基于数据挖掘的古代玻璃分析模型

摘要

丝绸之路是中西文化交流的重要通道，其中玻璃的见证贸易往来的重要见证。古代玻璃的类型、亚类的检验、分辨，不仅有助于挖掘其背后的历史迁移信息；还可以通过分析风化前后玻璃的化学成分改变，探究出风化对于不同种类玻璃的影响规律。

首先针对本问题，进行数据的初步清洗工作，**剔除**掉累计和不在合理性范围85%~105%之间的无效数据样本。结合相关文献，根据玻璃化学成分对其颜色具有决定性影响作用的结论，针对表单 1 的颜色缺失值，选取表单 2 中普通风化的样本数据点，进行KNN**插值**填补颜色的缺失值。

针对问题一：首先通过**桑基图**进行数据可视化，初步判断玻璃的表面风化情况同玻璃类型、纹饰和颜色的影响，接着进行**卡方检验**，判断风化情况是否对其各项指标具有显著性影响。接着结合玻璃类型、风化情况将玻璃分为**风化铅钡玻璃、未风化铅钡玻璃、风化高钾玻璃、未风化高钾玻璃**四类。分别对每类的化学成分进行**描述性统计**，进行数据可视化，并根据数据的正态分布性，分别进行**独立样本**t**检验、独立样本**$MannWhitney$**检验**，衡量其风化前后化学成分变化的差异性。最后结合风化前后高钾、铅钡玻璃的统计学规律，针对每一化学成分，进行**分位点分箱操作**，对于每个类别统计风化前后的**均值之比**，将其作为预测的转化率。构造出充分考虑**严重风化等异常情况、特殊化学成分、风化前后统计量规律**等多重因素的**综合分箱均值预测模型**。最后根据预测结果，同原未风化的数据分布进行**差异性检验**。

针对问题二：首先根据**决策树模型**对铅钡、高钾玻璃大类进行分类，得到两类玻璃在**氧化铅**(PbO)特征上具有**线性可分性**，同时根据数据可视化探究了**其他化学成分**在玻璃类别中的差异性。通过查阅大量文献，可得一般进行玻璃成分检验时需要对玻璃进行**去风化层、研磨清洗**等处理，同时结合问题一得知玻璃风化后的某些化学成分数据分布会发生显著性改变，从而影响亚类划分。故**我们利用问题一的预测模型，对表单中风化样本进行预测，得到其风化前的成分数据**，并利用**方差滤波、异常处理、**PCA **降维**对数据进行清洗最终根据未风化的数据进行聚类分析，将铅钡玻璃分为 **3 个亚类**：高Na铅钡、高Al,Ca、高Cu亚类，将高钾玻璃分为 **4 个亚类**：中Al中Ca高Cu高Fe、中Al 中Ca高Cu低Fe、高Al高Ca低Cu、低Al 低Ca 亚类，最后进行灵敏分析。

针对问题三：根据利用$XGBoost$分类模型，结合问题二的亚类标签进行模型学习，并根据Shap **解释模型**筛选出对模型贡献度较高的特征，将这些特征进行波动，检测模型输出结果的稳定性。

针对问题四：首先构建**关联规则挖掘**模型，依据**等距分箱**思想结合化学成分的含量比值对每个化学成分等距划分为 5 类，从而生成标识化学成分及其含量的项集集合，并通过$FP-Growth$**算法**进行**关联规则挖掘**，分析不同类别是否风化的化学成分的关联性以及不同类别化学成分**关联关系的差异性**。由于不同化学成分之间虽然含量分布差别较大，但其有可能呈现一定的序列相关，于是构建**灰色关联度分析**模型，计算**自关联系数矩阵**，定量分析各变量之间的相关性以及不同类别化学成分关联关系的差异性。

关键词：分箱均值预测、亚类划分、灵敏度分析、关联挖掘

一、 问题重述

1.1 背景分析

丝绸之路是连接中外文化的桥梁，其中出土的古代玻璃也是中外文化、技术交流的见证。但由于年代久远，玻璃类别的判别具有一定困难。

对古代玻璃的外观、颜色、纹饰，以及风化、未风化的化学成分进行数据挖掘，有助于探索玻璃的发展历史、发展起源地等重要历史信息，对研究中亚、东亚的玻璃文化具有非常重要的意义。

同时对于玻璃亚类的划分，也可以帮助推断古代玻璃的制作工艺，从而对古代玻璃的起源地进行溯源。对于玻璃风化前后化学成分变化的探究，也有助于进一步改善现代玻璃的防风化措施，有助于玻璃质量的维护工作。

因此对于对本题构建合理的模型，不仅具有重要的历史意义、历史价值，还对现代玻璃工业的制造、防风化措施、风化规律研究都有着非常重要的意义。

1.2 问题重述

针对问题一，首先通过玻璃的表面风化情况，可以探究其与纹饰、颜色、类型等各项指标的关系。接着需要结合玻璃类型，分别探究其风化前后对各类化学成分指标的影响。对这种影响的差异性，一方面进行参数检验，另一方面结合可视化的数据分布，进行数据挖掘操作。最后，需要根据风化前后的样本的化学成分差异规律，由风化后的样本成分指标，构建预测模型，预测其风化前的成分指标。

针对问题二，首先要建立合理的分类模型，寻找出分类铅钡玻璃、高钾玻璃的分类规律。并且根据特征选择，选出合适的指标对铅钡玻璃、高钾玻璃进一步进行亚类分类，并通过波动化学成分来检测聚类模型的灵敏性，通过结合相关的文献说明聚类模型的合理性。

针对问题三，由第二问的亚类分类标签，构造分类模型对表单 3 的中的未知样本进行亚类预测，同时需要通过选择贡献程度大的特征，进行波动，并以此检验模型的灵敏度。

针对问题四，需要探索出不同类别的玻璃类型的各个化学成分指标之间的关联程度，可以结合定量和定性分析，对这种关联程度进行综合分析说明。最后结合不同类别，说明化学成分之间的关联程度在类别间的差异性。

二、 问题分析

针对该问题，本文从多方面进行数据分析、特殊值检验，构造出合理的预测、分类模型，并对模型的合理性、灵敏性进行了检验，对于问题的具体分析如下

2.1 问题一分析

针对问题一第一小问，首先根据样本分布情况，使用KNN**插值**对颜色缺失值进行填补。利用**桑基图**对数据分布进行初步分析。针对玻璃风化情况，分别与玻璃类型、纹饰和颜色进行**卡方检验**，判断风化情况是否对其具有显著性影响。

针对问题一第二小问，首先结合玻璃类型、风化情况将玻璃分为**风化铅钡玻璃、未风化铅钡玻璃、风化高钾玻璃、未风化高钾玻璃**四类。分别对每类的化学成分进行描述性统计，并根据数据分布判断铅钡玻璃、高钾玻璃风化前后化学成分含量变化情况，最后根据数据的正态分布性，分别进行独立样本t检验、独立样本$MannWhitney$检验，衡量其风化前后化学成分变化的差异性，根据数据分布可发现高钾玻璃中化学成分易受风化影响。

针对问题一第三小问，首先结合风化前后高钾、铅钡玻璃的统计学规律，针对每一化学成分，进行**分位点分箱操作**，对于每个类别统计风化前后的**均值之比**，将其作为预测的转化率。由于本题数据较少，且高钾玻璃的氧化纳、氧化钾成分样本分布偏移较严重，故预测时考虑加入高斯随机量进行**随机采样预测**。同时本预测模型综合考虑**严重风化等异常情况、特殊化学成分**等多重因素。最后根据预测结果，同原未风化的数据分布进行**差异性检验**，检验结果非常合理。

2.2 问题二分析

针对问题二第一小问，首先根据**决策树模型**对铅钡、高钾玻璃大类进行分类，得到两类玻璃在**氧化铅**(PbO)特征上具有**线性可分性**，同时根据数据可视化探究了其他化学成分在玻璃类别中的差异性。

针对问题二第二小问，首先查阅大量文献，可得一般进行玻璃成分检验时需要对玻璃进行**去风化层、研磨清洗**等处理，且考虑到风化后同一类别的玻璃也会发生化学成分变化，从而影响聚类效果和合理性。故首先将风化后的数据预测为风化前的数据，同原数据中未风化的数据进行聚类分析。利用**方差滤波、异常处理、**PCA **降维**对数据进行清洗，然后根据提取的主成分进行聚类分析，最后根据**雷达图和相关文献**，结合化学成分含量，对亚类进行合理划分。

针对问题二第三小问，对样本特征数据进行波动，检查聚类结果是否合理。

2.3 问题三分析

针对问题三第一小问，首先同问题二进行数据清洗，构建$XGBoost$分类模型，针对铅钡玻璃、高钾玻璃分别进行分类模型构建。对表单 3 中的数据，首先根据问题二中，铅钡玻璃、高钾玻璃在**氧化铅**(PbO)特征上具有**线性可分性**，初步将样本分类。结合问题一预测模型，将表单 3 中风化点成分转化成未风化的成分，利用构建的分类模型进行类别划分。

针对问题三第二小问，首先根据Shap **解释模型**，筛选出重要特征，波动重要特征信息，观察模型输出结果是否有显著性变化。

2.4 问题四分析

针对问题四第一小问，首先根据关联规则挖掘，分**风化铅钡玻璃、未风化铅钡玻璃、风化高钾玻璃、未风化高钾玻璃**四类进行化学成分关联的定性分析，然后利用**灰色关联分析**定量分析。

针对问题四第二小问，根据第一小问不同类型的化学成分的关联性，进行差异性定性分析。

三、 符号说明

符号	含义
r_{ij}	第i个化学成分第j部分的风化前后转化率
a_{pi}	未风化成分的第p个样本的第i个化学成分含量
a'_{pi}	未风化成分预测的第p个样本的第i个化学成分含量
A'_{ij}	样本数据a'_{pi}属于四分位分箱结果的第j类

四、 模型假设

1. 假设表单 2 中风化点、未风化点化学成分数据，除特殊指明，同该样本其他部位的成分之间无显著差异，可近似作为该玻璃的指标。
2. 假设采样数据均来自现实，符合古代玻璃制作工艺要求、同参考文献和具体实验的结果相匹配。
3. 假设表单 2 中数据未检测到的成分可认为时微小量，其含量近似作为 0 处理。
4. 假设风化前后成分的变化具有一定的对应关系，即风化前含量较高的成分，风化后在其样本分布中仍处于较高水平。

五、 问题一模型的建立与求解

5.1 问题一的求解与分析

针对问题一，我们首先选取普通风化点的数据信息，对附件一中的样本颜色缺失值进行KNN插值，根据插值后的数据，针对玻璃表面风化情况同玻璃类型、纹饰、颜色进行数据可视化，并单独进行卡方检验，验证风化对各类指标影响是否有显著性。

接着，结合玻璃类型，将玻璃划分为风化铅钡玻璃、未风化铅钡玻璃、风化高钾玻璃、未风化高钾玻璃，对其各项化学成分进行描述性统计，观察其各项数据特征。并针对化学成分的分布情况可视化处理，初步判断高钾玻璃、铅钡玻璃风化前后化学成的变化情况，最后针对不同类别玻璃风化前后的成分进行差异性检验。

最后根据第二小问的类别划分，分别计算铅钡玻璃、高钾玻璃每个化学成分对应的风化前后转化率。考虑到成分浓度的差异、在样本分布中位置对风化前后转化率的影响，我们针对风化前后的化学成分进行四分位操作，计算利用每个分箱的均值之比作为风化预测的转化率。最终根据风化前后的化学成分的差异性检验预测模型的合理性。

其具体流程如下

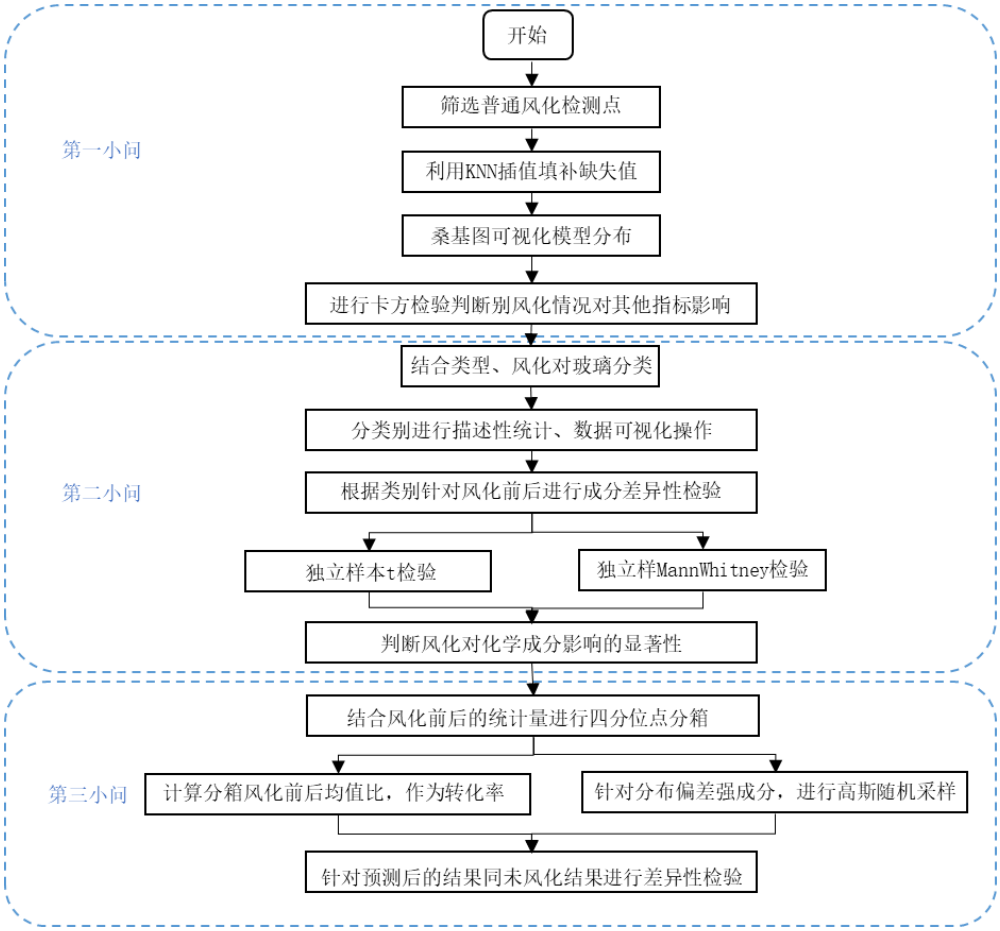


图 5-1 问题一流程图

5.2 问题一的模型构建

5.2.1 风化指标影响分析

1．数据清洗

首先根据题目要求剔除累加和不在 85%~105%之间的数据，由此剔除编号为 15,17 的样本数据。同时观察表单 1 内容，发现样本编号为 19,40,48,58 的样本存在颜色缺失。通过查阅文献可得，玻璃颜色受各类化学成分影响，故利用此规律和表单 2 中的化学成分，提取正常风化的数据点，利用基于聚类的KNN插值算法，根据化学成分将存在缺失值的样本点预测为与其化学成分类似的样本的颜色，进行缺失值的填充。

其中针对表单 2 中的数据，剔除了未风化点数据、严重风化点数据，并针对具有两部分的正常风化点数据取均值处理，最终选取到 23 条已知颜色的样本数据点。其编号如下

表 5-1 插值样本点编号

插值样本点编号
2,7,8,9,10,11,12,22,26,27,34,36,38,39,41,43,49,50,51,52,54,56,57

其中KNN插值算法的具体流程如下

表 5-2 *KNN* 插值算法流程

***KNN* 插值算法**
Step1 将样本的化学成分进行标准化处理，计算已知颜色的样本点的化学成分与当前点的距离
Step2 按照距离递增排序，选取与待插值点的距离最小的k个点
Step3 确定k个点所属类别出现的概率，返回k个类中类别中出现频率最高的类别，将缺失值的颜色特征填充为该类别对应的颜色

