C_{i2})^2 + \left(D_{i1j} - D_{i2j}\right)^2}$$

为使不同属性数据都拟合出较好的曲线，对于不同的属性我们使用了不同的核函数进行拟合：

$$k(x_i, x_j) = \left(x_i^T x_j\right)^d \ (d > 0)$$

高斯核函数

$$k(x_i, x_j) = e^{\left(-\frac{\left|x_i - x_j\right|^2}{2\sigma^2}\right)} \ (\sigma > 0)$$

多项式核函数

我们在这里选用$\upsilon - SVR$的方式进行数据回归，其最优化函数为：

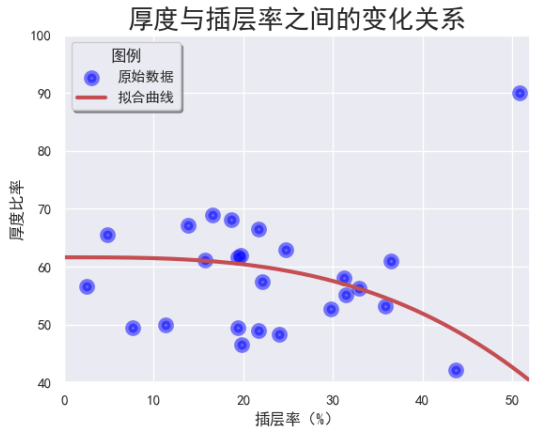
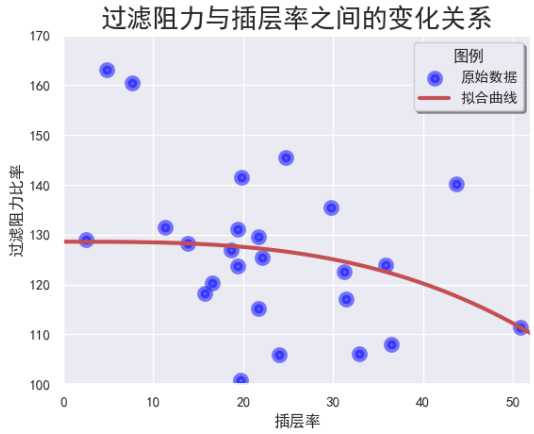
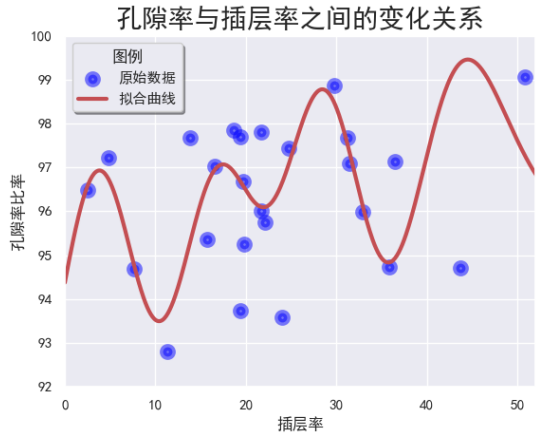
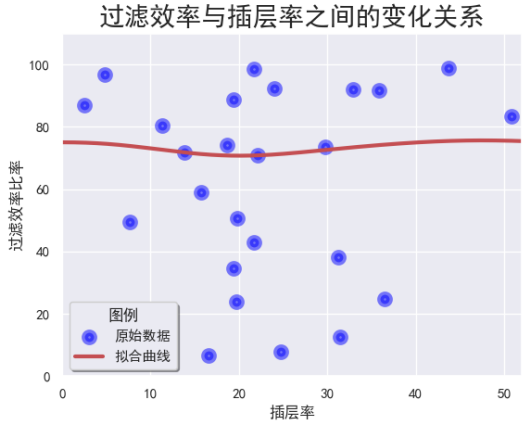
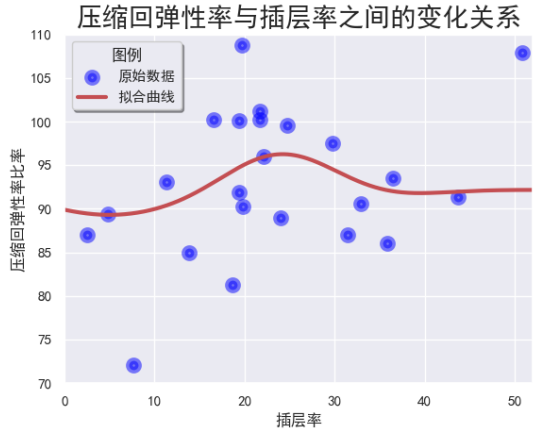
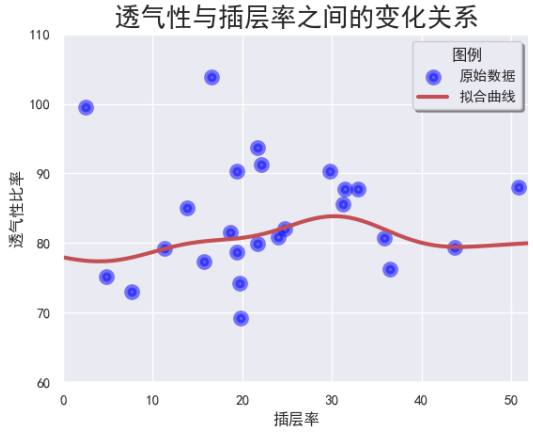
$$min_{w,b,\xi*,\xi,\epsilon}\left(\frac{1}{2}w^T w + \upsilon\varepsilon + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*)\right)$$

（其中 C 为支持向量机的软间隔）

在支持向量回归过程中：对于不同类型的数据，我们采用了不同类型的核函数，并调节多项式核函数的维度 degree 以及各个核函数中的软间隔 C 以达到对不同类型数据均具有最佳的回归效果的目的。

（3）插层、未插层数据的散点图及拟合曲线的表示

（PS：图表的内容均为未插层/插层）

结构变量变化规律	产品性能变化规律
	
	
	

4.1.3 探索性分析结果及讨论

由上述散点图及其拟合曲线我们可以得出结论：

1） **厚度：**插层使得厚度增大，且随着插层率的增大，厚度逐渐增大

2) **孔隙率**：插层使孔隙率的值增大，但随着插层率的增大，插层率的增加效果逐渐减小

3) **压缩回弹性率**：插层对压缩回弹的影响较小，但仍然会使得其值增大，并且随着插层率的增大，此影响先减小后增大

4) **过滤阻力**：插层使得产品的过滤阻力减小，但随着插层率的增大，此效果逐渐减小

5) **过滤效率**：插层使得产品的过滤效果增强，但此效果与插层率的大小基本无关

6) **透气性**：插层使得产品的透气性增强，但此效果与插层率的大小基本无关

4.2 问题二的分析与求解

4.2.1 工艺参数与结构变量数据的探索性分析

（1）数据分析

第二题我们要研究的是工艺参数与结构变量之间的关系，根据新给出的 8 个工艺参数组合，预测结构变量。

观察 C 题数据.xlsx 的 data3，发现是在插层率固定的条件下的，不同工艺参数组合的材料结构变量数据和产品性能数据，每个组合实验重复了三次。

使用 data3 数据集前的重要说明：

data3 中的每一行不能独立使用，因为是每一组的三个复制之一。数据的性质取决于每个工艺参数的搭配，但独立于每一个工艺参数。因此，机器学习的传统技术不能应用于该数据集，因为这些技术是基于实例的独立性。这里考虑的复制概念与统计重复测量的经典概念不匹配。术语“复制”是指从属于同一工艺参数搭配的工业流程中提取的特征的集合。由于在这种情况下，特征是从同一主题的多个连续工业流程中提取的，因此原则上特征应该是相同的。技术的不完善和变异性导致了不完全相同的重复特征，这些特征彼此之间比来自不同工业流程的特征更相似。