最终得到的填充情况如下

表 5-3 缺失值插值结果

编号	19	40	48	58
颜色	深蓝	浅蓝	浅蓝	深蓝

2．数据可视化

根据处理后的数据进行数据探索，得到风化情况同玻璃类型、纹饰和颜色的关系分布图，其具体图如下所示。

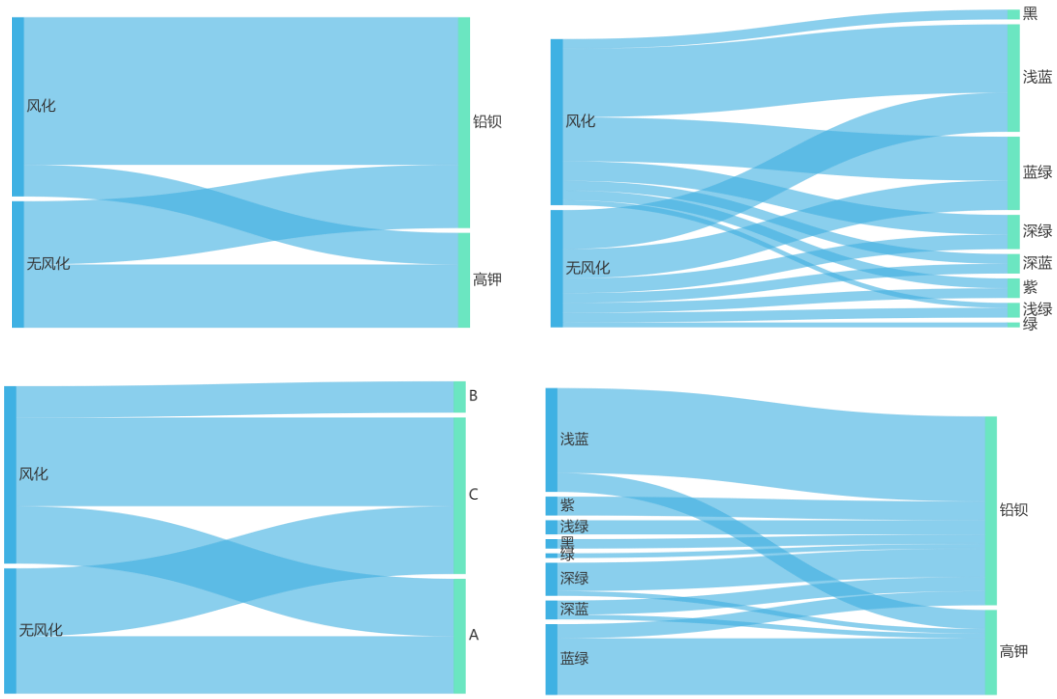


图 5-2 变量桑基图

针对桑基图可以初步看出，风化情况与类别的关联中，铅钡玻璃中风化的占比程度较大，高钾玻璃中无风化的占比程度较大；与颜色的关联中，黑色玻璃均为风化，绿色玻璃均为无风化情况，其他颜色中风化与未风化影响并不明显；在与纹饰的关联中，B类型的纹饰均为风化，其他纹饰中风化与未风化影响并不明显。经查阅文献知，Na元素对钾玻璃的颜色具有一定影响，通过进一步挖掘数据发现，颜色同玻璃种类之间存在一定的关联性，即在样本中紫色、浅绿、黑色、绿色均为铅钡玻璃，同时浅蓝

色玻璃大部分为铅钡玻璃，蓝绿色玻璃大部分为高钾玻璃。为进一步探索风化情况对于玻璃类型、纹饰和颜色的关系，进行卡方检验。

3．风化指标对其他类别影响显著性分析

针对表面风化情况，分别对玻璃纹饰、颜色和类型进行卡方检验，同时进一步探索颜色同玻璃类型之间的差异性关系，得到如下图表。

表 5-4 卡方检验结果

	名称	表面风化		P
		无风化	风化	
纹饰	A	11	11	0.084*
	B	0	6	
	C	13	17	
颜色	浅绿	2	1	0.767
	浅蓝	8	14	
	深绿	3	4	
	深蓝	2	2	
	紫	2	2	
	绿	1	0	
	蓝绿	6	9	
	黑	0	2	
类型	铅钡	12	28	0.009***
	高钾	12	6	

通过卡方检验可得，如下结论。

1. 有无风化对于纹饰而言$P>0.05$，5%水平上不呈现显著性，因此对于纹饰而言，风化情况对其影响不成显著性差异，但通过桑基图可观察到B类型的纹饰均为风化。
2. 有无风化对于颜色而言$P>0.05$，5%水平上不呈现显著性，因此对于颜色而言，风化情况对其影响不成显著性差异，但通过桑基图可观察到黑色玻璃均为风化，绿色玻璃均为无风化情况
3. 有无风化对于类型而言$P<0.01$，1%水平上呈现显著性，因此对于类型而言，风化情况对其影响成显著性差异。可以在桑基图中铅钡玻璃、高钾玻璃中的风化、未风化的分布情况得到验证。

同时，经过卡方检验可以得到类型对颜色的P值为 0.001***，显著性非常明显，同文献中某些元素对玻璃颜色具有显著影响的结论相符。

5.2.2 表面风化、玻璃类型与各类指标的关系

1. 描述性统计

首先根据题目，结合表面风化情况、玻璃类型将玻璃分为**风化铅钡玻璃、未风化铅钡玻璃、风化高钾玻璃、未风化高钾玻璃**四类，针对四类玻璃，对各类化学成分进

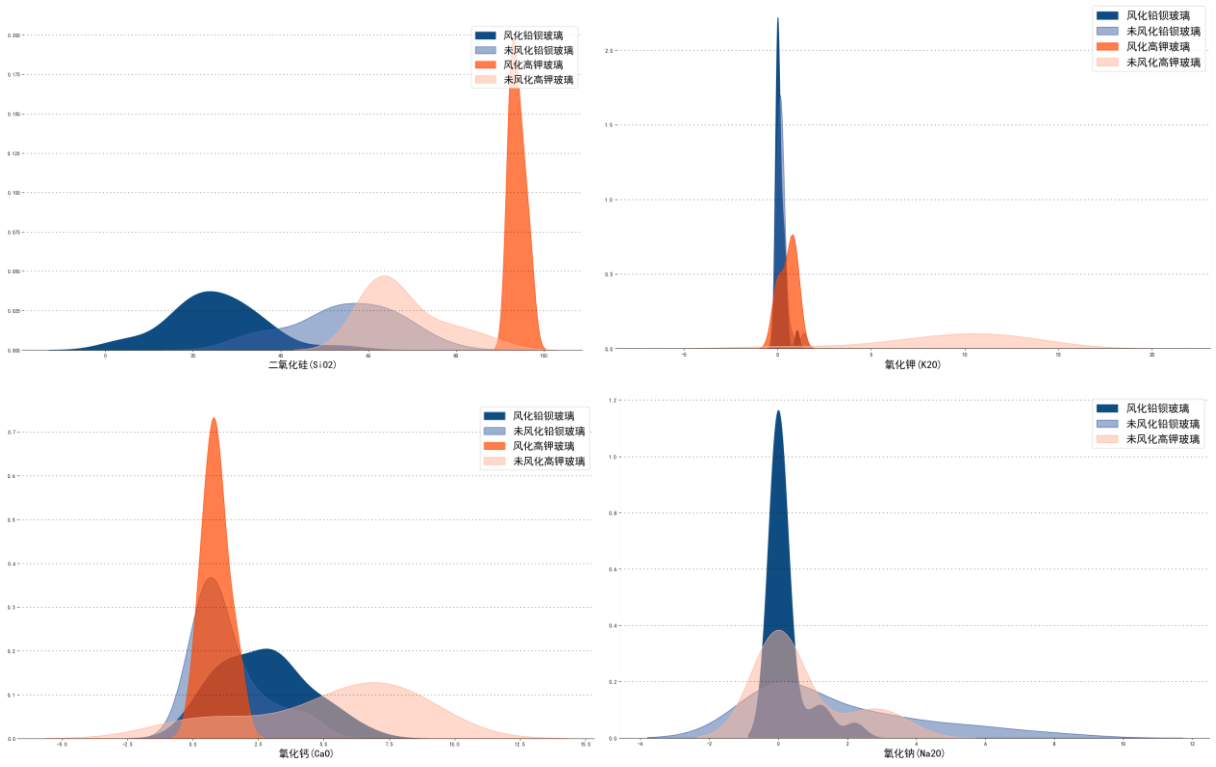
行数据特征的描述性统计，得到如下结果，由于篇幅有限只列出部分铅钡玻璃风化前后的描述性统计结果，全部结果见于附录。

表 5-5 描述性统计结果

化学成分	表面风化	平均值	标准差	中位数	方差	峰度	偏度	变异系数
二氧化硅 (SiO_2)	未风化	54.66	11.829	54.61	139.916	-0.538	-0.371	0.216
	风化	24.913	10.605	25.015	112.476	1.23	0.313	0.426
氧化钠 (Na_2O)	未风化	1.683	2.372	0	5.625	0.758	1.287	1.409
	风化	0.216	0.557	0	0.31	6.743	2.666	2.575
氧化钙 (CaO)	未风化	1.32	1.285	0.84	1.65	1.03	1.347	0.973
	风化	2.695	1.66	2.875	2.755	-0.461	0.383	0.616
氧化镁 (MgO)	未风化	0.64	0.547	0.71	0.299	-1.241	0.085	0.854
	风化	0.65	0.706	0.57	0.499	1.209	1.037	1.087
氧化铝 (Al_2O_3)	未风化	4.456	3.262	3.86	10.644	4.233	1.988	0.732
	风化	2.97	2.634	2.38	6.939	10.638	2.829	0.887
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
氧化锡 (SnO_2)	未风化	0.047	0.127	0	0.016	5.816	2.634	2.737
	风化	0.068	0.269	0	0.073	19.787	4.365	3.936
二氧化硫 (SO_2)	未风化	0.159	0.763	0	0.582	23	4.796	4.796
	风化	1.366	4.206	0	17.691	9.634	3.261	3.079

2. 数据分布可视化

针对风化前后的各类化学成分得到如下数据分布图，由于篇幅限制全部分布图见附录。



根据数据可视化结果，可以初步得到如下部分结论。

1. 对于铅钡玻璃，SiO_2在风化后其样本分布向左偏移，呈下降趋势；对于高钾玻璃，SiO_2在风化后其样本分布向右偏移，且集中于较高含量。
2. 对于铅钡玻璃，K_2O在风化前后的分布较为接近，影响并不明显；对于高钾玻璃，K_2O在风化后流失，向左偏移，且集中于较低含量。
3. 对于铅钡玻璃，CaO在风化后其样本分布发生改变逐渐向分布曲线右下角偏移；对于高钾玻璃，CaO在风化后其样本分布向左上偏移，且集中于较低含量。
4. 对于铅钡玻璃，Na_2O在风化后其样本分布中间方向聚拢；对于高钾玻璃，Na_2O在风化后几乎没有分布，呈现较大影响。

针对以上结论和其他成分风化前后的数据分布偏移情况，可以初步推断对于风化前后化学成分的影响中，**高钾玻璃化学成分分布受到影响较大**。通过查阅文献可得，钾硅酸盐玻璃在风化后SiO_2会出现明显富集。且会导致一定程度K元素流失，同模型结论相符合。

3. 差异性检验

为进一步检查风化前后对各类化学成分的影响，结合玻璃类别，对风化前后的各类化学成分进行差异性检验，得到如下结果。

首先对各类化学成分进行正态分布性检验，由于样本量较少，故采用$S-W$正态分布性检验，对通过正态分布性检验的成分，根据玻璃类别，针对风化前后的数据，进行独立样本t检验；对于未通过正态分布性检验的成分，进行独立$Mann Whitney$检验。其中检验结果如下图所示，由于篇幅限制只列出铅钡玻璃检验结果，全部结果见附录。

表 5-6-1 铅钡玻璃风化前后成分分布正态性检验

元素成分	$S-W$检验	元素成分	$S-W$检验
SiO_2	0.102	CuO	0.000***
Na_2O	0.000***	PbO	0.116
K_2O	0.000***	BaO	0.000***
CaO	0.003***	P_2O_5	0.000***
MgO	0.000***	SrO	0.008***
Al_2O_3	0.000***	SnO_2	0.000***
Fe_2O_3	0.000***	SO_2	0.000***

表 5-6-2 铅钡玻璃风化前后成分独立样本t检验

元素成分	P	$Conhen'd$值
SiO_2	0.000***	2.657
PbO	0.000***	2.104

表 5-6-3 铅钡玻璃风化前后成分独立样本 $MannWhitney$ 检验

元素成分	P	$Conhen'd$ 值	元素成分	P	$Conhen'd$ 值
Na_2O	0.011**	0.877	CuO	0.052*	0.343
K_2O	0.143	0.31	BaO	0.521	0.338
CaO	0.002***	0.919	P_2O_5	0.000***	1.277
MgO	0.853	0.015	SrO	0.031**	0.589
Al_2O_3	0.024**	0.505	SnO_2	0.620	0.102
Fe_2O_3	0.745	0.159	SO_2	0.208	0.388

可以得到对于铅钡玻璃，风化前后化学成分 $SiO_2,PbO,Na_2O,CaO,Al_2O_3,CuO,P_2O_5,SrO$ 变化具有显著性差异，化学成分 $K_2O,MgO,Fe_2O_3,BaO,SnO_2,SO_2$ 在风化前后成分的分布没有显著性差异，同上述数据分布可视化的初步结论高度吻合。

经过同样差异性检验，可以得到对于高钾玻璃而言，CuO,Al_2O_3 成分通过 $S-W$ 正态分布性检验，经过独立样本 t 检验后得到的结果我饿，CuO 在风化前后样本分布具有没有显著性差异，Al_2O_3 具有显著性差异；对未通过正态性检验的化学成分进行独立样本 $MannWhitney$ 检验后得到的具有显著性差异的指标具有 $K_2O,CaO,MgO,SiO_2,Fe_2O_3,PbO,P_2O_5,SrO$，风化前后分布没有显著性差异的指标有 Na_2O,BaO,SnO_2,SO_2，结合第二步可视化结果，具有很强合理性。

其中具体含量的增加、偏移趋势可以通过第二步的数据可视化分布图轻松得出，且为节省论文篇幅，将其均放置在附录中，不再赘述。

5.2.3 风化前后成分预测

1. 预测模型构建

针对该预测问题，考虑到数据样本点太少，且具有直接对照意义的样本点较少、风化前后化学成分分布差异较大等客观因素，只能初步探索出未风化、风化的化学成分的边际分布规律，其联合分布情况较难得知，故使用风化前后联合分布预测化学成分在本题中不太符合。故我们采取直接挖掘数据特征的方式来进行成分预测。

考虑到均值能衡量样本的数据综合情况，故初步考虑用风化前后的成分均值之比作为风化前后的成分转移率。但考虑到风化成分前后的变化具有非常明显的映射意义，即风化前成分高的点，经过风化后其在风化后的成分分布中，大概率能保持较高水平的含量。

综合考虑上述问题，我们采用分箱均值预测的方法。即首先根据玻璃类别进行分类，对未风化和风化的每个化学成分进行四分位点分箱处理，即每个化学成分含量被分为四部分，针对这四部分分别取含量的均值 $\mu_{ij}(i=1,2,\cdots,14;\ j=1,2,3,4)$ 以及 $\mu'_{ij}(i=1,2,\cdots,14;\ j=1,2,3,4)$。$\mu_{ij}$ 代表了风化前第 i 个化学成分第 j 部分的均值，μ'_{ij} 代表风化后第 i 个化学成分第 j 部分的均值。将两均值之比作为风化前后的转化率。

即构建如下的模型

$$r_{ij}=\frac{\mu_{ij}}{\mu'_{ij}}$$
$$a_{pi}=a'_{pi}\cdot r_{ij}\quad if\ a'_{pi}\in\{A'_{ij}\} \tag{1}$$

其中r_{ij}代表第i个化学成分第j部分的转化率，a_{pi}代表从风化后数据a'_{pi}预测的结果，a'_{pi}代表需要进行未风化成分预测的第p个样本的第i个化学成分含量，A'_{ij}代表样本数据a'_{pi}属于四分位分箱结果的第j类。

同时考虑到上一小问的显著性，故对于风化前后具有显著性差异的指标，我们进行预测，对没有显著性差异的保持原值。

2. 异常情况考虑

通过进一步探索数据，发现如下异常情况，需要对以分箱均值预测模型为基础的预测模型进行进一步的异常值处理，其中异常和处理情况如下。

1. 对严重风化的点预测时，通过表单 2 内容发现，对于每个严重风化点均有其对应的正常风化点，故其预测结果只取正常风化点的成分预测。
2. 对于高钾玻璃而言，其Na_2O和MgO其分布存在较大样本偏移，如对于未风化的高钾玻璃而言，其Na_2O出现两端聚集情况，即数据分布集中于 0 和 3 附近，中间缺少连续性数据，故利用基于分布的分箱预测模型并不能较好进行预测，此时采用高斯随机采样预测，即从未风化点的该化学成分含量中随机取值，并为其加入高斯随机扰动量，具体预测规则如下。

$$a_{pi}=\mathrm{rand}(\{b_p\})+\varepsilon_p$$
$$\varepsilon_p\sim N\left(0,\frac{\sigma_p}{2}\right) \tag{2}$$

通过综合考虑统计量特征、化学成分分布规律、异常情况构建出如下所示的风化成分预测综合模型。

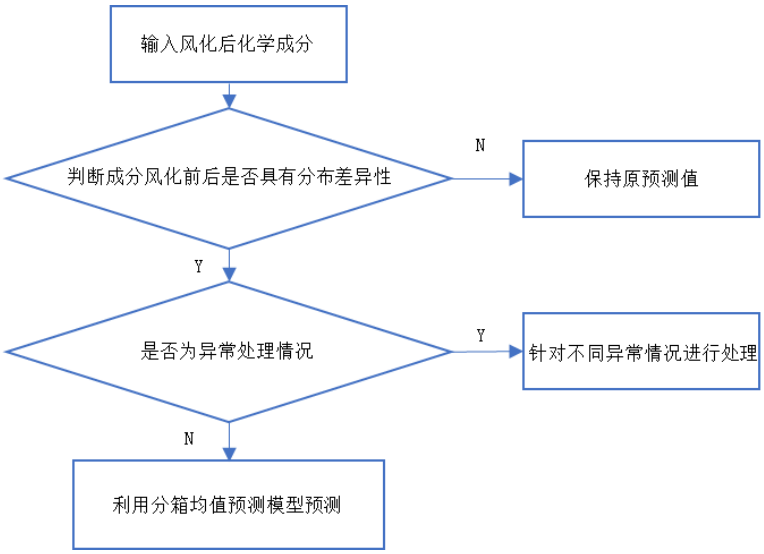


图 5-3 风化成分预测综合模型图

3. 模型合理性检验

为检验预测合理性，将预测后各项指标的分布同未风化的指标数据分布进行差异性检验，发现风化预测后的成分分布同未风化分布无显著性差异。具体检验结果如下所示，由于篇幅有限，这里只列出具有异常情况的高钾玻璃的差异性检验结果，铅钡检验结果见附录。

其中$SiO_2, Na_2O, CaO, P_2O_5, SrO, SnO_2$成分未通过正态性检验，其差异性检查结果果如下所示

表 5-7-1 高钾玻璃预测结果同未风化成分独立样本$MannWhitney$检验

元素成分	P	$Conhen'd$值	元素成分	P	$Conhen'd$值
SiO_2	0.892	0.096	P_2O_5	0.963	0.092
Na_2O	0.194	0.651	SrO	0.045**	1.038
CaO	0.851	0.08	SnO_2	0.480	0.348

其中除去SrO以外其余预测结果同原未风化数据分布均无显著性差异，说明预测模型非常合理。且SrO出现分布显著性差异的原因为，在风化后的高钾玻璃中SrO的成均为 0，无法获取分布的数据特征，因此无法达到较好的预测效果。

对其余通过正态性检验的成分指标$MgO, Al_2O_3, Fe_2O_3, CuO, K_2O$检测预测风化前的样本数据分布同原数据中的对应成分的分布差异，得到如下结果。

表 5-7-2 高钾玻璃预测结果同未风化成分独立样本t检验

元素成分	P	$Conhen'd$值	元素成分	P	$Conhen'd$值
MgO	0.738	0.170	CuO	0.989	0.007
Al_2O_3	0.944	0.036	K_2O	0.785	0.138
Fe_2O_3	0.981	0.012			

其中各项指标均无显著性差异，说明预测模型预测效果非常合理。

针对铅钡玻璃的差异性检验，可以得到除去Na_2O以外其他成分均无检测出明显差异性，而Na_2O出现明显差异性的原因一方面是由于未风化点中Na_2O的数据偏移较为严重，且分布呈两点聚集状，较难完成准确的风化前成分预测。

六、 问题二模型的建立与求解

6.1 问题二的求解与分析

针对问题二，首先利用决策树模型，对铅钡玻璃、高钾玻璃进行学习，通过结果发现，铅钡玻璃、高钾玻璃在氧化铅PbO的分布上是线性可分的，因此可以通过该指标作为区分玻璃类型的分类标准。同时，为进一步探究其他化学成分对分类的影响，我们可视化了其他化学成分在铅钡、高钾玻璃上的分布，得到其他化学成分可能影响玻璃分类的因素。

接着，构建亚类划分模型。通过查阅大量文献可得，由于风化会使得某些化学成分出现显著性差异，影响玻璃类别划分，故玻璃类别检测大多都基于未风化的化学成分进行分析，如一般会对表面风化玻璃进行去风化层、研磨、清洗等处理方法后，对其进行类别划分。所以我们利用问题一风化前的化学成分指标代替风化的数据，进行聚类分析，由第一问预测模型的合理性检验，可保证预测未风化数据的有效性。

进行聚类前，需要对异常点情况进行剔除，并利用方差滤波筛选出合适的化学成分，利用主成分分析法(PCA)对数据进行降维，利用提取的主成分进行$K-means$聚类。最终结合聚类雷达图、参考文献对亚类按照某些特定化学成分含量差异进行划分。

最后，利用蒙特卡洛方法，对样本的化学成分含量进行波动，重新聚类，进行聚类模型合理性的敏感性分析，最终得到较为合理的模型。

其具体流程如下

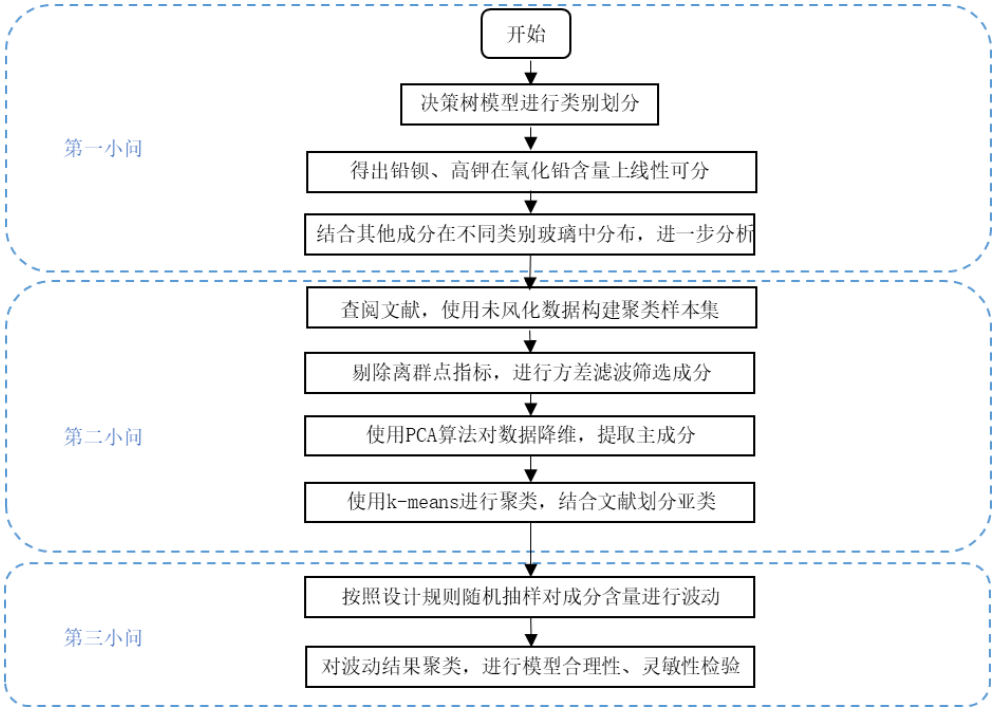


图 6-1 问题二流程图

6.2 问题二的模型构建

6.2.1 铅钡、高钾玻璃划分

1. 构建决策树模型对玻璃类别进行划分

针对铅钡玻璃、高钾玻璃，构建决策树分类器，对其进行类别划分，得到如下分类决策树。可以得到铅钡玻璃、高钾玻璃在PbO指标上是线性可分的，故可以将其直接作为划分铅钡玻璃、高钾玻璃大类的指标，但考虑到其他指标对于玻璃类别的影响，我们接下来通过可视化分布进一步说明其他指标对于玻璃类别划分的情况。

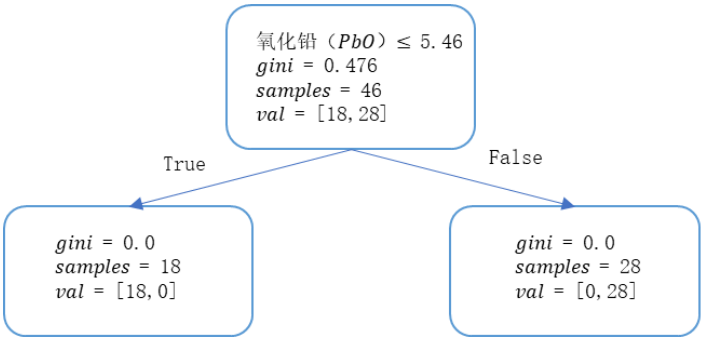


图 6-2 玻璃类别划分决策树

2. 探寻其他指标对玻璃划分的影响

通过观察两类玻璃类别的分布图，可以发现除去铅钡可以很好划分不同类别的玻璃外，K_2O, BaO, SiO_2 含量在两类玻璃数据分布也有较大差异。

1. 在高钾玻璃中 K_2O 含量分布相对较广，铅钡玻璃中 K_2O 相当少
2. 在高钾玻璃中 BaO 含量集中于 0 附近，铅钡玻璃中 BaO 分布较广
3. 在高钾玻璃中 SiO_2 的成分分布整体较铅钡玻璃分布大

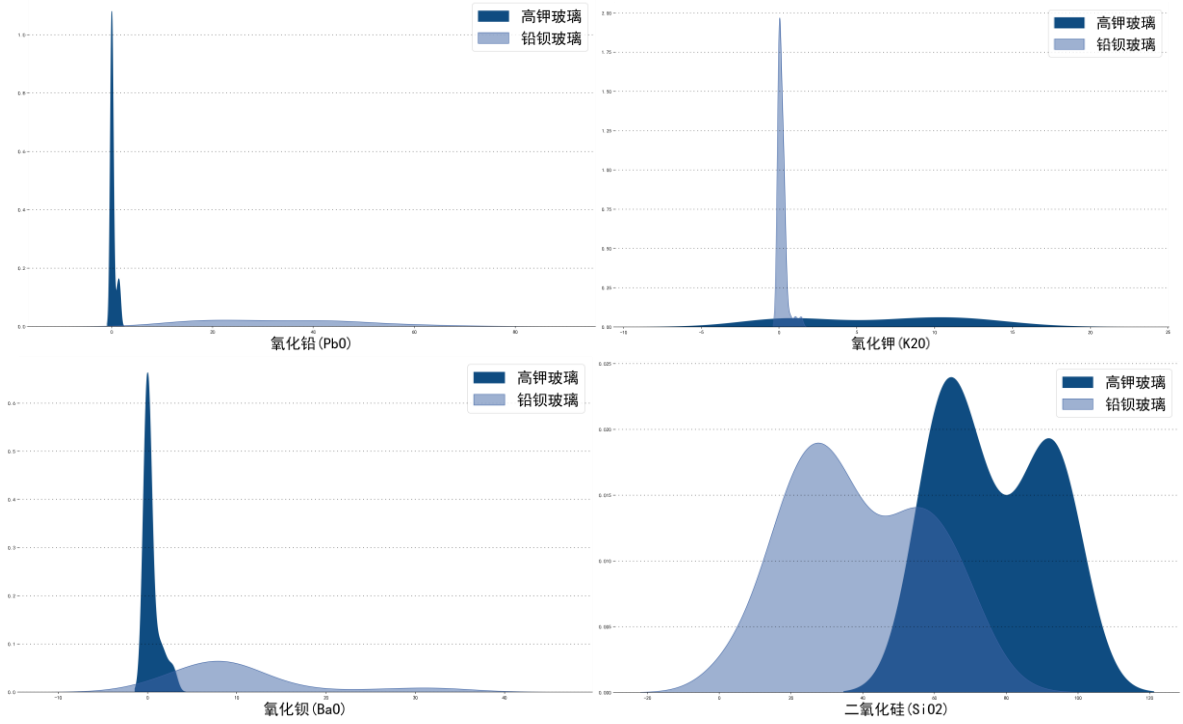


图 6-3 玻璃类别成分分布图

6.2.2 亚类的划分方法及结果

1. 特征筛选

经查阅资料发现，对于玻璃种类的划分，**大多采取去除风化层或取未风化区域进行分析**。结合本题所给数据，风化前后化合物含量差别很大，如果直接混合进行种类划分，是不合理的。因此我们采用第一问预测的风化前数据和未风化数据作为数据集，在保证合理性的基础上也体现模型设计的**关联性、综合性**。

由于数据样本较少，故需要对重要性指标进行筛选，否则考虑过多指标，会对模型亚类聚类分析造成一定影响。因此对于化学指标的选取利用**低通方差滤波、离群特征筛选**的方法，并结合文献资料的参考最终确定模型需要考虑的化学成分，具体考虑指标如下：

高钾玻璃考虑删除PbO,BaO指标，因为对高钾玻璃进行亚类划分是在高钾玻璃体系中。通过**离群特征筛选**删除SnO指标，并通过**低通方差滤波**删除SO_2,Na_2O,SrO，但是通过参考文献发现，Sr元素也能成为玻璃亚类划分指标。但由于本文的数据样本太少，且本文模型建立在含量为基础的亚类划分体系中，故不考虑Sr元素对亚类的划分影响。根据初步聚类效果发现，针对高钾玻璃而言，其各个类的聚类中心的K_2O,SiO_2无显著性差异，为进一步探索对高钾亚类的划分指标，考虑删除K_2O,SiO_2指标。

铅钡玻璃首先考虑删除K_2O指标，因为对铅钡玻璃进行亚类划分是在铅钡玻璃体系中，且通过查阅文献，也未发现K_2O在铅钡玻璃亚类划分中起的作用。通过**离群特征筛选**删除SnO指标，并通过**低通方差滤波**删除SO_2指标，最终得到的聚类特征如下所示。

表 6-1 高钾、铅钡玻璃聚类筛选指标

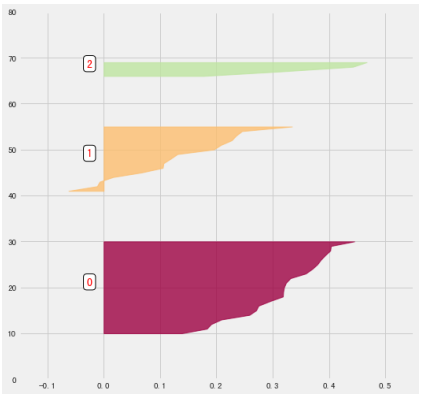
玻璃分类	化学成分
高钾玻璃	$CaO,MgO,Al_2O_3,Fe_2O_3,CuO,P_2O_5$
铅钡玻璃	$SiO_2,Na_2O,CaO,MgO,Al_2O_3,Fe_2O_3$ CuO,P_2O_5,SrO,PbO,BaO

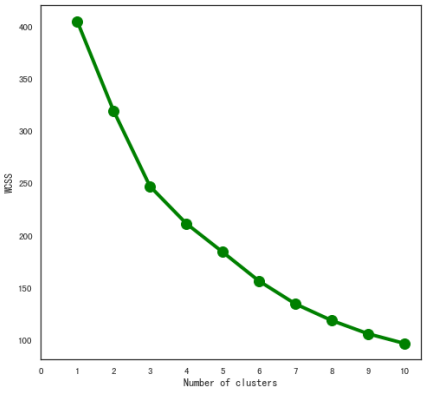
通过查阅文献，发现在以往研究的研究中，一般将亚洲各地发现的古代钾玻璃进一步划分为中等$Ca\cdot Al$、低Ca和低Al，3 个亚类。但是由于样本数目限制，此类划分准则在本题中分类效果较差。

2. $PCA-Kmeans$ 聚类模型

对于所选取的化学成分，利用PCA降维得到若干主成分，选择累加方差解释性超过 90%的主成分作为$Kmeans$聚类的输入。

根据**肘部法则**，确定铅钡玻璃的亚类分为 3 类，将高钾玻璃亚类分为 4 类，下图分别为铅钡玻璃的轮廓系数图以及肘部法则图。





根据**轮廓系数**大致处于 0.4-0.5 范围内，说明聚类效果较好，聚类模型较为合理。其中选取 3 个主成分的聚类效果如下，可以看到针对每个类别划分效果较明显。

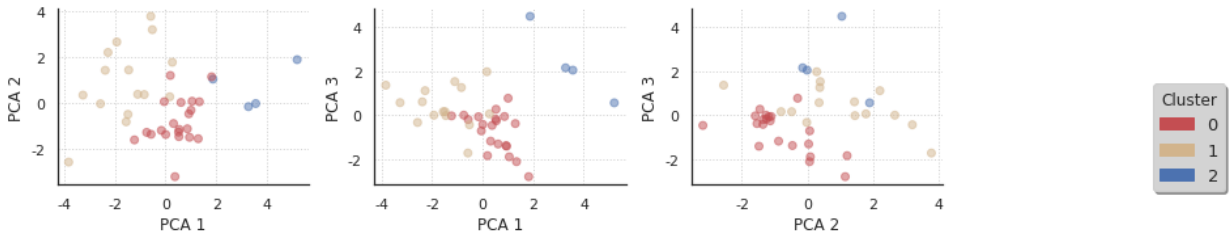


图 6-4 主成分聚类图

3. 亚类划分及合理性解释

雷达图反映聚类中心各个化学成分的含量，并以此为依据结合相关文献，确定最后亚类的划分。如下分别为铅钡玻璃亚类划分和高钾玻璃亚类划分的雷达图。

对于铅钡玻璃，观察雷达图后发现三个聚类在 Na 、 $Al-Ca$ 、 Cu 元素的含量有显著性差异，第一个聚类含 Na ，但是 $Al-Ca$ 以及 Cu 含量很低，第二个聚类 $Al-Ca$ 含量较高，但是基本不含 Cu 和 Na ，而第三个聚类 Cu 含量极高，但是 Na 、 $Al-Ca$ 含量很低。因此，通过分析三个聚类的四种元素含量分布，很清晰地可以将铅钡玻璃分为了三个具有显著性差别的亚类：