（2）数据预处理

在此基础上，我们使用贝叶斯数据融合而不是直接求平均值来将三组相同工艺参数搭配的数据进行合并，存放于 data3(副本).xlsx。

我们设 data3 中的相同的工艺参数的三行数据为同一组。

同一组数据的第 i 个和第 j 个的均值为$\boldsymbol{\mu_{ij}}$，同一组中所有的数据的均值为$\boldsymbol{\mu_0}$；对应的方差为$\boldsymbol{\sigma_{ij}}$和$\boldsymbol{\sigma_0}$，经过贝叶斯数据处理后的“均值”为$\boldsymbol{\mu'}$

$$\mu' = \frac{\Sigma_{i=1}^{k}\Sigma_{j=1}^{k}\frac{\mu_{ij}}{\sigma_{ij}^2} + \frac{\mu_0}{\sigma_0^2}}{\Sigma_{i=1}^{k}\Sigma_{j=1}^{k}\frac{1}{\sigma_{ij}^2} + \frac{1}{\sigma_0^2}}$$

贝叶斯数据融合公式

在此，我们让第 i 组数据的第 j 个属性为U_{ij}，每个U_{ij}都经过上述的公式计算得到，在第二题、第三题与第四题中计算数据均使用U_{ij}

4.2.2 模型建立：二维三阶样条插值模型

（1）模型理论解释

在问题 2 中，我们需要使用 data3 中离散的工艺参数与结构变量的数据，预测出连续变化趋势，从而获取各个结构变量在不同工艺参数搭配下的理论值。

三阶样条插值适用于：数据量小，对预测区间中部精度要求高

在问题 2 中，每一个结构变量都受到两个工艺参数的影响，因此，我们需要采用二维三阶样条的方法进行插值拟合。

利用样条插值，既可以保持分段低次插值多项式，又可以提高插值函数光滑性[2]。样条插值模型具有参数少、简洁、精度高等优点[3]。

我们设现在需要求解的三阶样条是第 i 组与第 j 组的第 k 个属性数据中的点，并设第 i 组和第 j 组的工艺参数（接收距离、热风速度）分别为（L_i，L_j，V_i，V_j），设二维三阶样条方程为

$$f(x,y) = a_1x^3 + a_2y^3 + a_3x^2 + a_4y^2 + a_5x + a_6y + a_7$$

那么该方程有以下的限定：

$$f(L_i,V_i) = f(L_j,V_j) = U_{ij}$$

$$f_x^{'}(L_i,V_i) = f_x^{'}(L_j,V_j)$$

$$f_y^{'}(L_i,V_i) = f_y^{'}(L_j,V_j)$$

并且相邻的顶点二阶导数必须连续（即计算得出的$\boldsymbol{f}$：后面设为$\boldsymbol{f_1}$，$\boldsymbol{f_2}$，若它们计算的来源是相邻的顶点，那他们的二阶导数值必须连续)

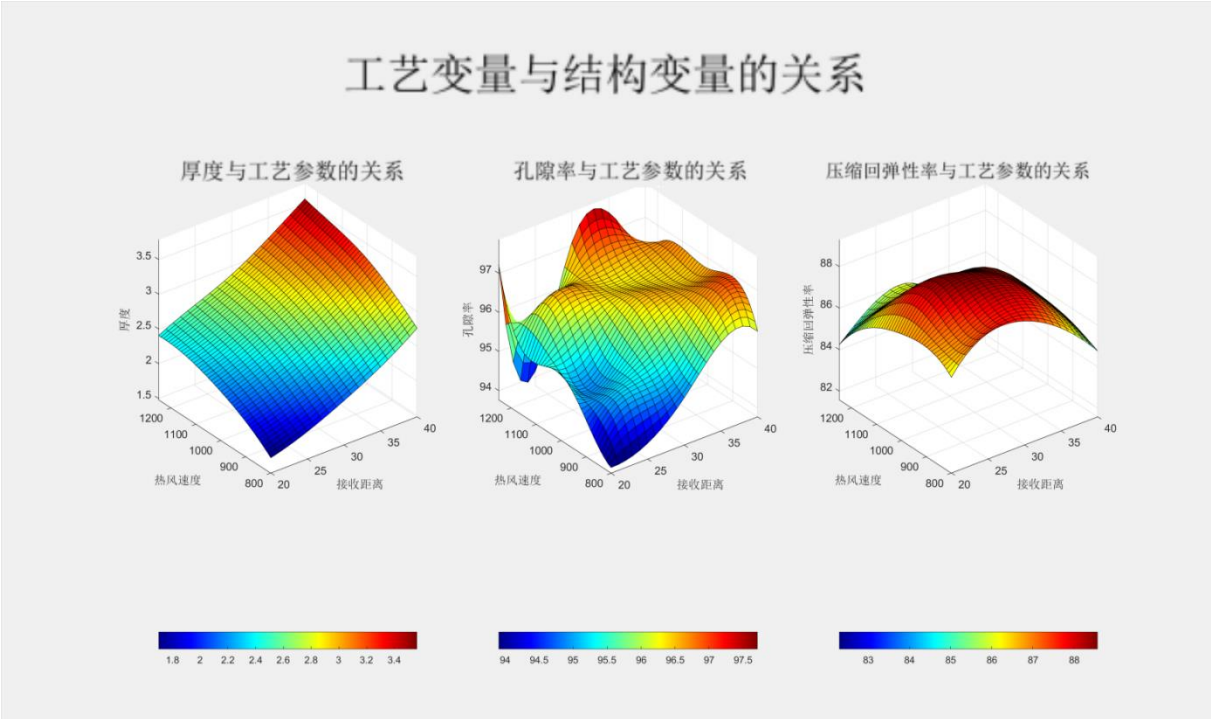
$$\frac{\boldsymbol{d^2 f_1}}{\boldsymbol{dx^2}} = \frac{\boldsymbol{d^2 f_2}}{\boldsymbol{dx^2}}$$

$$\frac{\boldsymbol{d^2 f_1}}{\boldsymbol{dxdy}} = \frac{\boldsymbol{d^2 f_2}}{\boldsymbol{dxdy}}$$

$$\frac{\boldsymbol{d^2 f_1}}{\boldsymbol{dy^2}} = \frac{\boldsymbol{d^2 f_2}}{\boldsymbol{dy^2}}$$

（2） 工艺参数与结构变量关系可视化及三维曲面色图的表示

由上述公式可以求出函数系数，并大致拟合出工艺参数与各个结构变量之间的“函数$\boldsymbol{f}$”，经由 **matlab** 可以得到相应的函数图像与对连续值的预测结果



4.2.3 探索性分析结果

由上述三维曲面色图我们可以得出结论：

1）由厚度与工艺参数的关系图可以看出：随着接收距离、热风速度的增大，样品的厚度

逐渐增大。

2）由孔隙率与工艺参数的关系图可以看出：当接收距离在 30-35cm 时，孔隙率上升平缓；当接收距离在 20-30cm、35-40cm 时，孔隙率上升明显。当热风速度在 1000-1100r/min 时，孔隙率上升平缓；当热风速度在 800-1000r/min、1100-1250r/min 时孔隙率上升明显。