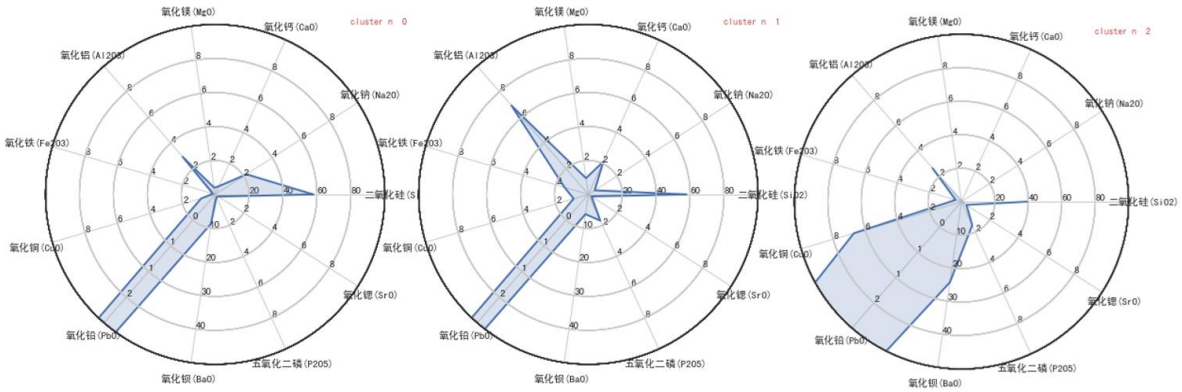


图 6-5 铅钡玻璃亚类划分雷达图

对于高钾玻璃，观察雷达图后发现四个聚类在 $Al-Ca$ 、 Fe 、 Cu 元素的含量有显著性差异，第一个聚类和第二个聚类 $Al-Ca$ 含量在中等水平，但是第一个聚类含较高的 Cu 和 Fe ，而第二个聚类 Fe 含量偏低，第三个聚类 $Al-Ca$ 含量很高，而第四个聚类 $Al-Ca$ 含量很低。因此可以通过 $Al-Ca$ 大致分为三个大类，而结合 Fe 和 Cu 的具体含量，可以将 $Al-Ca$ 中等水平的类进一步细化成两个小类，这样就得到了钾玻璃的四个亚类：

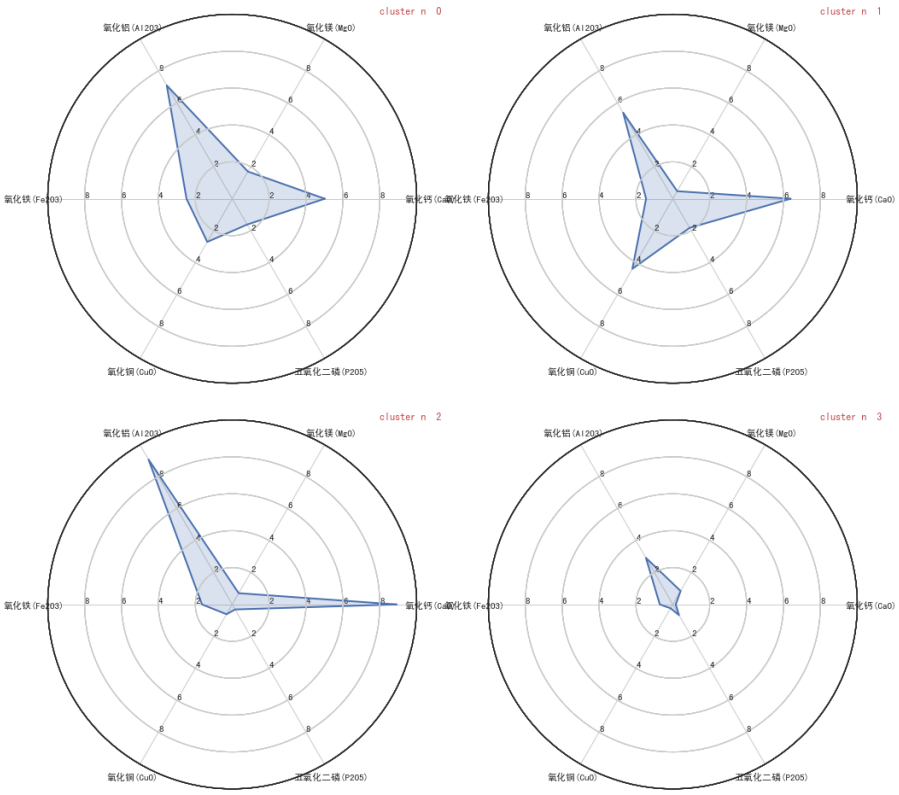


图 6-6 高钾玻璃亚类划分雷达图

6.2.3 合理性与敏感性分析

对于敏感性的分析，，考虑对每种玻璃的样本，随机抽取聚类所用到的特征进行扰动，将结果乘上 PCA 的坐标变换矩阵，得到降维后的坐标点，重新带回已经建立好的聚类模型中，和聚类中心计算欧氏距离，进行分类，和原本样本点类别对比，最终计算准确率。

灵敏度扰动变化 rate 分别取 3%，5%，10%、15%、20%和 30%，对于每个样本的数据，钾玻璃随机取三个化学成分进行 rate 的扰动，铅钡玻璃随机取五个成分进行上下幅度为 rate 的扰动。

进行 100 轮测试后的具体结果计算如下表所示：

表 6-2 铅钡玻璃扰动灵敏度检验

	3%	**5%**	**10%**	**15%**	**20%**	**30%**
$(Pb,Ba\cdot Al\cdot Ca)$	100%	100%	99%	97%	94%	90%
$(Pb,Ba\cdot Cu)$	100%	100%	100%	99%	97%	92%
$(Pb,Ba\cdot Na)$	100%	100%	100%	100%	97%	93%

表 6-3 高钾玻璃扰动灵敏度检验

	3%	5%	10%	15%	20%	30%
中Al中Ca高Cu高Fe	100%	100%	95%	93%	89%	85%
中Al 中Ca高Cu低Fe	100%	100%	96%	93%	91%	87%
高Al高Ca低Cu	100%	100%	98%	97%	94%	92%
低Al 低Ca	100%	100%	100%	99%	96%	94%

分析上述表格可以发现，对于铅钡玻璃，在样本扰动幅度小于等于 10%时，模型预测的结果和未扰动的情况一致，而当扰动幅度较大，预测的准确率略有降低，但总体保证模型也能保证较好的稳定性和准确性。

而对于钾玻璃，我们可以发现中Al 中Ca的两个亚类，对于扰动变化较为明显，这可能是因为这两个亚类本身的划分就较为接近，受Cu和Fe的变化影响大。总体来说，在扰动小于 10%的情况下，模型的表现十分良好，说明模型具有准确性和稳定性，体现出设计的合理性。

七、 问题三模型的建立与求解

7.1 问题三的求解与分析

针对问题三，首先根据问题二中相同的方差滤波筛选特征、异常值处理、PCA降维，具体处理方法同问题二。然后，利用问题二的聚类模型对样本点铅钡玻璃、高钾玻璃进行亚类分析，并得到亚类标签。由于问题二划分亚类的聚类模型是基于未风化或者预测为风化前的化学成分进行分类，故对表单 3 中的数据进行预测时，需要利用问题一的预测模型将风化的数据样本预测为风化前的。并根据问题二中决策树的划分，先将表单 3 中的样本初步划分为高钾玻璃、铅钡玻璃。再根据不同玻璃类别，最后利用$XGBoost$算法进行亚类的预测，结合问题二的聚类预测结果进行比对，检验模型的合理性。

针对问题三模型的灵敏度分析，一方面，模型的样本集特征是基于PCA降维后的主成分，针对$XGBoost$模型的分类结果，使用SHAP解释性模型分析其主成分的重要性，选取对分类模型具有较大贡献的主成分，将其数值进行波动，检验最后的分类结果波动情况；另一方面，观察$XGBoost$损失函数随样本点增加的变化情况，发现其损失函数值逐渐收敛，且数值的置信区间逐渐缩短，证明模型逐渐趋于稳定。

其具体流程如下

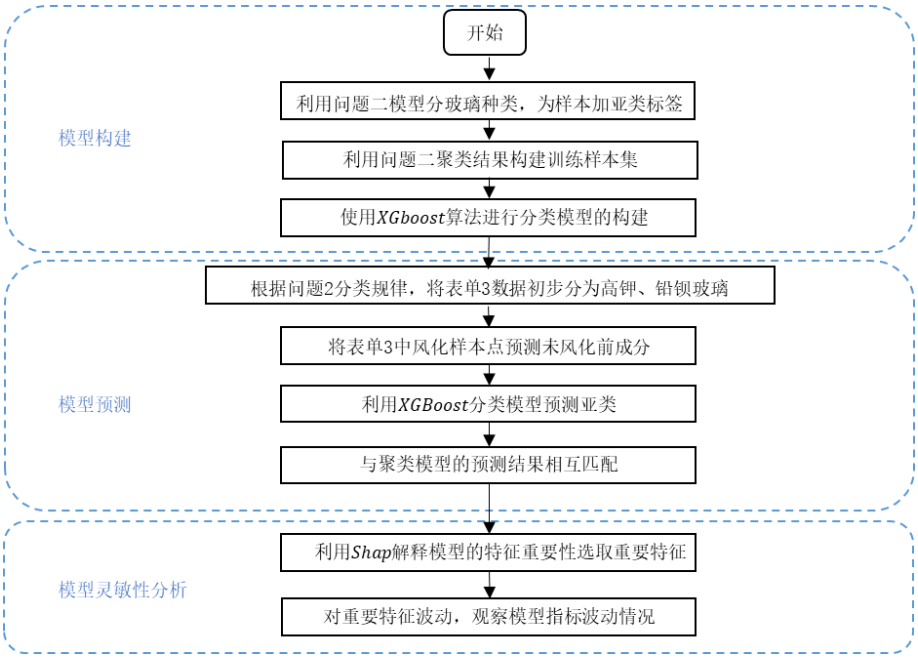


图 7-1 玻璃类别成分分布图

7.2 问题三的模型构建

7.2.1 分类模型选择、样本集构建

利用 $XGBoost$ 算法对问题二中的聚类结果，分为铅钡玻璃、高钾玻璃分别建立分类模型。由于本题中的样本数较少，模型分类预测很可能出现过拟合现象。故综合考虑准确度、拟合情况，我们选用 $XGBoost$ 算法即极度梯度提升树模型，作为问题三的分类算法，由于其模型支持列抽样，可以一定程度上降低过拟合问题，同时兼顾集成学习分类器的高表现力的优势，因此作为本题的分类器较为合适。由于篇幅受限，其具体的数学推导及算法流程见附录。

该模型的样本集标签选择为问题二聚类模型的亚类标签，结合铅钡玻璃、高钾玻璃分别预测。

7.2.2 模型预测

问题三中需要对表单三的样本进行分类，考虑到问题二中 PbO 的成分含量可以直接将铅钡、高钾玻璃区分开，对其分大类的意义不大。故我们考虑分别结合高钾、铅钡玻璃，利用问题二的亚类标签对表单 3 中的数据进行亚类分类，最终高钾、铅钡的亚类分类结果如下所示

表 7-1 表单 3 样本类别预测

文物编号	预测类别
$A1$	中 Al 中 Ca 高 Cu 高 Fe 高钾玻璃
$A2$	高 $Al-Ca$ 铅钡玻璃
$A3$	高 $Al-Ca$ 铅钡玻璃
$A4$	高 $Al-Ca$ 铅钡玻璃
$A5$	高 $Al-Ca$ 铅钡玻璃
$A6$	中 Al 中 Ca 高 Cu 低 Fe 高钾玻璃

$A7$	中Al 中Ca高Cu低Fe高钾玻璃
$A8$	高Cu铅钡玻璃

7.2.3 模型灵敏度检验

1. 灵敏度检验方法

针对问题三构建的$XGBoost$分类模型，我们采用波动重要特征的方法进行检验。重要特征的筛选欲通过Shap解释性模型得到特征的重要程度分布情况，然后选取重要的主成分进行波动，观察分类模型的输出情况。同时结合问题二聚类模型的结果进行对照分析。

2. Shap解释模型

Shap解释模型是受到合作博弈论启发的一个加性解释模型，其核心思想是计算特征输出的边际贡献，再从全局和局部两个层面对“黑盒模型”进行解释。Shap构建一个加性的解释模型，所有的特征都被视为“贡献者”。模型会为每个特征分配一个Shap $value$作为对最终模型的边际贡献值，其计算思想为计算将某特征加入到模型时的边际贡献，再考虑到该特征在所有特征序列情况下的不同贡献，计算期望值，最终得到该特征的Shap $baseline\ value$

将第j个特征的Shap $value$记做为$\phi_j(val)$，其计算公式如下

$$\phi_j(val)=\sum_{S\subseteq\{1,2,\cdots,p\}\backslash\{j\}}\frac{|S|!(p-|S|-1)!}{p!}\left(val(S\cup\{j\})-val(S)\right) \tag{3}$$

$$val_x(S)=\int\hat{f}(x_1,\ldots,x_p)d\,\mathbb{P}_{x\notin S}-E_X\left(\hat{f}(X)\right) \tag{4}$$

其中S为特征空间的子集，p为总特征数，$val_x(S)$即为固定子集中的特征的模型输出期望与模型输出期望之差。

3. 灵敏度分析

首先根据$XGBoost$分类模型的训练过程，其损失函数值随训练样本数的增加逐渐趋于收敛，且其置信区间逐渐缩小，表明其训练模型趋于稳定。根据Shap解释模型，计算$XGBoost$训练样本集中每个主成分的Shap $value$，将其作为对模型的贡献值，选取，由于篇幅受限，这里只放入铅钡玻璃分类模型的训练过程和各个主成分的Shap $value$值。

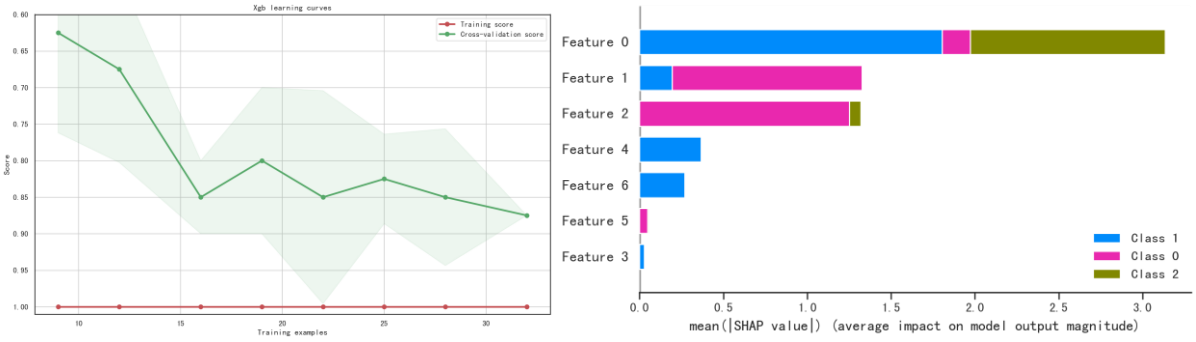


图 7-2 铅钡玻璃训练效果和各个特征Shap $value$

根据可知Shap $value$第一个主成分对于第一类和第三类的重要性很大,而第二和第三个主成分对于第二类重要很大,由此为依据来去进行扰动分析，得到如下结果。

表 7-2 铅钡玻璃扰动灵敏度检验

	3%	**5%**	**10%**	**15%**	**20%**	**30%**
$(Pb,Ba\cdot Al\cdot Ca)$	100%	100%	100%	100%	98%	97%
$(Pb,Ba\cdot Cu)$	100%	100%	100%	100%	97%	95%
$(Pb,Ba\cdot Na)$	100%	100%	100%	100%	98%	96%

表 7-3 高钾玻璃扰动灵敏度检验

	3%	**5%**	**10%**	**15%**	**20%**	**30%**
中Al中Ca高Cu高Fe	100%	100%	100%	100%	100%	96%
中Al 中Ca高Cu低Fe	100%	100%	100%	100%	98%	95%
高Al高Ca低Cu	100%	100%	100%	100%	98%	97%
低Al 低Ca	100%	100%	100%	100%	97%	95%

八、 问题四的建立与求解

8.1 问题四的求解与分析

针对问题四，利用$FP-Growth$**算法**进行**关联规则挖掘**，同时建立灰色关联系数矩阵进行变量关联性挖掘。首先构建关联规则挖掘模型，剔除SO_2等特殊变量，依据等距分箱分别对不同类别有无风化四组数据进行预处理，用 1,2,3,4,5 分别表示其与自身的含量对比的大小，其中**数值越大代表其处于该成分较高含量水平**，接着依据化学成分名称及其对应的含量大小构建初始项集，最终通过$FP-Growth$算法进行关联规则挖掘，分别分析不同类别有无风化的化学成分之间的关联性。

第二步，构造**灰色关联度分析模型**，首先对变量进行筛选，剔除检测出的频率较低的变量以及变量含量波动幅度较小的变量，对其余的变量构建**自关联矩阵**，即分别将每列化学成分分别作为参考数列计算其与其余列化学成分的灰色关联度。

最终结合**关联规则挖掘和灰色关联度分析**，对不同类别玻璃的化学成分之间关联关系进行差异性分析。

具体流程如下

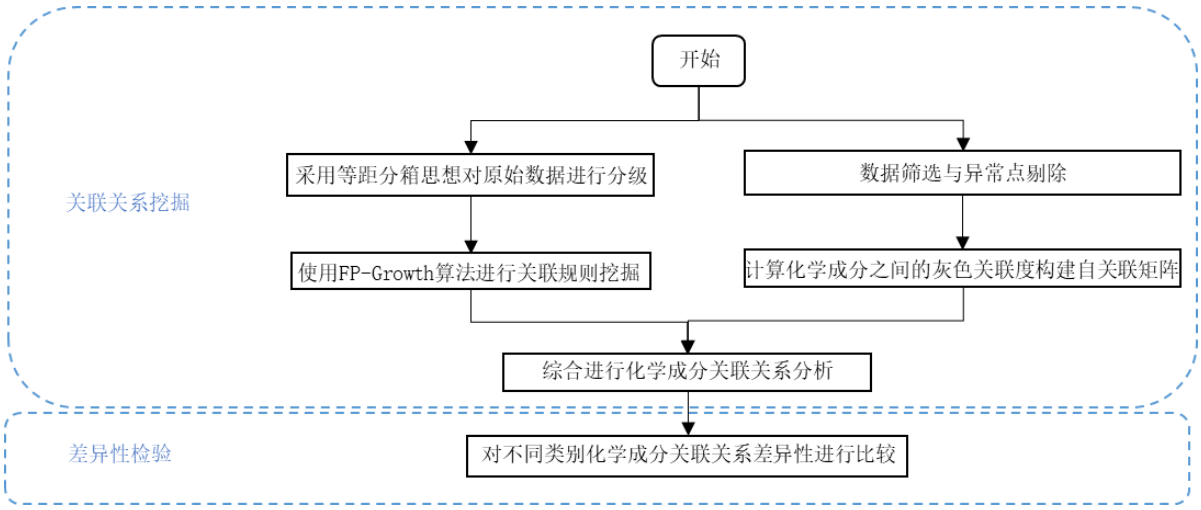


图 8-1 问题四分析流程

8.2 问题四的模型构建

8.2.1 关联规则挖掘模型

1. 初始项集的构建

依据等距分箱的思想对化学成分进行分类，对每个化学成分含量x进行如下划分：

表 8-1 分箱定级原则

$\frac{x}{x_{max}}$	$[0,0.2)$	$[0.2,0.4)$	$[0.4,0.6)$	$[0.6,0.8)$	$[0.8,1]$
分级数	1	2	3	4	5

其中x_{max}为每个化学成分在不同玻璃类型有无风化中的最大含量。

对每个数据依据其化学成分名称和含量分级构建初始项，最终得到初始项集。（项集：指若干带有含量标识的化学成分的集合）

2. 关联规则的求解

项集之间关联规则的挖掘算法通常有经典的算法以及$FP-Growth$算法，$Apriori$算法的核心思想是通过连接产生候选项及其支持度然后通过剪枝生成频繁项集，但是其由于需要多次扫描初始项集而带来的较高的时间复杂度，$FP-Growth$算法弥补了$Apriori$算法的这一缺陷，使得算法更加高效，因此这里决定采用$FP-Growth$算法来进行关联规则挖掘。

$FP-Growth$算法流程如下：

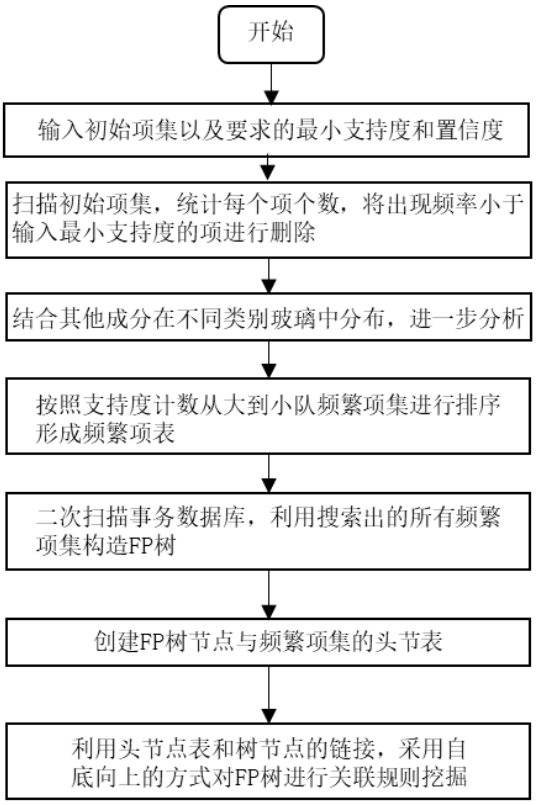


图 8-2 FP-Growth 算法流程

依据上述算法求解可以得到变量之间关联关系的各项指标，包括支持度，置信度，提升度，下面对这三个指标进行简要介绍：

- **支持度**：表示项集$\{X,Y\}$在总项集里面出现的概率，即

$$support(X \to Y) = P(X,Y)$$

- **置信度**：表示在先决条件 X 发生的情况下，由关联规则“$X \to Y$”推出Y的概率，即

$$confidence(X \to Y) = P(Y \mid X) = \frac{P(X,Y)}{P(X)} = \frac{P(XUY)}{P(X)}$$

- **提升度**：表示含有X的条件下，同时含有Y的概率与Y总体发生的概率之比，可反映X对Y发生概率的提升程度，即

$$lift(X \to Y) = \frac{P(Y \mid X)}{P(Y)}$$

（1）未风化铅钡玻璃化学成分关联分析：

根据$FP-Growth$关联规则挖掘算法得到的结果如下（完整结果可查阅附录）：

表 8-2 未风化铅钡玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	BaO 分级 1	CuO 分级 1	30.4%	100%	1.53

2	SiO_2 分级 5	CuO 分级 1	39.1%	100%	1.53
3	K_2O 分级 1 SiO_2 分级 5	CuO 分级 1	26.1%	100%	1.53
⋮	⋮	⋮	⋮	⋮	⋮
10	BaO 分级 1	CuO 分级 1 SiO_2 分级 5	26.1%	85.7%	2.19
11	PbO 分级 3 CuO 分级 1	SiO_2 分级 5	26.1%	85.7%	2.19
12	P_2O_5 分级 1	CuO 分级 1	39.1%	81.8%	1.25

我们按照设定条件所筛选出的未风化铅钡玻璃的关联分析结果如表 8-2 所示。为方便分析关联规则挖掘结果，我们依据表格整理出部分图示如下：

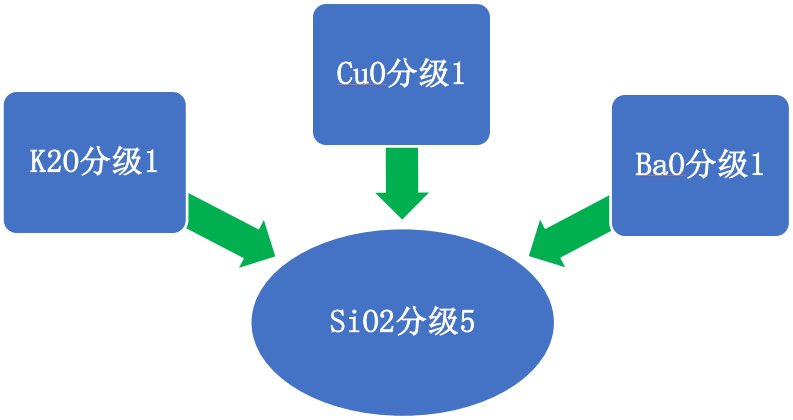


图 8-3 SiO_2 与 CuO，BaO，K_2O 的关联关系

依据上图我们可以知道，变量 K_2O 分级 1，BaO 分级 1，CuO 分级 1，SiO_2 分级 5 相互之间均有着较高的支持度，图中绿色箭头表明其余变量对 SiO_2 分级 5 的提升度较高，说明在 K_2O，CuO 或者 BaO 含量较低时我们有**较大把握**推断出 SiO_2 含量较高，即此时石英玻璃有着更高的纯度。

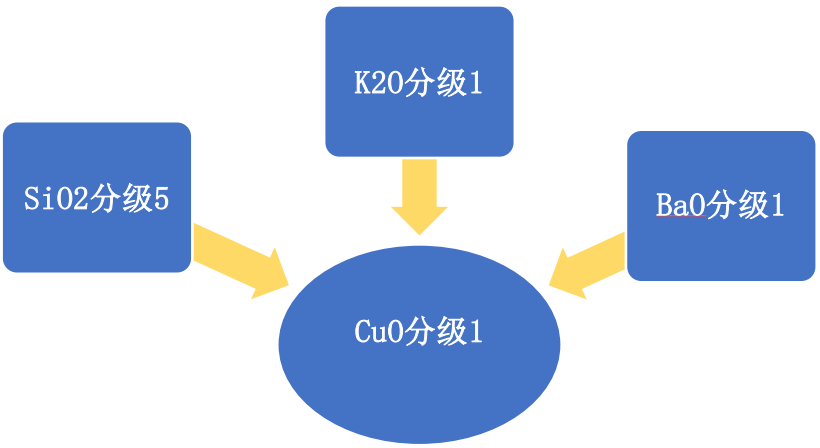


图 8-4 CuO 与 SiO_2，BaO，K_2O 的关联关系

图中**黄色**表明其余变量对CuO分级 1 的支持度较低，因此我们仅有**一定把握**在已知K_2O，BaO含量较低或者SiO_2含量较高时推断CuO含量较低。同时根据表格中的第12项，P_2O_5和CuO的组合虽然有着较高的支持度，但其提升度仅为1.25，我们查阅程序输出结果可知，CuO单独作为一个项集时其支持度为 65.2%，初步推断是由于未风化铅钡玻璃内CuO含量普遍较低导致其与P_2O_5的组合同样有着较高的支持度，而不能说明CuO含量低可由P_2O_5推断而出。

除了挖掘出SiO_2与其余变量的关联关系外，表 8-2 中同样也展示了Al_2O_3分级 1 和CaO分级 1 较高的支持度，且其提升度为 1.83，相对较高，可以说明在未风化铅钡玻璃中Al_2O_3含量较低时可以推断出CaO同样有着较低的含量。

（2）风化铅钡玻璃化学成分关联分析

根据$FP-Growth$关联规则挖掘算法得到的结果如下（完整结果可查阅附录）

表 8-3 风化铅钡玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	SrO分级 3	Na_2O分级 1	34.6%	100%	1.18
2	Al_2O_3分级 2	Na_2O分级 1	30.8%	100%	1.18
⋮	⋮	⋮	⋮	⋮	⋮
10	SiO_2分级 3	CuO分级 1	34.6%	85.7%	1.39
11	CuO分级 1	Na_2O分级 1	50.0%	85.7%	1.01
12	Al_2O_3分级 1	Na_2O分级 1	50.0%	81.8%	0.97

通过观察表 8-3 我们可以知道，对于风化铅钡玻璃，其余大部风化学成分与Na_2O有着较高的支持度，但其提升度普遍较低，同时统计Na_2O含量可知，在多数风化铅钡玻璃中Na_2O有着较低的含量，从而影响了各化学成分与Na_2O之间支持度的测量，因此，我们决定暂时删去Na_2O，重新对剩余变量进行关联规则挖掘，以更好地检测变量关联关系。删去Na_2O后的关联规则挖掘结果如下表所示：

表 8-4 删去Na_2O后风化铅钡玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	Fe_2O_3分级 1	Al_2O_3分级 1	26.9%	100%	1.625
2	PbO分级 4	CuO分级 1	38.4%	90.9%	1.477
3	SiO_2分级 3	CuO分级 1	34.6%	85.7%	1.39

根据表 8-4 所示，删去Na_2O进行关联规则后按照同样的分析逻辑可以发现，在风化铅钡玻璃中，Fe_2O_3含量较低时，有一定把握可以推断Al_2O_3含量同样较低；PbO和SiO_2含量相对较高时，CuO含量可能较低。

（3）未风化高钾玻璃化学成分关联分析

根据$FP-Growth$关联规则挖掘算法得到的结果如下（完整结果可查阅附录）

表 8-5 未风化高钾玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	K_2O 分级 5	SiO_2 分级 4	33.3%	100%	1.33
2	CaO 分级 5	SiO_2 分级 4	41.67%	100%	1.33
⋮	⋮	⋮	⋮	⋮	⋮
11	Al_2O_3 分级 3 K_2O 分级 5	SiO_2 分级 4	25.0%	100%	1.33

根据表 8-5 的关联规则挖掘结果，发现在未风化高钾玻璃中，绝大部风化学成分相对较高的含量均有一定把握推断出SiO_2分级 4，假设在未风化高钾玻璃中，大部分金属元素均对提升SiO_2含量有帮助作用，但是由于其余金属元素本身为杂质，因此无法使得SiO_2的含量达到最高。

下面为进一步分析未风化高钾玻璃其余化学成分之间的关联关系，暂时删去SiO_2重新进行关联规则挖掘，得到的结果如下表所示：

表 8-6 删去SiO_2未风化高钾玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	CuO 分级 4	P_2O_5 分级 2	25.0%	100%	3
2	K_2O 分级 4	P_2O_5 分级 2	25.0%	75.0%	2.25
3	K_2O 分级 5	Al_2O_3 分级 3	25.0%	75.0%	1.8
4	K_2O 分级 5	CaO 分级 5	25.0%	75.0%	1.8

根据表 8-6 的关联规则挖掘结果，当CuO或者K_2O含量较高时，有较大可能会降低未风化高钾玻璃中P_2O_5的含量。

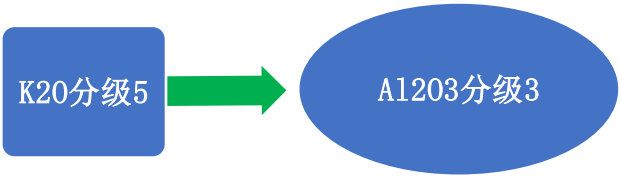


图 8-5 K_2O 与 Al_2O_3 的关联关系

由未风化高钾玻璃原始数据的等距分箱结果可知，Al_2O_3 的分级结果大多为 3-5 之间，说明在这一玻璃中，Al_2O_3 含量普遍较高，再结合上图可知，尽管 Al_2O_3 分级 3 并不是一个相对较低的含量，但 K_2O 含量较高时可一定程度降低 Al_2O_3 的含量。同理可说明，K_2O 含量较高时可以推断出未风化高钾玻璃中 CaO 含量同样较高。

（4）风化高钾玻璃化学成分关联分析

根据 $FP-Growth$ 关联规则挖掘算法得到的结果如下：

表 8-7 化高钾玻璃关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	CuO 分级 3	SiO_2 分级 5	50%	100%	1
2	Fe_2O_3 分级 5	SiO_2 分级 5	50%	100%	1

由于风化高钾玻璃提供的数目较少，故所能挖掘出的关联规则较少。

由表 8-7 结合数据预处理结果可知，SiO_2 含量普遍较高，同时伴随着 CuO 较低的含量和 Fe_2O_3 较高的含量。

上述分别分析了不同玻璃类型有无风化内部的关联规则挖掘结果，但由于样本数量太少，仍有一定的局限性，为了进一步挖掘出不同化学成分之间的序列相关性，我们引入灰色关联度分析求解自关联矩阵以辅助分析。

8.2.2 灰色关联度模型

1. 灰色关联度模型的建立

该模型主要任务为求解自关联系数矩阵，一般的求解灰色关联度算法较为常见，本次题目要求化学成分相互之间的关联关系，因此我们依次将每个序列作为参考序列求解灰色关联度，从而构建自关联系数矩阵，具体求解步骤可见附录。

2. 灰色关联度模型的求解

下面分别以未风化铅钡玻璃以及风化高钾玻璃为例进行结果说明，自关联矩阵热力图如下所示：

图 8-6 未风化铅钡、风化高钾玻璃化学成分自关联矩阵

➢ **未风化铅钡化学成分关联性分析：**

如图 8-6-1 所示，关联系数较高的化学成分有CuO， BaO；P_2O_5与BaO，CuO，Fe_2O_3，CaO；CaO和Fe_2O_3；Na_2O和CuO。

下面以CuO， BaO两个化学成分为例进行关系系数结果合理性分析：

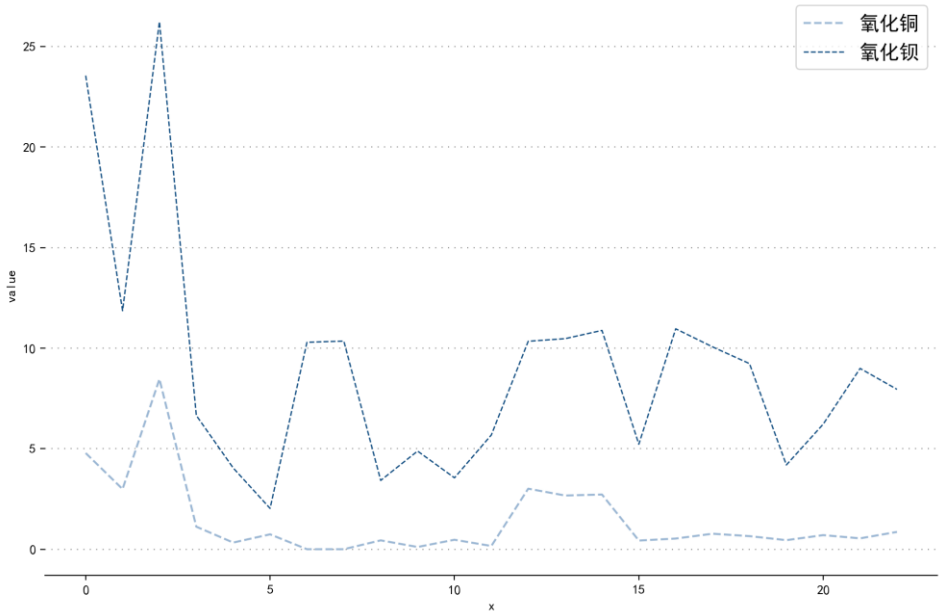


图 8-7 未风化铅钡玻璃CuO和BaO变化趋势

由上图可以看出，CuO与BaO变化趋势呈现较高的相关性，因此该灰色自关联矩阵计算结果合理。

➢ **风化高钾化学成分关联性分析：**

如图 8-6-2 所示，关联系数较高的化学成分由Al_2O_3和Mg_2O_3；Al_2O_3和CaO；K_2O和Fe_2O_3；CuO和P_2O_5。下面以Al_2O_3和CaO为例进行风化高钾灰色关联系数的