3）由压缩回弹性率与工艺参数的关系图可以看出：二者的图像形似一个上凸的抛物面，经过计算得出，极值点为(27,990,88.5708)

问题 2 的结果

接收距离 (cm)	热风速度 (r/min)	厚度 mm	孔隙率 (%)	压缩回弹性率 (%)
38	850	2.6561	96.6881	86.2718
33	950	2.6034	96.2979	88.0909
28	1150	2.6569	96.584	86.8837
23	1250	2.5028	94.9258	84.9877
38	1250	3.3321	96.8939	83.7457
33	1150	2.9318	96.3445	86.8836
28	950	2.3401	95.4959	88.3651
23	850	1.8856	94.6767	86.8978

表 1

4.3 问题三的分析与求解

4.3.1 基于主成分的类外关联度分析

4.3.1.1 符号假设

我们要研究的是结构变量与产品性能之间的关系。

我们设结构变量主成分记为$\boldsymbol{F1}$，产品性能主成分记为$\boldsymbol{F2}$以方便后续的表示。

调用流行甚广的 SPSS 软件，对数据进行主成分分析。

4.3.1.2 模型建立：主成分分析(PCA)模型

（1）模型理论解释

主成分分析法是通过恰当的数学变换，使新变量主成分成为原变量的线性组合，并选取少数几个在变差总信息量中比例较大的主成分来分析事物的一种方法。主成分在变差信息量中的比例越大，它在综合评价中的作用就越大[4]。

用主成分分析法确定权数有以下优点：

①可消除评价指标之间的相关影响．因为主成分分析在对原指标变量进行变换后形成了彼此相互独立的主成分，而且实践证明指标间相关程度越高，主成分分析效果越好。

②主成分分析中各主成分是按方差大小依次排列顺序的，在分析问题时，可以舍弃一部分主成分，只取前后方差较大的几个主成分来代表原变量，从而减少了计算工作量[4]。

（2）模型的建立与求解

①结构变量

结构变量因子载荷系数

名称	因子载荷系数	共同度（公因子方差）
厚度(mm)	0.966	0.934
孔隙率（%）	0.906	0.821
压缩回弹性率（%）	-0.675	0.456

表 2

结构变量成分矩阵

名称	各成分在 F1 中的占比
厚度 mm	0.437
孔隙率（%）	0.41
压缩回弹性率（%）	-0.305

表 3

结构变量主成分公式

$$F1_i = 0.437U_{i1} + 0.41U_{i2} - 0.305U_{i3}$$

②产品性能

产品性能因子载荷系数

名称	因子载荷系数	共同度（公因子方差）
过滤阻力 Pa	0.642	0.413
过滤效率（%）	0.909	0.826
透气性 mm/s	-0.905	0.820

表 4

产品性能成分矩阵

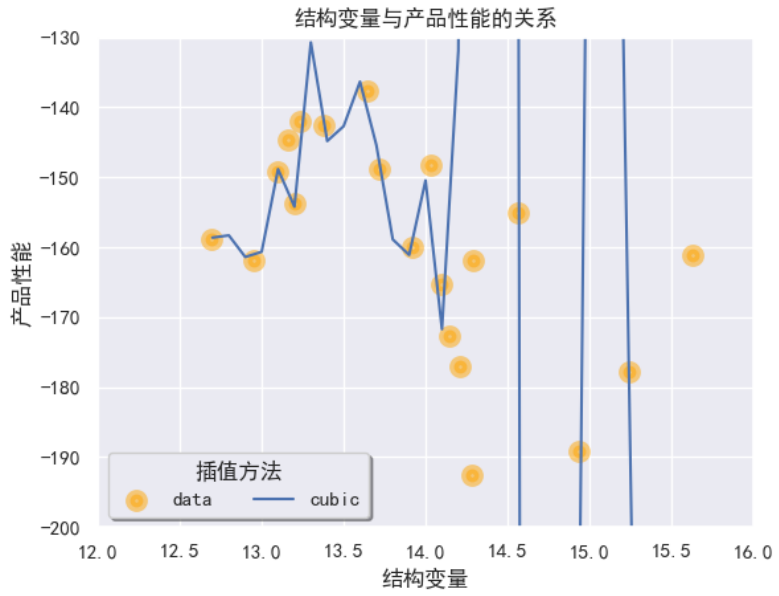
名称	各成分在 F2 中的占比
过滤阻力 Pa	0.312
过滤效率（%）	0.442
透气性 mm/s	-0.44

表 5

产品性能主成分公式

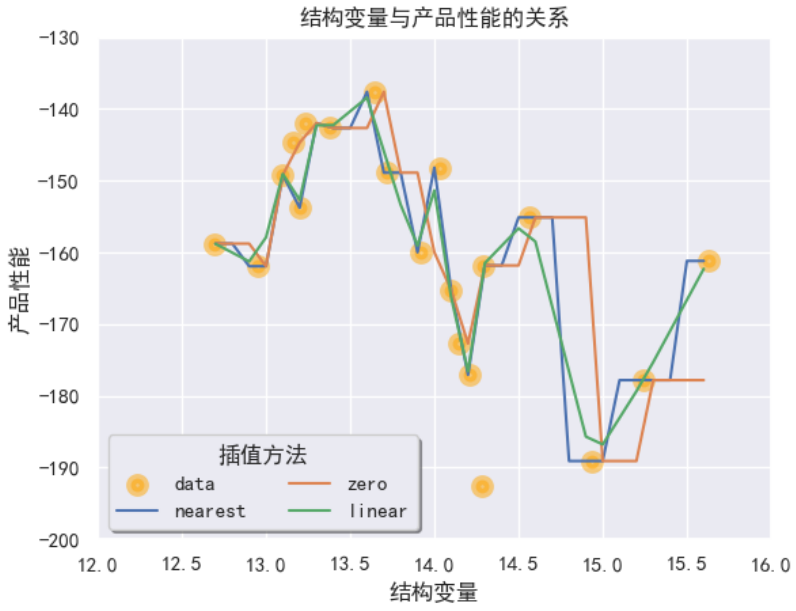
$$F2_i = 0.312U_{i4} + 0.442U_{i5} - 0.44U_{i6}$$

（3）多维角度进行插值拟合曲线的表示



①三阶样条插值

可以从图中明显看出，对于结构变量的主成分小于 14.0 前，拟合效果较好。缺点：在结构变量的主成分大于 14.0 后，三阶样条插值拟合产生的函数的二阶导数不连续，产品性能主成分的数值具有较高的波动，与实际值相差较大。因此，我们不采用三阶样条插值的方法。



②zero（阶梯插值）：欠拟合，数据偏差较大。

③nearest（最近邻插值）：经过了绝大多数的数据点，偏差过小，属于过拟合的现象

④linear 插值（线性插值）：

我们通过计算与观察，上述两种方法产生的拟合曲线平滑程度较小（即其一阶导数存在较多的间断点，且间断点的数据相差较大），为使得拟合曲线较为平滑以方便结构变量与产品性能关系的比较，我们采用的是 linear 插值（线性插值）