结果合理性分析：

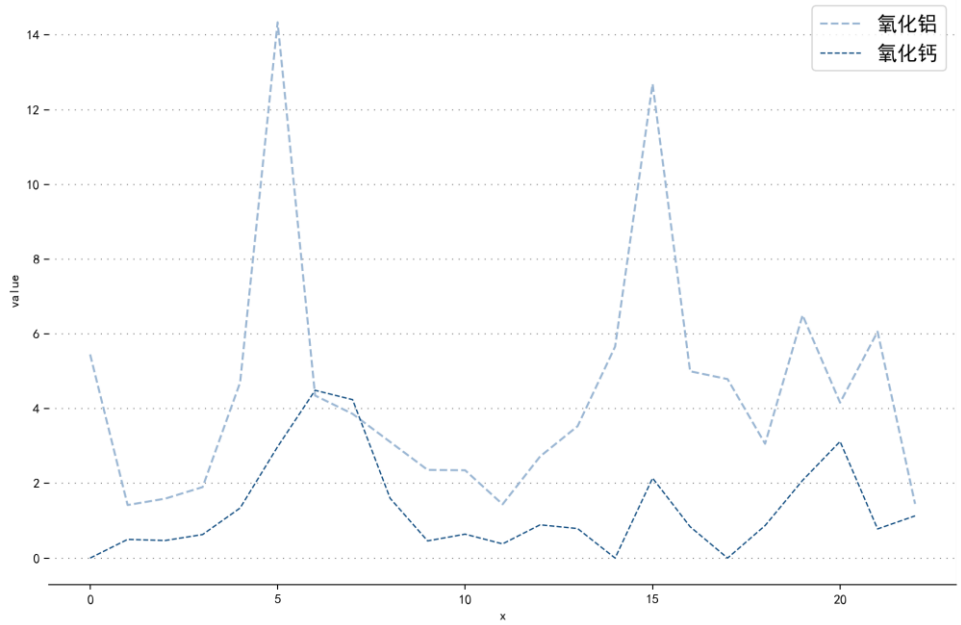


图 8-8 风化高钾玻璃Al_2O_3和CaO变化趋势

由上图可以看出，CuO与BaO变化趋势呈现较高的相关性，因此该灰色自关联矩阵对风化高钾玻璃化学成分的计算分析结果较为合理，由于篇幅受限，其余分析结果可见附录。

8.2.3 不同玻璃类型有无风化化学成分关联关系总结

1. 未风化铅钡玻璃化学成分关联关系：

K_2O,CuO或者BaO含量较低时，SiO_2含量较高；Al_2O_3含量较低时，CaO有着较低的含量。同时，CuO，BaO；P_2O_5与BaO，CuO，Fe_2O_3，CaO；CaO和Fe_2O_3；Na_2O和CuO有着较高的灰色关联度。

2. 未风化高钾玻璃化学成分关联关系：

绝大部风化学成分含量较高时，均可以一定程度提升SiO_2的纯度，但其余化学成分本身作为杂质也会降低SiO_2的纯度。K_2O含量较高时，Al_2O_3含量相对较低，同时CaO含量相对较高。同时PbO和BaO；K_2O和CaO；Fe_2O_3和CuO有着较高的灰色关联度。

3. 风化铅钡玻璃化学成分关联关系：

Fe_2O_3含量较低时，Al_2O_3含量同样较低；PbO和SiO_2含量相对较高时，CuO含量可能较低。同时Al_2O_3与Na_2O，Mg_2O_3，Fe_2O_3，CuO等化学成分以及P_2O_5和CaO之间有着较高的灰色关联度。

4. 风化高钾玻璃化学成分关联关系：

SiO_2含量普遍较高，同时伴随着CuO较低的含量和Fe_2O_3较高的含量。同时Al_2O_3与Fe_2O_3，Mg_2O_3；CuO和P_2O_5有着较高的灰色关联度。

8.2.4 不同玻璃类型化学成分关联关系差异性比较

1. SiO_2纯度与其余化学成分关联性差异比较:

在未风化铅钡玻璃中，若SiO_2含量较高，则需要K_2O，CuO或者BaO等化学成分保持较低的含量。而在未风化高钾玻璃中，绝大部风化学成分含量较高时，均可以一定程度提升SiO_2的纯度。

在风化后的铅钡玻璃中，SiO_2含量受其余变量的综合影响，其不同样本含量差异较大。而在风化后的高钾玻璃中，SiO_2含量普遍较高，且此时CuO含量较低和Fe_2O_3含量较高。

2. 具有显著关联关系化学成分的关联性差异比较:

在未风化铅钡玻璃中，Al_2O_3含量较低时，可以一定程度推断CaO同样有着较低的含量，而在未风化高钾玻璃中，Al_2O_3含量相对较低时，CaO含量反而相对较高。

在风化铅钡玻璃中，Fe_2O_3与CuO含量均较低，而在风化高钾玻璃中虽然CuO含量较低，但是Fe_2O_3含量较高。同时在风化铅钡玻璃中P_2O_5和CaO之间有着较高的灰色关联度，而在风化高钾玻璃中P_2O_5和CuO有着较高的灰色关联度。

九、 模型的改进与评价

9.1 模型优点

1. 本文充分考虑**实际问题、数据问题**。
   - 本文查阅大量资料，针对问题一的风化前后高钾玻璃的成分变化结合具体文献证明其变化趋势的合理性
   - 针对问题二，**通过查阅大量文献可得，玻璃类别的成分检验大都需要去风化层、进行清洗研磨等操作**，同时风化对于玻璃化学成分影响较大，**据此我们并不简单将风化、未风化玻璃放在一起进行聚类分析，而是通过合理的预测模型，预测风化数据点在未风化时的成分含量**，由此进行聚类分析，符合现实需求。
   - 进行亚类分类时，通过查询资料，结合雷达图得到符合现实的分类结果
2. 本文**处理数据手段**完备
   - 针对问题一，首先针对缺失值并未直接用统计量进行填补，采用基于样本特征相似的KNN算法进行插值
   - 针对严重风化点、未风化点、数据异常点在建模的全部过程中均有涉及考虑。充分增强了模型的表现力、鲁棒性。
3. 本文**数据分析**完备。
   - 首先结合玻璃类型，对风化前后的化学成分的分布，都进行了可视化操作，并针对数据分布差异性均进行了合理的检验。结合可视化与检验结果相互应证，增强文章的说服力和解释性。
   - 针对问题二，探讨分类规律时，不仅仅只考虑到高钾玻璃、铅钡玻璃在PbO特征上含量具有线性可分性，同时还通过数据可视化，进一步考虑其他特征对玻璃类别的影响。
   - 针对问题四，不仅通过**关联规则挖掘**模型，结合实际进行定性分析，还通过灰

色关联分析进行定量统计

4. 本文**合理性检验**完备。
   - 针对问题一的预测模型，将风化的预测结果同未风化的数据分布进行差异性检验，未发现显著性差异，从而说明模型合理性
   - 针对问题三的分类结果同问题二的聚类模型的结果进行比对，进一步说明分类模型构造的合理性。
5. 本文**创新性较强**。
   - 针对问题一，预测模型构建时，充分考虑到样本点较少的缘故，只能简单获取风化前后成分的边际分布规律，较难得到其联合分布情况，故没有采取通过联合分布的预测模型。且同时综合考虑到样本统计规律、小样本点数据、风化前后成分变化的对应关系、异常情况，创新性提出分箱均值预测模型，且通过差异性检验，证明了预测模型的合理性。
   - 针对问题三，在进行分类预测时，充分考虑到了风化、未风化的情况，将风化样本点利用问题一的预测模型进行成分预测，并根据其未风化的结果进行分别分析，符合现实规律。

9.2 模型缺点

1. **关联规则挖掘**的差异性分析缺乏定量分析结果。

9.3 模型的改进和推广

1. 由于高钾玻璃的数据样本点过于少，且其化学成分易受风化影响，对其进行分析时，需要参考大量资料进行人为的合理性分析。对样本数据过于少的模型建模时，其模型的稳定性可能会受到一定程度的影响，可以考虑增加高钾玻璃的数据进一步增强模型的学习能力、稳定性。
2. 配对的数据样本过少。即缺失对于单一样本的风化前后化学成分指标，无法形成对照，只能根据样本的成分分布情况、统计规律进行成分的探索，无法定量获得风化前后成分变化的对应关系。可以考虑增加样本对照点，来检验模型对风化前后的分布影响的合理性，增强预测模型的效益。
3. 由于玻璃表面的不同部分可能受到的风化程度可能不一致，故可以针对单一样本进行多个部位的采样分析，来综合衡量其化学成分指标。

十、 参考文献

[1] 陈波,詹明强,黄梓莘.基于关联规则的库岸边坡监测数据挖掘方法[J].长江科学院院报,2022,39(08):58-64.

[2] 何希杰,劳学苏.抗磨白口铸铁化学成分自关联性分析[J].中国铸造装备与技术,2015(05):58-61.

[3] 陈敏.中铅晶质玻璃表面风化的研究[J].玻璃与搪瓷,1989(06):12-15+28.

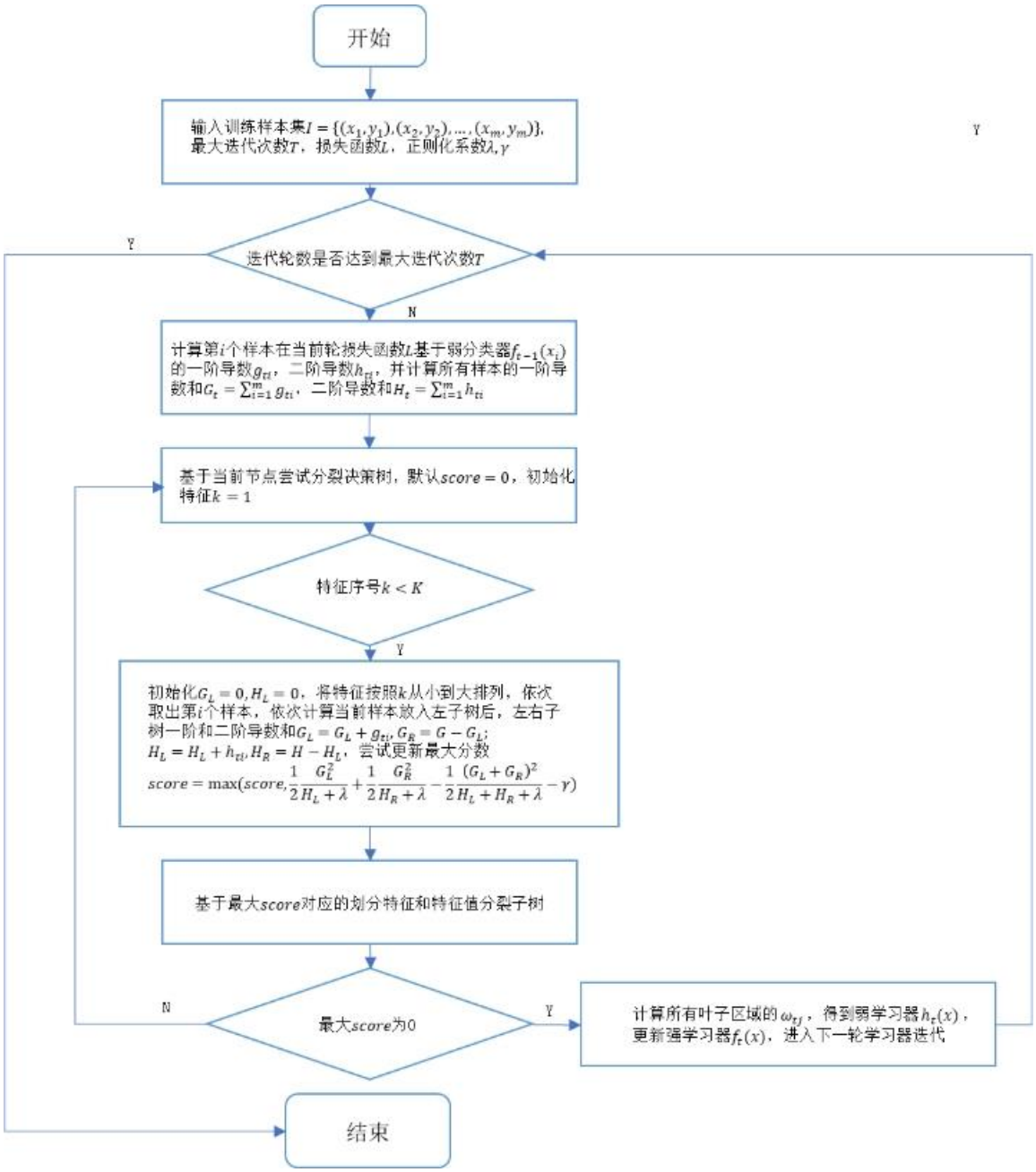
[4] 王承遇,陶瑛.硅酸盐玻璃的风化[J].硅酸盐学报,2003(01):78-85.

[5] 李青会,董俊卿,干福熹.中国早期釉砂和玻璃制品的化学成分和工艺特点探讨[J].广西民族大学学报(自然科学版),2009,15(04):31-41.

十一、 附录

算法流程：

XGBoost：



$Kmeans$ 求解算法
Step1 从数据集中随机选取k 个初始聚类中心$C_i(1\leq i\leq k)$，计算所有对象与聚

类中心的距离，其中$d(x,C_i)=\sqrt{\sum_{j=1}^{m}(x_j-C_{ij})^2}$，将对象划分到距离其最近的聚类中心$C_i$所属的簇中

Step2 将每个簇中所有数据对象的平均值作为新的聚类中心，进行下一次迭代，计算所有样本对象到聚类中心的距离，重新划分到不同的簇中

Step3 重复 **Step2** 步骤，直到聚类中心不再发生变化

Step4 返回所有样本所属簇的信息

灰色自关联矩阵求解算法流程

Step1 将评价指标进行预处理，即进行一致化和无量纲化得到评价矩阵 $B=(b_{ij})_{nxm}$

Step2 确定比较数列（评价对象）和参考数列（评价标准），比较数列为：

$$b_i=\{b_{ij}\mid j=1,2,...,m\},i=1,2,...,n,i\neq t$$

即b_i为第i个评价对象的标准化向量值，参考数列为

$$b_t=\{b_{tj}\mid j=1,2,...,m\},t=1,2,3,\ldots n$$

这里$b_{tj}=\max\limits_{1\le i\le n} b_{ij},j=1,2,...,m$，即分别将每一数列作为参考数列进行关联度分析。

Step3 对每一种参考序列的选取(即 t 取不同的值)，计算其对应的灰色关联系数：

$$\xi_{ij}=\frac{\min\limits_{1\le s\le n}\max\limits_{1\le k\le m}|b_{tk}-b_{sk}|+\rho\max\limits_{1\le s\le n}\max\limits_{1\le k\le m}|b_{tk}-b_{sk}|}{|b_{tj}-b_{ij}|+\rho\max\limits_{1\le s\le n}\max\limits_{1\le k\le m}|b_{tk}-b_{sk}|}$$