4.3.1.3 类外关联度分析结果

主成分分析总结

主成分的总体趋势是：当$F1$增大时，$F2$先增大，后减小，呈现波动下滑的趋势。

我们可以通过$F1$与$F2$之间的关系判断各个结构变量与各个产品性能之间的关系，即当某一个结构变量变化时，会影响到整体的结构变量主成分的改变，进而改变产品性能的主成分，从而影响其成员成分的量化变化。如：当产品厚度逐渐增大的时候，会导致$F1$的逐步上升，使得主成分$F2$整体波动下滑，使得产品性能中的过滤阻力也逐步产生波动下滑的趋势。

4.3.2 基于相关性的类内关联度分析

4.3.2.1 符号假设

我们要研究的是各个结构变量之间、产品性能之间的关系。

在此我们设第 i 个属性与第 j 个属性的相关性为$\boldsymbol{E_{ij}}$

依旧调用 SPSS 软件，对数据进行关联度分析。

4.3.2.2 模型建立：相关性分析模型

（1）模型理论解释

1.先对不同变量之间是否存在统计上的显著关系（p 值小于 0.05 或 0.01，严格为 0.01，不严格为 0.05）进行检验。

2.分析相关系数为的正负向以及相关性程度。

3.对分析结果进行总结。

（2）模型的建立与求解

厚度、孔隙率、压缩回弹性率的相关性矩阵

$$\begin{pmatrix} \boldsymbol{E_{11}} & \boldsymbol{E_{12}} & \boldsymbol{E_{13}} \\ \boldsymbol{E_{21}} & \boldsymbol{E_{22}} & \boldsymbol{E_{23}} \\ \boldsymbol{E_{31}} & \boldsymbol{E_{32}} & \boldsymbol{E_{33}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{1(0)} & \boldsymbol{0.898(0)} & \boldsymbol{-0.527(0.007)} \\ \boldsymbol{0.089(0)} & \boldsymbol{1(0)} & \boldsymbol{-0.340(0.097)} \\ \boldsymbol{-0.527(0.007)} & \boldsymbol{-0.340(0.097)} & \boldsymbol{1(0)} \end{pmatrix}$$

过滤压力、过滤效率、透气性的相关性矩阵

$$\begin{pmatrix} E_{44} & E_{45} & E_{46} \\ E_{54} & E_{55} & E_{56} \\ E_{64} & E_{65} & E_{66} \end{pmatrix} = \begin{pmatrix} 1(0) & 0.380(0.061) & -0.370(0.069) \\ 0.380(0.061) & 1(0) & -0.793(0) \\ -0.370(0.069) & -0.793(0) & 1(0) \end{pmatrix}$$

括号内左边的数值为相关性，符号为正表示正相关，反之为负相关，其值的绝对值越接近 1，表示相关性越强；括号内的值为 p 值（用于表示相关性的可信度），当$\boldsymbol{p} < 0.05$时，相关性为可信的。当$\boldsymbol{p} < 0.01$时，说明相关性的可信度极高。当$\boldsymbol{p} \geq 0.05$说明量变量相关性很小，可以近似的认为两变量在统计学中不存在相关性。

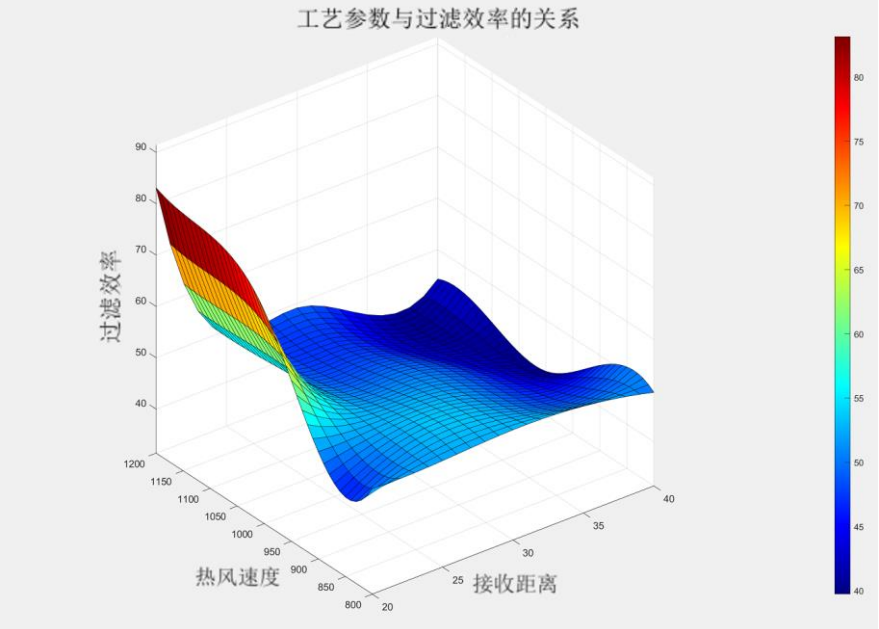
4.3.2.3 类内关联度分析结果

根据分析得到的相关性矩阵，我们$\boldsymbol{p} < 0.05$的数据，即统计学上存在相关性的数据。从数据中可以看出，在结构变量内，厚度与孔隙率存在着严格强烈的正相关关系，且厚度与压缩回弹性率存在严格但并不强烈的负相关关系；在产品性能中，过滤效率与透气性存在严格强烈的负相关关系。

4.3.3 基于三阶样条插值的最优化方案探索

4.3.3.1 工艺参数与过滤效率关系可视化及三维曲面色图的表示

结合第二问的模型：二维三阶样条插值与第二问经过贝叶斯数据融合后的 data3（副本）.xlsx，得到工艺参数与过滤效率的关系。



4.3.3.2 理论最优化方案

从图中可以明显看出，当$L_i = 20$，$V_i = 1200$，理论上，过滤效率可以达到最高。

4.4 问题四的分析与求解

4.4.1 题目分析

按照实际生产应用的要求，我们要研究的是厚度小于等于 3mm，压缩回弹性率大于等于 85%
初步控制：

1. 厚度小于等于 3mm：
   通过第二题得到的厚度与工艺参数的三维曲面色图分析设置线性方程组：（其中左侧的矩阵为使得厚度为 3 的$\boldsymbol{L_i}$与$\boldsymbol{V_i}$的两组值）

$$\begin{pmatrix} \mathbf{32} & \mathbf{1250} \\ \mathbf{40} & \mathbf{880} \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} = \begin{pmatrix} \mathbf{3} \\ \mathbf{3} \end{pmatrix}$$