$$i=1,2,\ldots,n,\ j=1,2,\ldots,m$$

ξ_{ij}为比较数列b_i对参考数列b_t在第 j 个指标上的关联系数，其中ρ为分辨系数。

Step4 对每一种参考序列的选取(即 t 取不同的值)，根据各指标权重和灰色关联系数计算灰色关联度：

$$r_{ti}=\sum_{j=1}^{m}w_j\xi_{ij},i=1,2,\ldots,n,t=1,2,\ldots n$$

其中w_j为第 j 个指标的权重，这里由于不清楚各成分影响大小，我们计算r_{ti}时取平均值即可。

Step5 由r_{ti}组成的R_{nxn}矩阵即为各成分的自关联性系数矩阵。

***PCA* 求解算法**

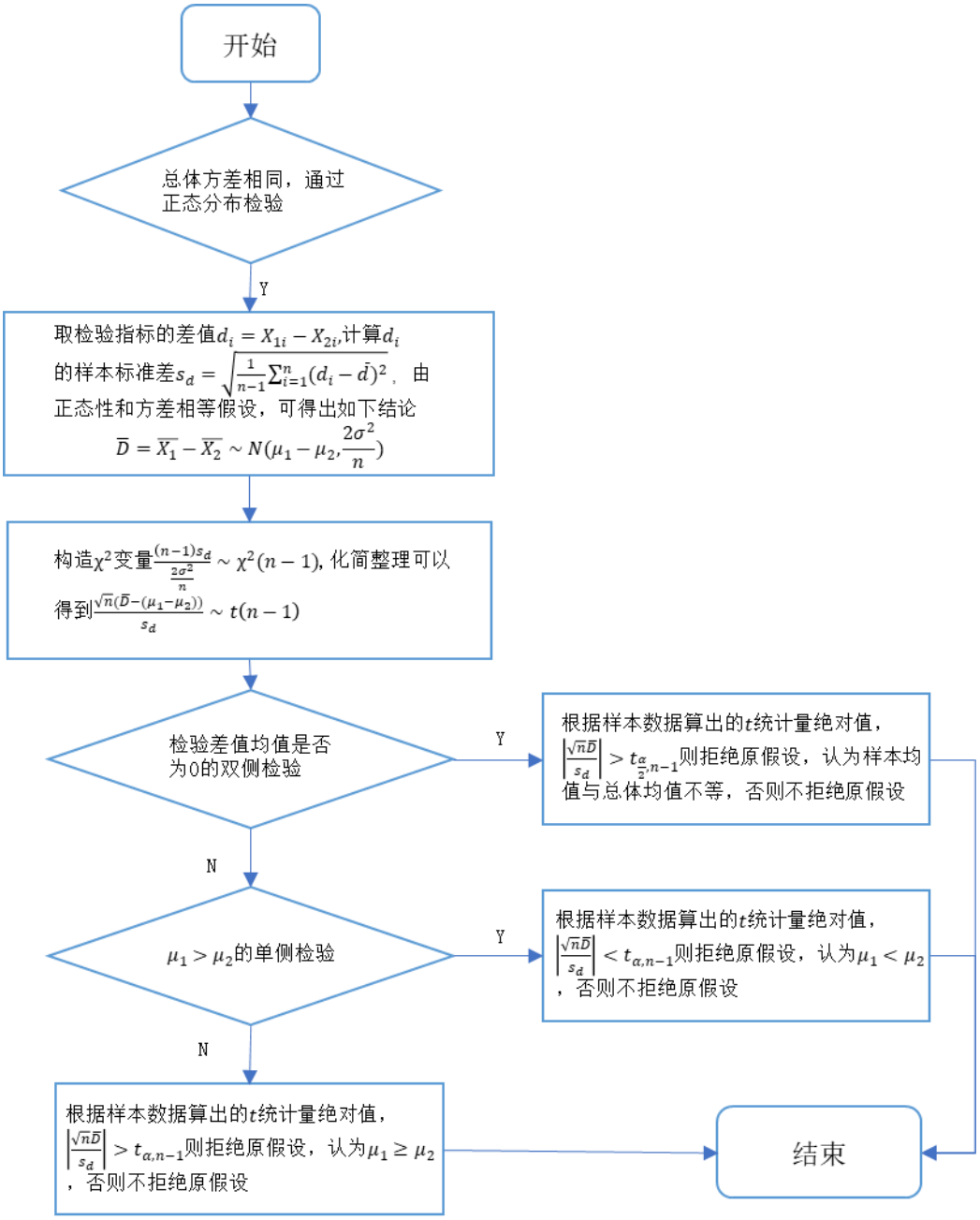
Step1 对样本特征矩阵进行标准化处理，得到标准化矩阵$\boldsymbol{X}$

Step2 求出标准化矩阵$\boldsymbol{X}$所对应的协方差矩阵$\boldsymbol{R}$

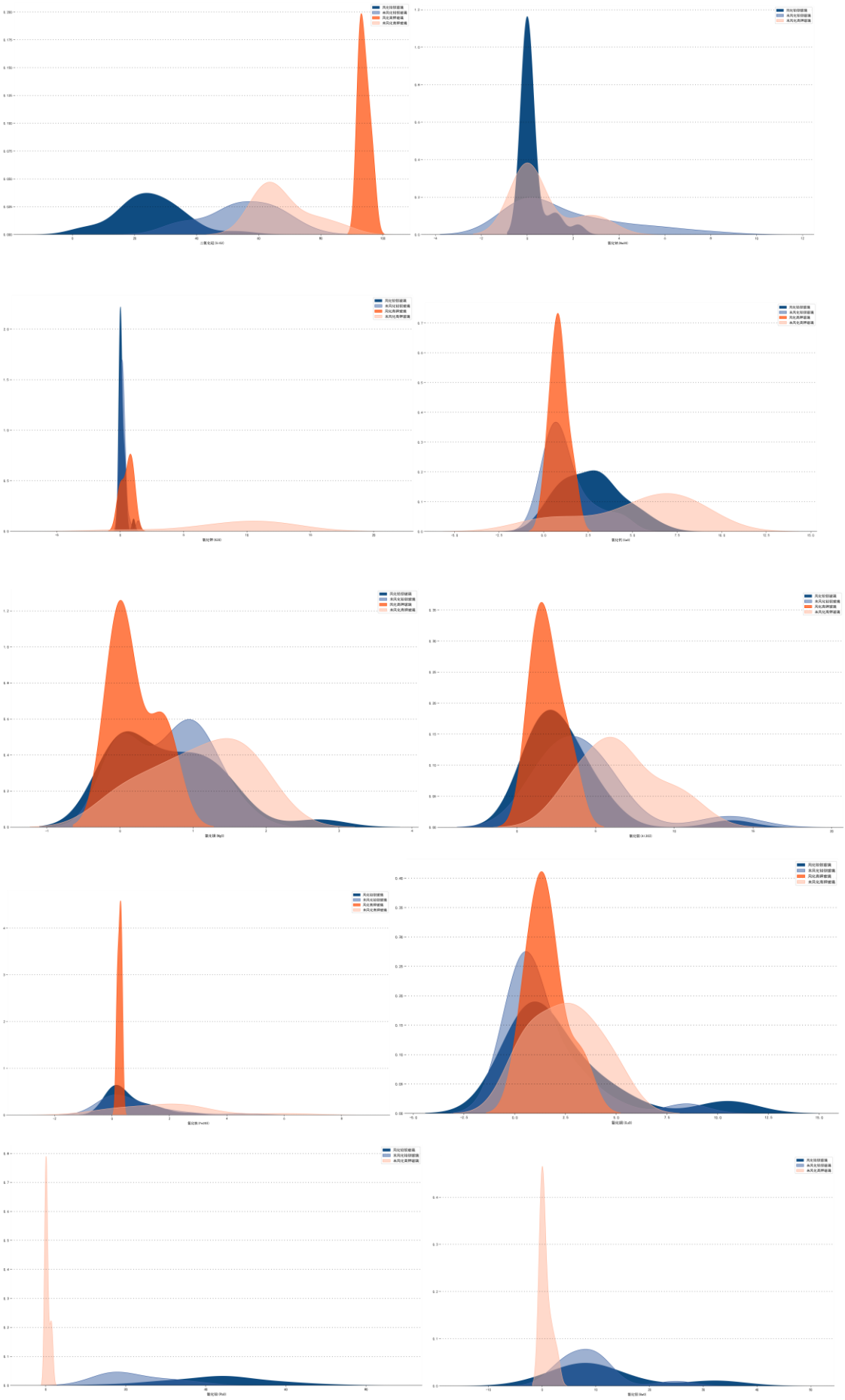
Step3 求出协方差矩阵$\boldsymbol{R}$的特征值与特征向量，将特征向量按对应特征值大小进行排序后，组成特征向量矩阵$\boldsymbol{P}$

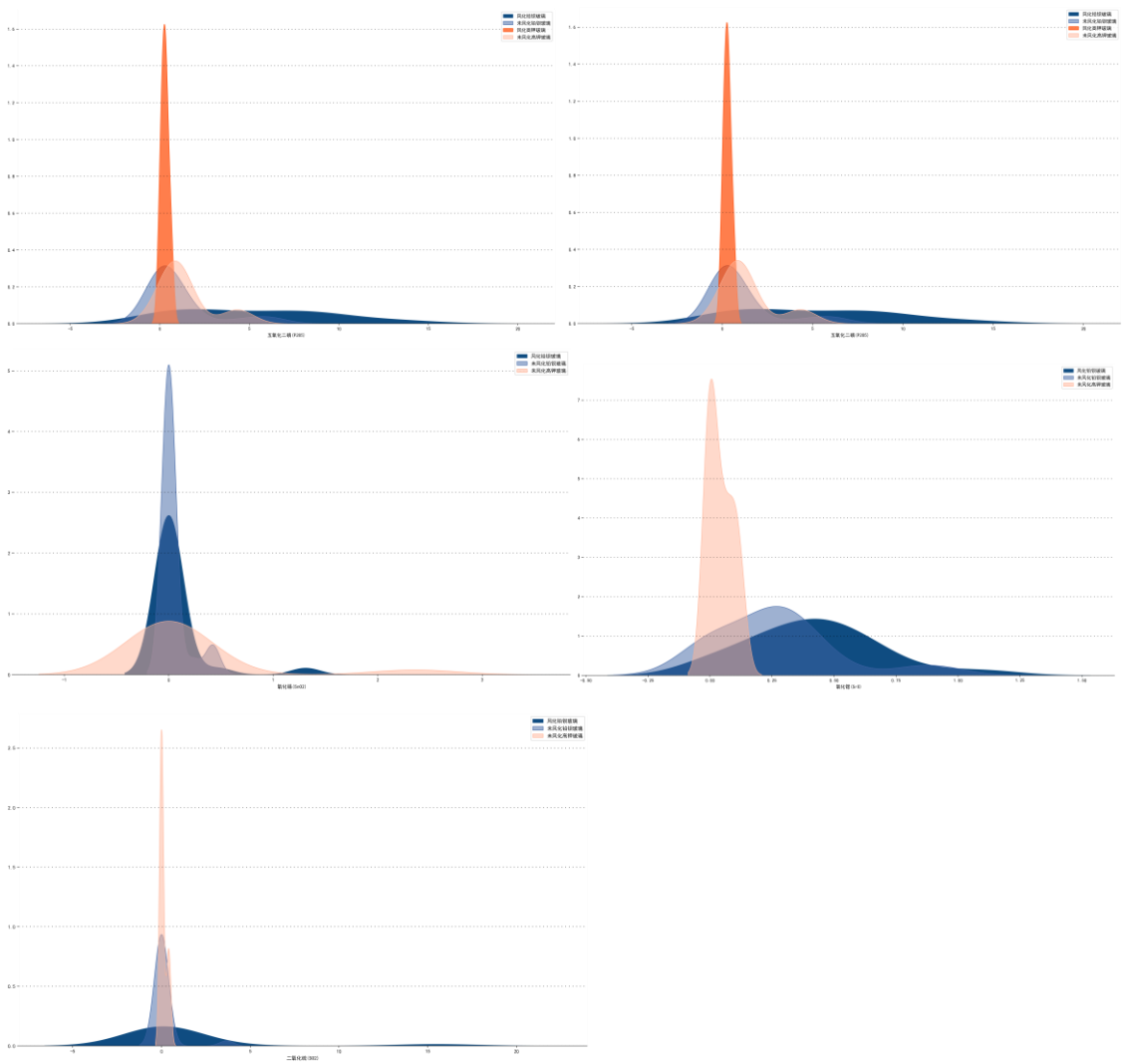
Step4 根据特征值，求出贡献率和累计贡献率，再根据累计贡献率选取合适的主成分，对原特征矩阵进行降维

独立 t-检验：



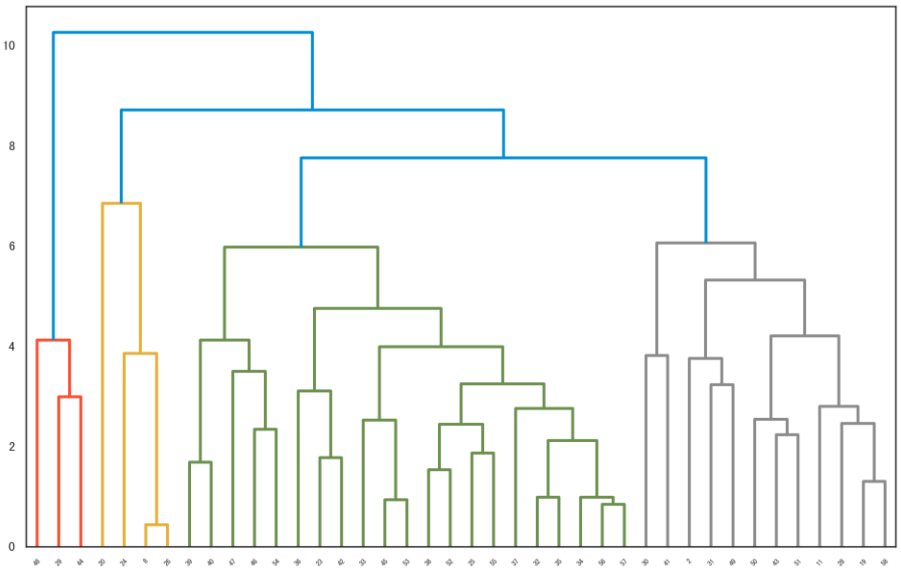
Question1



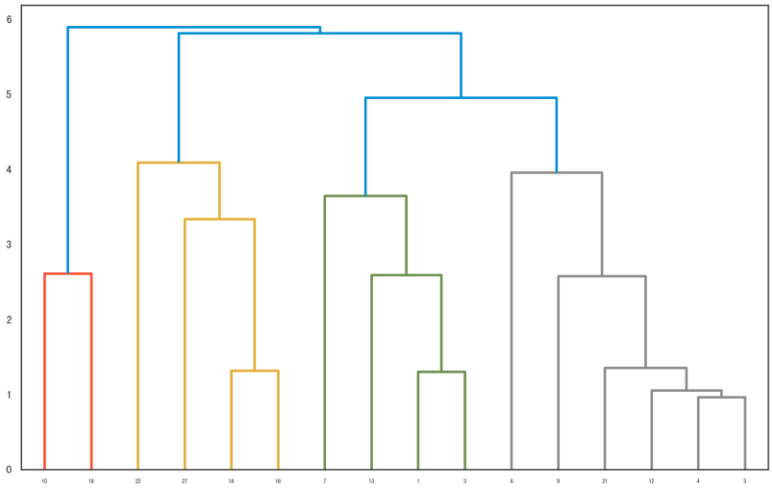


Question 2 层次聚类结果

铅钡



高钾



未风化铅钡关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	BaO 分级 1	CuO 分级 1	30.4%	100%	1.53
2	SiO2 分级 5	CuO 分级 1	39.1%	100%	1.53
3	K2O 分级 1 SiO2 分级 5	CuO 分级 1	26.1%	100%	1.53
4	SiO2 分级 5 BaO 分级 1	CuO 分级 1	26.1%	100%	1.53
5	PbO 分级 3 SiO2 分级 5	CuO 分级 1	26.1%	100%	1.53
6	Al2O3 分级 1	CaO 分级 1	30.4%	87.5%	1.83
7	BaO 分级 1	SiO2 分级 5	26.1%	85.7%	2.19
8	K2O 分级 1 CuO 分级 1	SiO2 分级 5	26.1%	85.7%	2.19
9	CuO 分级 1 BaO 分级 1	SiO2 分级 5	26.1%	85.7%	2.19
10	BaO 分级 1	CuO 分级 1 SiO2 分级 5	26.1%	85.7%	2.19
11	PbO 分级 3 CuO 分级 1	SiO2 分级 5	26.1%	85.7%	2.19

12	P2O5 分级 1	CuO 分级 1	39.1%	81.8%	1.25

风化铅钡关联规则挖掘结果

序号	规则前项	规则后项	支持度	置信度	提升度
1	SrO 分级 3	Na2O 分级 1	34.6%	100%	1.18
2	Al2O3 分级 2	Na2O 分级 1	30.8%	100%	1.18
3	PbO 分级 4	CuO 分级 1	38.5%	90.9%	1.48
4	SiO2 分级 3	Na2O 分级 1	38.5%	90.9%	1.07
5	PbO 分级 3	Na2O 分级 1	30.8%	88.9%	1.05
6	CaO 分级 3	Na2O 分级 1	30.8%	88.9%	1.05
7	Na2O 分级 1	CuO 分级 1	30.8%	88.9%	1.44
8	PbO 分级 4 SiO2 分级 3 CuO 分级 1	Na2O 分级 1	30.8%	88.9%	1.05
9	PbO 分级 4	Na2O 分级 1	34.6%	81.8%	1.01
10	SiO2 分级 3	CuO 分级 1	34.6%	81.8%	1.33
11	CuO 分级 1	Na2O 分级 1	50.0%	81.8%	0.97
12	Al2O3 分级 1	Na2O 分级 1	50.0%	81.8%	0.97