最终工艺参数对应范围

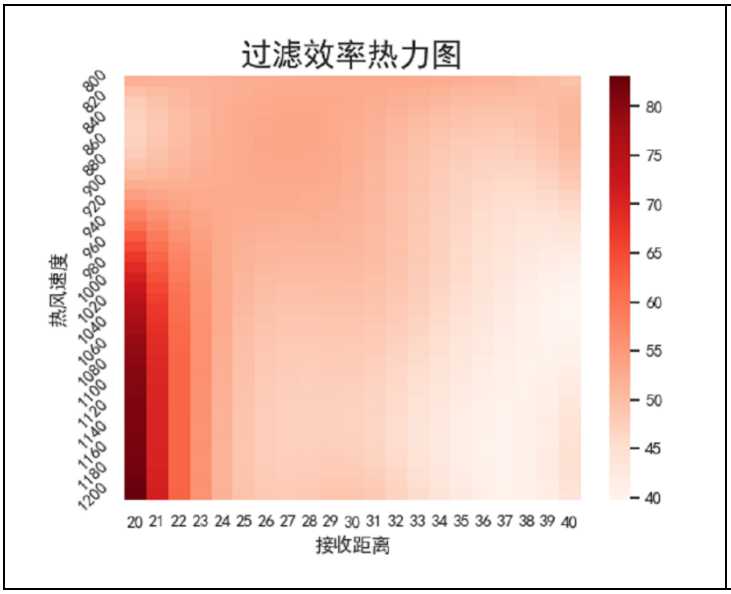
$$\boldsymbol{aL_i + bV_i \leq 3}$$

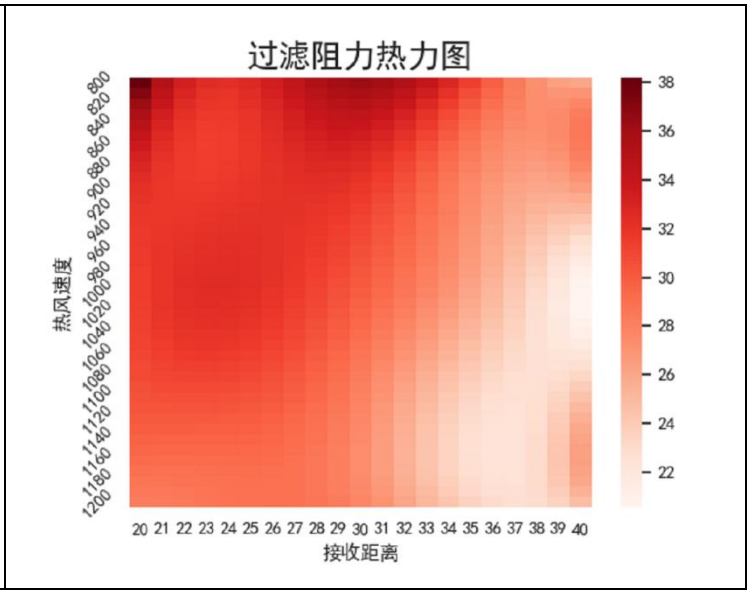
得到$\boldsymbol{a} = 0.0508$， $\boldsymbol{b} = 0.0011$

2. 压缩回弹性率大于等于 85%(根据 data3(副本).xlsx 数据以及相应属性的三维曲面色图，下同)：由此我们限制：$\boldsymbol{L_i} <= 38$

3. 根据数据集以及理论上的分析，我们可以进行一些合理的假设：
   理论上，透气性越大越好，我们合理地假设透气性大于 450 为优。
   据此我们得到，透气性限制：$23 \leq \boldsymbol{L_i} \leq 35$，$900 \leq \boldsymbol{V_i} \leq 1100$

根据工艺参数与过滤效率热力图进行分析





4. 根据过滤效率较高、过滤阻力较低的限制与相应热力图的限制：

接收距离与过滤效率：当$\boldsymbol{L_i}$小于 25 后，过滤效率才会出现明显的上升（25cm：50%左右，20cm：可达 80%以上）；热风速度与过滤效率：$\boldsymbol{V_i}$大于 950 后，过滤效率才会明显上升；在所给数据范围中再次确定范围：$\boldsymbol{L_i \leq 25}$，$\boldsymbol{V_i \geq 950}$

最终确定范围：

根据上述分析，得到最终限制范围为：$\boldsymbol{23 \leq L_i \leq 25}$；$\boldsymbol{950 \leq V_i \leq 1100}$（其内所有数据均满足：$\boldsymbol{aL_i + bV_i \leq 3}$）

4. 4. 2 模型求解

根据上述分析得到的限制范围，我们进行了工艺参数的限制，得到相应范围内的过滤阻力与过滤效率的值。

不同工艺参数组合下的过滤阻力

热风速度 接收距离	23	24	25
950	32.240306	32.302583	32.289106

960	32.340697	32.386638	32.337542
970	32.421813	32.450761	32.368505
980	32.477706	32.489586	32.377253
990	32.502429	32.497745	32.359046
1000	32.490033	32.469873	32.309143
1010	32.436248	32.402199	32.224255
1020	32.343513	32.297341	32.106897
1030	32.215942	32.159514	31.961036
1040	32.057651	31.992932	31.790639
1050	31.872756	31.801812	31.599672
1060	31.665372	31.590367	31.392103
1070	31.439614	31.362812	31.171897
1080	31.199597	31.123362	30.943021
1090	30.949438	30.876233	30.709443
1100	30.693251	30.625639	30.475128

表 6

不同工艺参数组合下的过滤效率

热风速度 接收距离	23	24	25
950	54.141694	52.836799	52.073836
960	54.490670	52.863977	51.896083
970	54.822624	52.880143	51.710594
980	55.130151	52.883964	51.519789
990	55.405843	52.874108	51.326085
1000	55.642295	52.849242	51.131902
1010	55.833997	52.808615	50.939477
1020	55.983030	52.753789	50.750324
1030	56.093369	52.686908	50.565779
1040	56.168992	52.610115	50.387175
1050	56.213876	52.525552	50.215846
1060	56.231997	52.435362	50.053125
1070	56.227332	52.341690	49.900348
1080	56.203857	52.246676	49.758848
1090	56.165550	52.152465	49.629958
1100	56.116387	52.061200	49.515014

表 7

4.4.1 结合实际得到的最优化方案

通过上述表格，我们需要在插值结果中筛选出在其中找到效率尽可能高，阻力尽可能小的值。在工业上认为过滤效率高所占的权重更高，经过综合考量，我们得到结果：

$\boldsymbol{L_i = 23，\ V_i = 1090}$所对应的组别为最佳的实际工艺参数。

五、模型的评价与推广

5.1 模型评价

5.1.1 模型优点

1) 在本题中，我们使用了支持向量机与三阶样条等多种拟合模型，这些模型均具有数据量需求小，系统性、关联性强的特点。

2) 采用题目要求外的实际条件进行限制，使得结果可信度更高，可以细致、精确的评价参数之间的关系。

3) 在解决问题时，融合多个模型进行解决，使得出的结论具有综合性。

5.1.2 模型缺点

1) 在第四题中，我们对透气性的指标进行了模糊数学量化，但其定值具有一定的主观性。
2) 模型时间复杂度较高。

5.2 模型推广

本题中给出的数据量偏小，但在工艺流程中，给出的数据一般是庞大的。当我们获取足够多的数据时，可以采取贝叶斯数据融合先对相同“复制”的数据进行融合；若得到的数据仍然是庞大的，我们可以采用聚类与回归分析的搭配或 bp 神经网络的方法对问题进行求解。

六、参考文献:

[1]. 丁世飞, 齐丙娟与谭红艳, 支持向量机理论与算法研究综述. 电子科技大学学报, 2011. 40(01): 第 1 页.

[2]. 许小勇与钟太勇, 三次样条插值函数的构造与 Matlab 实现. 兵工自动化, 2006(11): 第 1 页.

[3]. 陈岭等, 基于三次样条插值的无线信号强度衰减模型. 浙江大学学报(工学版), 2011. 45(09): 第 1527 页.

[4]. 李艳双曾珍香张闽于树江, 主成分分析法在多指标综合评价方法中的应用. 河北工业大学学报, 1999(01): 第 94-95 页.

[5]. 武辉, 插层熔喷气流场模拟及其过滤材料性能的研究, 2018, 天津工业大学.

附录 A：数据分离预处理

data_pre_processing.py

```python
# 第一题数据预处理
import numpy as np
import pandas as pd

# 导入赛题数据
df = pd.read_excel('C 题数据.xlsx', sheet_name='data1', index_col=0)

# 分离插层前和后的数据
df_of_1 = df.iloc[:49:2, :]
df_of_2 = df.iloc[1::2, :]
df_of_1.index = np.arange(1, 26)
df_of_2.index = np.arange(1, 26)
df_of_1.drop('插层率（%）', axis=1, inplace=True)

# 测试
print(df_of_1)
print(df_of_2)

# 保存为两个新的 excel 表格方便后续操作
df_of_1.to_excel('未插层.xlsx', na_rep=' ')
df_of_2.to_excel('插层.xlsx')
```

附录 B：支持向量回归

intercalating_contrast.py

```python
# 第一题数据分析（支持向量机 SVR 回归分析）
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import seaborn as sns
# 支持向量机的回归函数操作
# 用于非线性，局部极小点的数据回归
from sklearn import svm

# 设置绘图风格
sns.set()
```

```
# 加载中文字体
font = FontProperties(fname="./data/SimHei.ttf", size=14)
plt.rcParams["font.sans-serif"] = ["SimHei"]  # 设置字体
plt.rcParams["axes.unicode_minus"] = False  # 该语句解决图像中的“-”负号的乱码问题

# 导入插层前后的数据
df_of_1 = pd.read_excel('未插层.xlsx')
df_of_2 = pd.read_excel('插层.xlsx')

# 测试
# print(df_of_1)

# 对数据进行对比得出各自增长的比率
df_contrast = df_of_1.iloc[:, 2:8] / df_of_2.iloc[:, 2:8]
df_contrast['插层率（%）'] = df_of_2['插层率（%）']

# 对数据按照插层率进行排序
df_contrast.sort_values(by='插层率（%）
', inplace=True, ascending=True)
# 测试
print(df_contrast)
df_contrast.to_excel('比率数据.xlsx')

# 根据插层率排序的DataFrame 表格，绘出指定的表格

# 厚度
df = pd.DataFrame()
df['厚度 mm'] = df_contrast['厚度 mm'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及 x，y 轴标题
plt.title('厚度与插层率之间的变化关系',fontsize=20)
plt.xlabel('插层率（%）')
plt.ylabel('厚度比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(40, 100)
# 绘制散点图
x = df.index
y = df['厚度 mm']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
```

```
x = x.values
x = x[:, np.newaxis]
y = y.values
# 删除失效点
x = x[0:23]
y = y[0:23]
clf = svm.SVR(kernel='poly', degree=3)
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('厚度与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('厚度与插层率关系.png')
plt.show()

# 孔隙率
df = pd.DataFrame()
df['孔隙率（%）'] = df_contrast['孔隙率（%）'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及x，y轴标题
plt.title('孔隙率与插层率之间的变化关系',fontsize=20)
plt.xlabel('插层率')
plt.ylabel('孔隙率比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(92, 100)
# 绘制散点图
x = df.index
y = df['孔隙率（%）']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
```

```
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
x = x.values
x = x[:, np.newaxis]
y = y.values
# 删除失效点
x = x[0:23]
y = y[0:23]
clf = svm.SVR(C=100)
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('孔隙率与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('孔隙率与插层率关系.png')
plt.show()

# 压缩回弹性率
df = pd.DataFrame()
df['压缩回弹性率（%）'] = df_contrast['压缩回弹性率（%）'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及x，y轴标题
plt.title('压缩回弹性率与插层率之间的变化关系',fontsize=20)
plt.xlabel('插层率')
plt.ylabel('压缩回弹性率比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(70, 110)
# 绘制散点图
x = df.index
```

```
y = df['压缩回弹性率（%）']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
x = x.values
x = x[:, np.newaxis]
y = y.values
# 删除失效点
x = x[0:23]
y = y[0:23]
clf = svm.SVR(C=2)
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('压缩回弹性率与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('压缩回弹性率与插层率关系.png')
plt.show()

# 过滤阻力Pa
df = pd.DataFrame()
df['过滤阻力 Pa'] = df_contrast['过滤阻力 Pa'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及x，y 轴标题
plt.title('过滤阻力与插层率之间的变化关系', fontsize=20)
plt.xlabel('插层率')
plt.ylabel('过滤阻力比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(100, 170)
```

```
# 绘制散点图
x = df.index
y = df['过滤阻力 Pa']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
x = x.values
x = x[:, np.newaxis]
y = y.values
clf = svm.SVR(kernel='poly', degree=3)
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('过滤阻力与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('过滤阻力与插层率关系.png')
plt.show()

# 过滤效率
df = pd.DataFrame()
df['过滤效率（%）'] = df_contrast['过滤效率（%）'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及x，y轴标题
plt.title('过滤效率与插层率之间的变化关系', fontsize=20)
plt.xlabel('插层率')
plt.ylabel('过滤效率比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(0, 110)
# 绘制散点图
```

```
x = df.index
y = df['过滤效率（%）']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
x = x.values
x = x[:, np.newaxis]
y = y.values
clf = svm.SVR()
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('过滤效率与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('过滤效率与插层率关系.png')
plt.show()

# 透气性
df = pd.DataFrame()
df['透气性 mm/s'] = df_contrast['透气性 mm/s'] * 100
df.index = df_contrast['插层率（%）']
# 设置绘图标题以及x，y轴标题
plt.title('透气性与插层率之间的变化关系', fontsize=20)
plt.xlabel('插层率')
plt.ylabel('透气性比率')
# 设置横轴的上下限值
plt.xlim(0, 52)
# 设置纵轴的上下限值
plt.ylim(60, 110)
# 绘制散点图
x = df.index
```

```
y = df['透气性 mm/s']
plt.scatter(x, y, c='blue', s=50, alpha=0.5,
            linewidths=5, norm=1, label='原始数据')
# 数据回归分析
x = x.values
x = x[:, np.newaxis]
y = y.values
# 删除失效点
x = x[1:24]
y = y[1:24]
clf = svm.SVR(C=2)
clf.fit(x, y)
func = []
for i in np.arange(0, 60, 0.1):
    i = np.array(i).reshape(1, -1)
    j = clf.predict(i)
    # 测试
    # print(j[0])
    func.append(j[0])
# 测试
# print(func)
plt.plot(np.arange(0, 60, 0.1), func,
         c='r', linewidth=3.0, label='拟合曲线')
plt.legend(ncol=1,
           title='图例',
           shadow=True,
           fancybox=True)
# 最终得分
# print('透气性与插层率得分: ', clf.score(x, y), sep='')
plt.savefig('透气性与插层率关系.png')
plt.show()
```

附件 C：贝叶斯数据融合

dispose.py

```
import numpy as np
import pandas as pd

# # 显示所有行和列
# pd.set_option('display.max_columns', None)
# pd.set_option('display.max_rows', None)

# 导入数据
```

```
df = pd.read_excel('C 题数据.xlsx', sheet_name='data3')

df_empty = pd.DataFrame()

# 处理相同索引的数据
for i in np.arange(1, 75, 3):
    for j in np.arange(0, 8):
        temp = df.iloc[i - 1:i + 2, j]

        # 贝叶斯数据融合算法
        miu_0 = np.mean(temp)
        sigma_0 = np.std(temp)
        if sigma_0 != 0:
            miu = np.array([np.mean([temp.iloc[0], temp.iloc[1]]), np.mean([
temp.iloc[0], temp.iloc[2]]),
                            np.mean([temp.iloc[1], temp.iloc[2]])])
            sigma = np.array([np.std([temp.iloc[0], temp.iloc[1]]), np.std([
temp.iloc[0], temp.iloc[2]]),
                              np.std([temp.iloc[1], temp.iloc[2]])])
            if 0 not in sigma:
                sum_data = np.sum(miu / np.power(sigma, 2))
                data = miu_0 / np.power(sigma_0, 2)
                sum_sigma = np.sum(1 / np.power(sigma, 2))
                result = (sum_data + data) / (sum_sigma + 1 / np.power(sigma
_0, 2))
                df_empty.loc[(i + 2) / 3, j] = result
            else:
                np.sort(sigma)
                sum_data = np.sum(miu[0:2] / np.power(sigma[0:2], 2))
                data = miu_0 / np.power(sigma_0, 2)
                sum_sigma = np.sum(1 / np.power(sigma[0:2], 2))
                result = (sum_data + data) / (sum_sigma + 1 / np.power(sigma
_0, 2))
                df_empty.loc[(i + 2) / 3, j] = result
        else:
            df_empty.loc[(i + 2) / 3, j] = miu_0
        # 测试数据
        # print(f'[{(i-1)/3},{j}]:{result}')
col = ['接收距离(cm)',
       '热风速度(r/min)',
       '厚度 mm',
       '孔隙率（%）',
       '压缩回弹性率（%）',
       '过滤阻力 Pa',
```

```
        '过滤效率（%）',
        '透气性 mm/s']
df_empty.columns = col

# 保存数据
df_empty.to_excel('data3(副本).xlsx')
print('Done!')

# 测试
# print(df)
# print(df_empty)
# print(df.iloc[:, 3:])
```

附录 D：二维三阶样条插值

third_order_spline.m

```
% 三阶样条进行拟合回归
% x 为接收距离，y 为热风速度
% z0，z1，z2 分别为厚度、孔隙率、压缩回弹率

% 生成 xy 轴网格
[x,y]=meshgrid(20:5:40,800:100:1200);

% 辅助打表用的真实数据
x0=[40,40,40,40,40,35,35,35,35,35,30,30,30,30,30,25,25,25,25,25,20,2,20,20,2
0];
y0=[800,900,1000,1100,1200,800,900,1000,1100,1200,800,900,1000,1100,1200,800
,900,1000,1100,1200,800,900,1000,1100,1200];

% 打表生成 z0,z1,z2
% 修改完成的 z 坐标
z0=[1.699389632,1.874583949,2.116319634,2.400991161,2.756112284;
    1.983640319,2.158243889,2.391072477,2.691339166,3.051307177;
    2.183017188,2.372667633,2.619843517,2.922552932,3.282372522;
    2.328540663,2.515279317,2.777513519,3.068409312,3.420284435;
    2.402799728,2.607630724,2.86202991,3.159244728,3.513136799];
z1=[93.90525022,94.16622905,94.97505054,96.03321258,95.95201707;
    94.70376317,95.24836,95.68352106,96.55658245,96.93359982;
    95.44995731,95.23137408,96.37553658,96.53492249,96.72605359;
    95.31148823,96.36043,96.40546742,96.37943623,96.8242702;
    95.93017396,96.0327448,96.57059445,96.80461344,96.53716842];
```

```
z2=[86.19373792,87.64058712,87.54937632,86.53177116,84.75469899;
    86.85307739,87.74454678,88.18484327,87.27670901,85.3137326;
    86.71074147,88.47038704,88.32022052,87.37556785,85.53550721;
    86.2271887,87.50980909,87.36800617,87.11788081,85.08414223;
    85.13900713,85.98861674,86.16977751,85.68193627,83.57417539];

% x0,y0 为待预测的数对
xt=[38,33,28,23,38,33,28,23];
yt=[850,950,1150,1250,1250,1150,950,850];

% 创建预测网格并进行三阶样条预测
[xi,yi]=meshgrid(20:1:40,800:10:1250);
zt0=interp2(x,y,z0,xi,yi,"spline");
zt1=interp2(x,y,z1,xi,yi,"spline");
zt2=interp2(x,y,z2,xi,yi,"spline");

% 厚度填表
tmp0=[];
tmp0=[tmp0,zt0(5,18)];
tmp0=[tmp0,zt0(15,13)];
tmp0=[tmp0,zt0(35,8)];
tmp0=[tmp0,zt0(45,3)];
tmp0=[tmp0,zt0(45,18)];
tmp0=[tmp0,zt0(35,13)];
tmp0=[tmp0,zt0(15,8)];
tmp0=[tmp0,zt0(5,3)];

% 孔隙率填表
tmp1=[];
tmp1=[tmp1,zt1(5,18)];
tmp1=[tmp1,zt1(15,13)];
tmp1=[tmp1,zt1(35,8)];
tmp1=[tmp1,zt1(45,3)];
tmp1=[tmp1,zt1(45,18)];
tmp1=[tmp1,zt1(35,13)];
tmp1=[tmp1,zt1(15,8)];
tmp1=[tmp1,zt1(5,3)];

% 压缩回弹性率填表
tmp2=[];
tmp2=[tmp2,zt2(5,18)];
tmp2=[tmp2,zt2(15,13)];
tmp2=[tmp2,zt2(35,8)];
tmp2=[tmp2,zt2(45,3)];
```

```
tmp2=[tmp2,zt2(45,18)];
tmp2=[tmp2,zt2(35,13)];
tmp2=[tmp2,zt2(15,8)];
tmp2=[tmp2,zt2(5,3)];

% 绘图
% 厚度
subplot(1,3,1)
surf(xi,yi,zt0)
title('厚度与工艺参数的关系')
xlabel('接收距离')
ylabel('热风速度')
zlabel('厚度')
colormap(jet)  %颜色的风格选择
colorbar('SouthOutside')
axis vis3d
% 孔隙率
subplot(1,3,2)
surf(xi,yi,zt1)
title('孔隙率与工艺参数的关系')
xlabel('接收距离')
ylabel('热风速度')
zlabel('孔隙率')
colormap(jet)  %颜色的风格选择
colorbar('SouthOutside')
axis vis3d
% 压缩回弹率
subplot(1,3,3)
surf(xi,yi,zt2)
title('压缩回弹性率与工艺参数的关系')
xlabel('接收距离')
ylabel('热风速度')
zlabel('压缩回弹性率')
colormap(jet)  %颜色的风格选择
colorbar('SouthOutside')
axis vis3d
% 总标题
sgtitle('工艺变量与结构变量的关系')
```

附录 E：结构变量与产品性能关系（利用主成分）

pca.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
# 插值库
from scipy import interpolate
import seaborn as sns

# 设置绘图风格
sns.set()

# 加载中文字体
font = FontProperties(fname="./data/SimHei.ttf", size=14)
plt.rcParams["font.sans-serif"] = ["SimHei"]  # 设置字体
plt.rcParams["axes.unicode_minus"] = False  # 该语句解决图像中的“-”负号的乱码问题

# 读取数据
df = pd.read_excel('data3(副本).xlsx', index_col=0)

# 通过得到的公式计算结构变量与产品性能的主成分
# 结构变量与产品性能之间的关系可以直接使用主成分中的数据进行解释
pca1 = df['厚度 mm'] * 0.437 + df['孔隙率（%）'] * 0.41 - df['压缩回弹性率（%）'] * 0.305
pca2 = df['过滤阻力 Pa'] * 0.312 + df['过滤效率（%）'] * 0.442 - df['透气性 mm/s'] * 0.44

# 计算标准数据
# 均值与方差
mean_data = pca2.mean()
std_data = pca2.std()
# 测试
# print(mean_data, std_data)

# 在这里我们只选用[μ-σ,μ+σ]中的值
# 筛除异常值并保存到一个DataFrame 中
newdf = pd.DataFrame({'结构变量': pca1, '产品性能': pca2})
newdf = newdf.loc[(newdf['产品性能'] < mean_data + std_data)
                  &
                  (newdf['产品性能'] > mean_data - std_data)]
```

```
newdf = newdf
newdf.sort_values(by='结构变量', inplace=True, ascending=True)

# 测试
print(newdf)

# 绘图
# 结构变量与产品性能之间的关系
# 绘制散点图
# newdf.plot(x='结构变量', y='产品性能')
plt.scatter(x=newdf['结构变量'], y=newdf['产品性能'],
            c='orange', s=50, alpha=0.5,
            linewidths=5, norm=1, label='data')

# 多种插值方法进行比较(分别为最近邻，阶梯，线性，二阶样条)
f1 = interpolate.interp1d(x=newdf['结构变量'], y=newdf['产品性能
'], kind='nearest')
f2 = interpolate.interp1d(x=newdf['结构变量'], y=newdf['产品性能
'], kind='zero')
f3 = interpolate.interp1d(x=newdf['结构变量'], y=newdf['产品性能
'], kind='linear')
f4 = interpolate.interp1d(x=newdf['结构变量'], y=newdf['产品性能
'], kind='cubic')

# 设置新的x坐标
x = list(np.arange(12.7, 15.6, 0.1))
y1 = list(f1(x))
y2 = list(f2(x))
y3 = list(f3(x))
y4 = list(f4(x))

# 绘出多条插值曲线
plt.plot(x, y1, label='nearest')
plt.plot(x, y2, label='zero')
plt.plot(x, y3, label='linear')
plt.plot(x, y4, label='cubic')

plt.title('结构变量与产品性能的关系')
plt.xlabel('结构变量')
plt.ylabel('产品性能')
plt.legend(loc='lower left',
           ncol=2,
           title='插值方法',
           shadow=True,
```

```
            fancybox=True)
# 设置轴的上下限值
plt.xlim(12.0, 16.0)
plt.ylim(-200, -130)
plt.savefig('拟合结构变量与产品性能.png')
plt.show()
print('Done!')
```

附录 F：过滤效率与过滤阻力三维图

extreme_value.m 与 tmp.m

extreme_value.m

```
% 工艺参数与过滤效率关系
% 数据是贝叶斯数据融合得到的数据
% 三阶样条进行拟合回归
% x 为接收距离，y 为热风速度
% z 为过滤效率

% 生成 xy 轴网格
[x,y]=meshgrid(20:5:40,800:100:1200);
z = [51.99469402,52.16730075,52.90402058,52.24293926,49.65186213;
    52.32356268,52.76192642,51.94916283,47.06969671,48.04627695;
    72.94600032,51.13190165,50.03488218,45.46861006,40.01550411;
    80.74083592,49.51501412,47.50205654,42.36857135,42.69132151;
    83.1216874,49.42602694,49.17154633,42.58149384,44.25644457];

% 创建预测网格并进行三阶样条插值
[xi,yi]=meshgrid(20:1:40,800:10:1200);
zt=interp2(x,y,z,xi,yi,"spline");
% 绘图
surf(xi,yi,zt)

% 图像设置
title('工艺参数与过滤效率的关系')
xlabel('接收距离')
ylabel('热风速度')
zlabel('过滤效率')

colorbar  %显示颜色栏
colormap(jet)  %颜色的风格选择
```

```
axis vis3d

xlswrite('efficiency.xlsx',zt);
```

tmp.m

```
% 工艺参数与过滤效率关系
% 数据是贝叶斯数据融合得到的数据
% 三阶样条进行拟合回归
% x 为接收距离，y 为热风速度
% z 为过滤阻力

% 生成 xy 轴网格
[x,y]=meshgrid(20:5:40,800:100:1200);
z=[38.22296426,32.68928687,36.27205265,31.39278343,26.07147561;
   32.27805089,31.95073333,31.85018272,27.71005435,26.55880133;
   31.53920649,32.30914308,29.98453635,26.18132637,20.58498049;
   30.53049783,30.47512832,28.0177848,23.41447211,24.53537213;
   28.29051524,28.96417004,27.80688626,23.54595161,24.7579071];

% 创建预测网格并进行三阶样条插值
[xi,yi]=meshgrid(20:1:40,800:10:1200);
zt=interp2(x,y,z,xi,yi,"spline");
% 绘图
surf(xi,yi,zt)

% 图像设置
title('工艺参数与过滤效率的关系')
xlabel('接收距离')
ylabel('热风速度')
zlabel('过滤阻力')

colorbar  %显示颜色栏
colormap(jet)  %颜色的风格选择
axis vis3d

xlswrite('resistance.xlsx',zt);
```

附录 G：效率与阻力热力图与考量分析

efficiency_resistance.py

```python
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
from matplotlib import cm
import seaborn as sns

# 设置绘图风格
sns.set()

# 加载中文字体
font = FontProperties(fname="./data/SimHei.ttf", size=14)
plt.rcParams["font.sans-serif"] = ["SimHei"]  # 设置字体
plt.rcParams["axes.unicode_minus"] = False  # 该语句解决图像中的“-”负号的乱码问题

# 读取数据
df = pd.read_excel('data3(副本).xlsx', index_col=0)
tmp_efficiency = pd.read_excel('efficiency.xlsx', index_col=0)
tmp_resistance = pd.read_excel('resistance.xlsx', index_col=0)

# 测试
# sns.heatmap(efficiency)
# plt.show()
# sns.heatmap(resistance)
# plt.show()

# 绘制热力图
map_vir = cm.get_cmap('Reds')

sns.heatmap(tmp_efficiency, cmap=map_vir)
plt.yticks(rotation=45)
plt.xlabel('接收距离')
plt.ylabel('热风速度')
plt.title('过滤效率热力图'fontsize=20)
plt.savefig('过滤效率热力图.png')
plt.show()

sns.heatmap(tmp_resistance, cmap=map_vir)
plt.yticks(rotation=45)
plt.xlabel('接收距离')
```

```
plt.ylabel('热风速度')
plt.title('过滤阻力热力图'fontsize=20)
plt.savefig('过滤阻力热力图.png')
plt.show()

tmp_efficiency.index.set_names('热风速度', inplace=True)
tmp_efficiency.columns.set_names('接收距离', inplace=True)
tmp_resistance.index.set_names('热风速度', inplace=True)
tmp_resistance.columns.set_names('接收距离', inplace=True)
# print(tmp_efficiency)

print('效率', tmp_efficiency.iloc[15:31, 3:6])
print('阻力', tmp_resistance.iloc[15:31, 3:6])

# 测试
# print(df)
```