未风化高钾关联规则挖掘结果

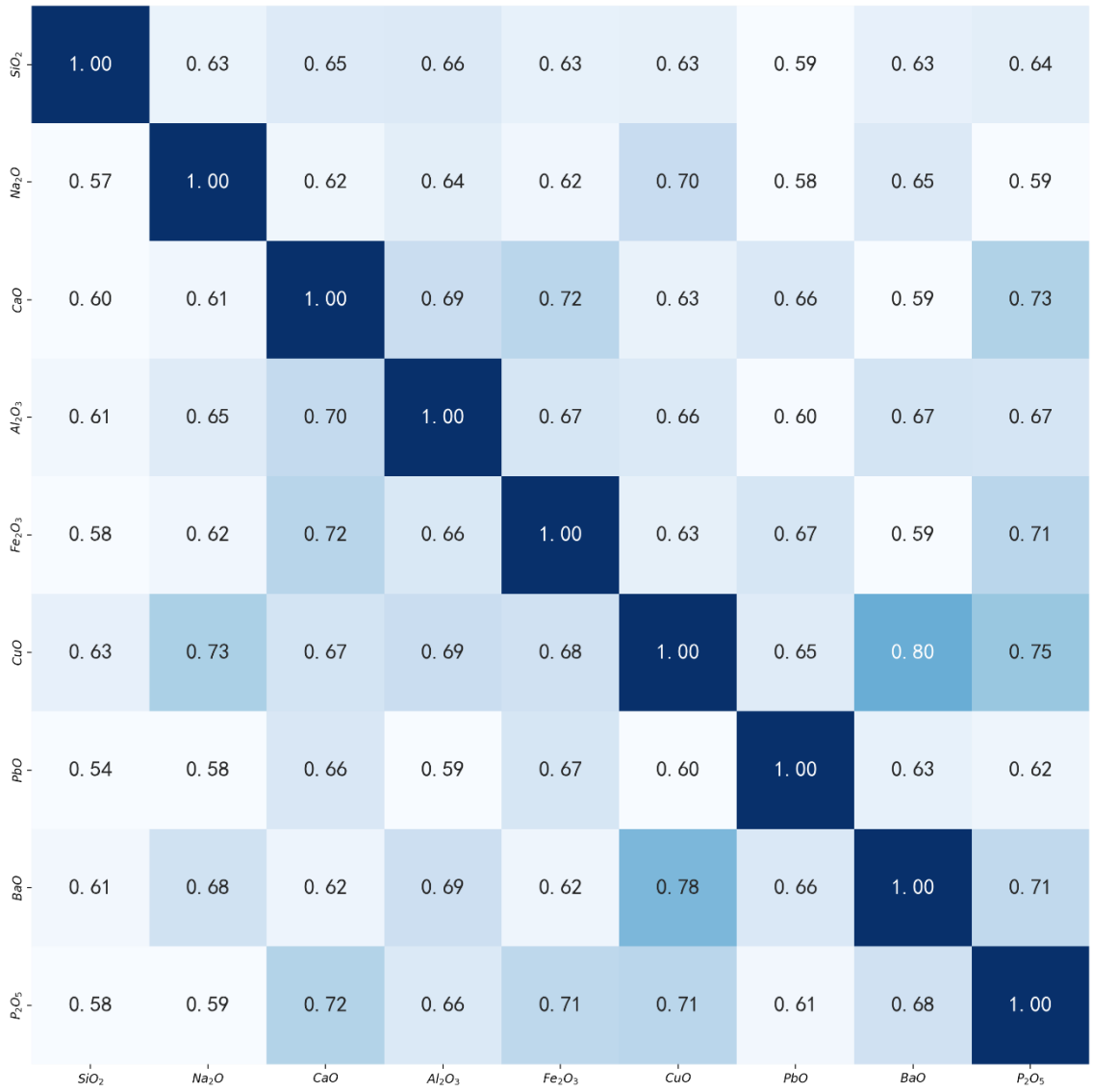
序号	规则前项	规则后项	支持度	置信度	提升度
1	K2O 分级 5	SiO2 分级 4	33.3%	100%	1.33
2	CaO 分级 5	SiO2 分级 4	41.67%	100%	1.33
3	CuO 分级 3	SiO2 分级 4	25.0%	100%	1.33
4	CuO 分级 4	P2O5 分级 2	25.0%	100%	2.4

5	MgO 分级 5	SiO2 分级 4	25.0%	100%	1.33
6	CaO 分级 4	SiO2 分级 4	25.0%	100%	1.33
7	SrO 分级 5	SiO2 分级 4	25.0%	100%	1.33
8	Al2O3 分级 5	SiO2 分级 4	25.0%	100%	1.33
9	CaO 分级 5	SiO2 分级 4	25.0%	100%	1.33
10	Al2O3 分级 3 CaO 分级 5	SiO2 分级 4	25.0%	100%	1.33
11	K2O 分级 5 AL2O3 分级 3 K2O 分级 5	SiO2 分级 4	25.0%	100%	1.33

风化高钾关联规则挖掘结果

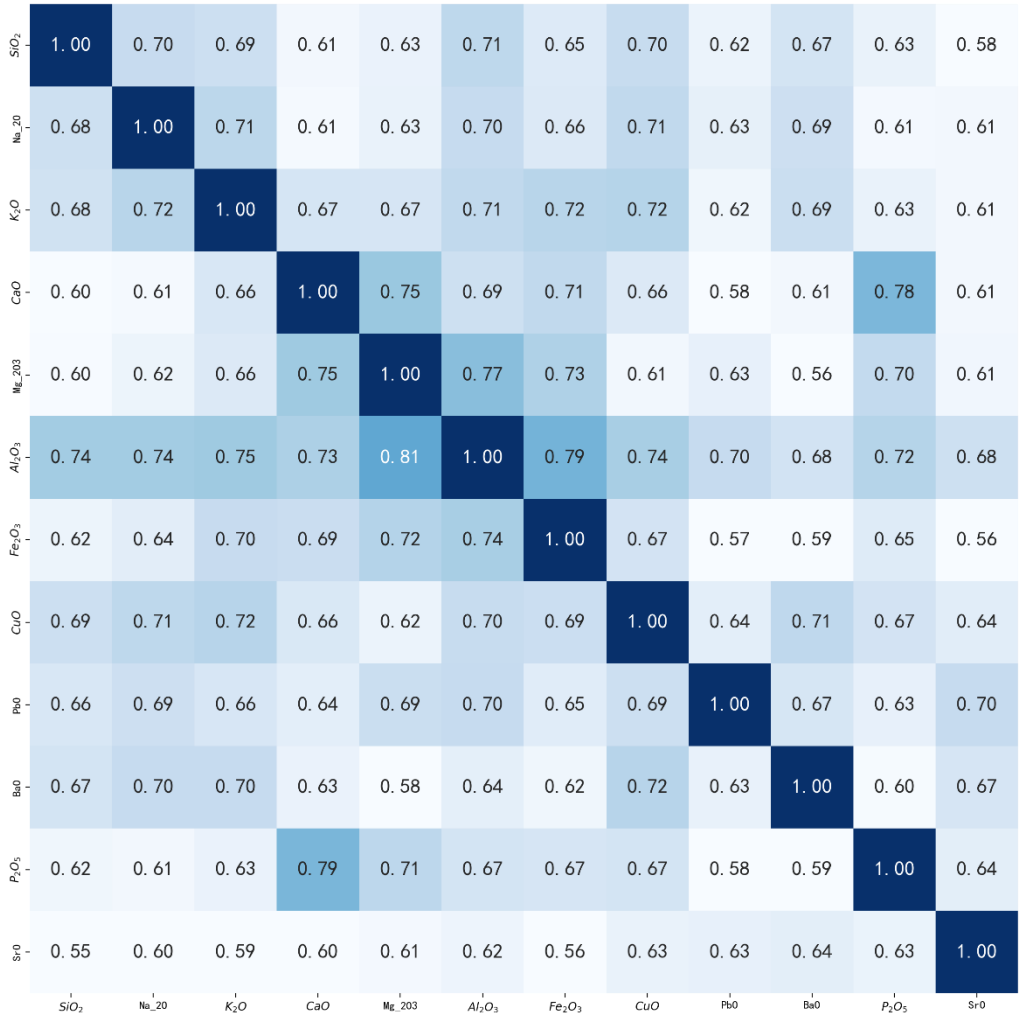
序号	规则前项	规则后项	支持度	置信度	提升度
1	CuO 分级 3	SiO2 分级 5	50%	100%	1
2	Fe2O3 分级 5	SiO2 分级 5	50%	100%	1

未风化铅钡玻璃热力图

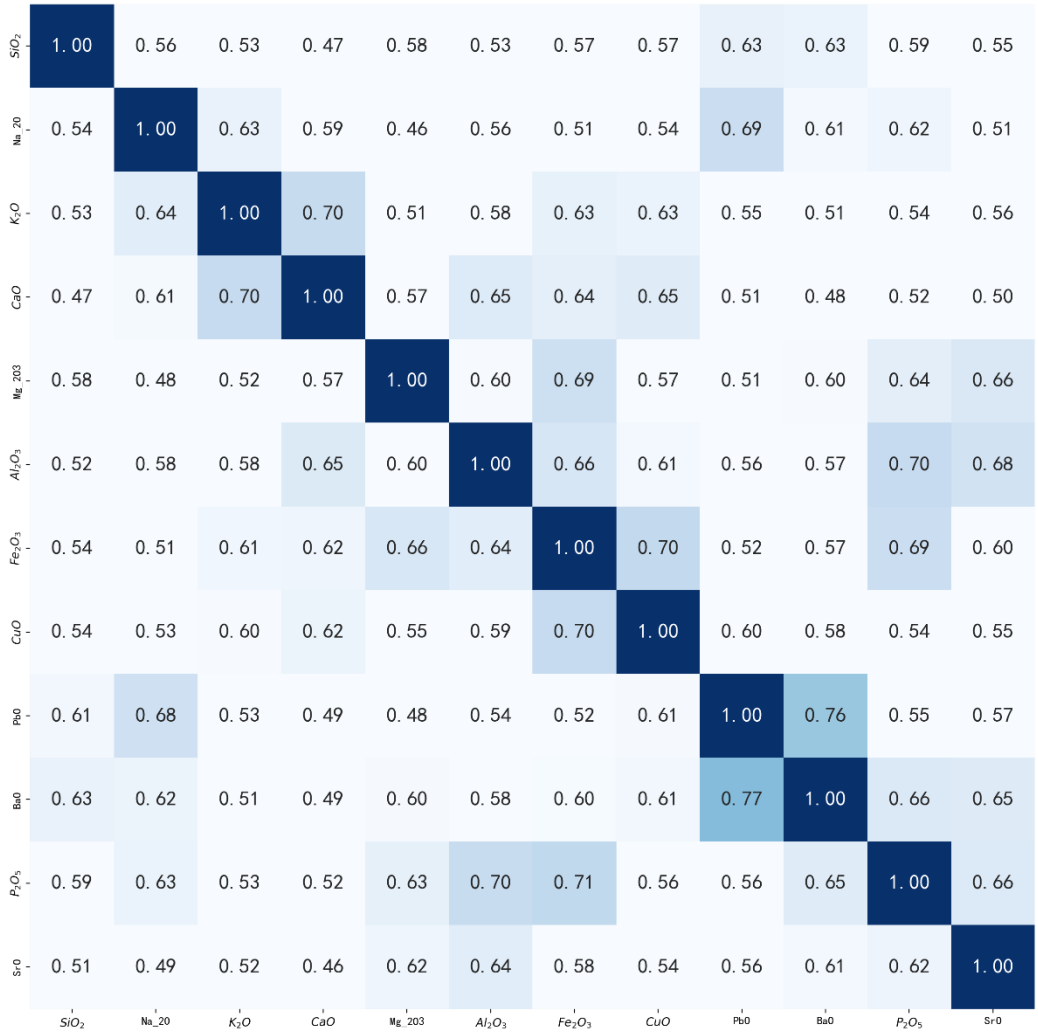

SiO2 1.00 0.63 0.65 0.66 0.63 0.63 0.59 0.63 0.64
Na2O 0.57 1.00 0.62 0.64 0.62 0.70 0.58 0.65 0.59
CaO 0.60 0.61 1.00 0.69 0.72 0.63 0.66 0.59 0.73
Al2O3 0.61 0.65 0.70 1.00 0.67 0.66 0.60 0.67 0.67
Fe2O3 0.58 0.62 0.72 0.66 1.00 0.63 0.67 0.59 0.71
CuO 0.63 0.73 0.67 0.69 0.68 1.00 0.65 0.80 0.75
PbO 0.54 0.58 0.66 0.59 0.67 0.60 1.00 0.63 0.62
BaO 0.61 0.68 0.62 0.69 0.62 0.78 0.66 1.00 0.71
P2O5 0.58 0.59 0.72 0.66 0.71 0.71 0.61 0.68 1.00
SiO2 Na2O CaO Al2O3 Fe2O3 CuO PbO BaO P2O5

风化高钾热力图

风化铅钡热力图



未风化高钾热力图



Python 代码

Question1-1

```
import pandas as pd
import numpy as np
data1 = pd.read_excel('data11.xlsx')
data2 = pd.read_excel('data2.xlsx')
data1.head()
data2.fillna(0,inplace=True)
data2['颜色'] = data1['颜色']
data2.to_excel('填充数据.xlsx')
data2_val = data2.iloc[:,range(1,len(data2.columns)-1)]
data2_sum = np.sum(data2_val.values,axis=1)
np.argwhere(data2_sum<85)
x data2.drop()
```

Question1-1-impute

```
import pandas as pd
from sklearn.impute import KNNImputer
import numpy as np
data = pd.read_excel('填充数据.xlsx')
data['颜色'] = data['颜色'].replace(0,np.nan)
from sklearn.preprocessing import minmax_scale
data1 = data.iloc[:,range(1,len(data.columns))].values[:,range(0,14)]
data2 = data.iloc[:,-1].values
data1 = minmax_scale(data1)
data2 = data2[:,np.newaxis]
tol_data = np.hstack([data1,data2])
tol_data = np.hstack([data1,data2])
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=6, weights="uniform")
tol_data = imputer.fit_transform(tol_data)
imputer_data = tol_data[:,-1]
imputer_data
```

Question1-2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pyplot import MultipleLocator
%matplotlib inline
import matplotlib as mpl
from matplotlib.font_manager import fontManager
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
data_p3 = pd.read_excel('3problem.xlsx')
def plot_pieplot(df, col,labels):
    colors = ['#1E79A6','#349ECE','#54B5E2','#AAE2F8']
    data = df[col].value_counts()

    colormap = ['firebrick','LightCoral']
    fig=plt.figure(figsize=(7,7),dpi=100)
    plt.pie(data, labels=labels, explode=[0.02,0.01,0, 0],
            autopct='%1.1f%%',startangle=90,colors=colors)
    plt.legend(fontsize=5)
    plt.show()
```

```
24.
25. labels = ['风化铅钡玻璃','未风化铅钡玻璃','风化高钾玻璃','未风化高钾玻璃']
26. plot_pieplot(data_p3,'类型',labels)
27. data_cat = []
28. for i in range(1,5):
29.     data_type_1 = data_p3[data_p3['类型'] == i]
30.     data_cat.append(data_type_1)
31.     data_type_1.to_excel('data_type_{}.xlsx'.format(i))
32. def plot_con_cat(df,j,cols=None):
33.     if cols == None:
34.         n = np.ceil(len(df.columns))
35.         data = df.copy()
36.         cols = df.columns.tolist()
37.     else:
38.         n = np.ceil(len(cols))
39.         data = df.loc[:,cols]
40.
41.
42.     plt.figure(figsize=(20,12),dpi=200)
43.     colors = ['#0f4c81',"#3f64a5",'#ff4700','#ffb599']
44.     species = ['风化铅钡玻璃','未风化铅钡玻璃','风化高钾玻璃','未风化高钾玻璃']
45.
46.     alphas = [1,0.5,0.7,0.5]
47.
48.     ax1 = plt.subplot()
49.     for i in range(len(df['类型'].unique())):
50.         sns.kdeplot(df[df['类型
'] == i+1][cols[j]], color=colors[i], shade=True, label=species[i],alpha=alphas[i])
51.
52.     plt.legend()
53.     ax1.set_ylabel('')
54.
55.     ax1.set_ylabel('')
56.     ax1.grid(which='both', axis='y', zorder=0, color='black', linestyle=':
', dashes=(2,7))
57.     for direction in ['top','right','left']:
58.         ax1.spines[direction].set_visible(False)
59.
60.     plt.legend()
61.     plt.show()
62. index = range(2,len(data_p3.columns))
63. for j in index:
64.     plot_con_cat(data_p3, j)
```

Question1-3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
data1 = pd.read_excel('data_type_1.xlsx')
data2 = pd.read_excel('data_type_2.xlsx')
def function1(mu1,mu2,sigma1,sigma2,x):
    return (mu1+x*sigma1)/(mu2+x*sigma2)
def plot_line(x,y1):
    plt.figure(figsize=(10,8),dpi=200)
    ax = plt.subplot()
    ax.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
    ax.plot(x, y1, 'b*--', label='y1',color='#9bb7d4')

    for direction in ['top','right','left']:
        ax.spines[direction].set_visible(False)

    plt.xlabel('x')
    plt.ylabel('value')
    plt.show()
def tol_function(data1,data2,col):

    mu1,sigma1 = data1[col].mean(), data1[col].std()
    mu2,sigma2 = data2[col].mean(), data2[col].std()

    max1,min1 = data1[col].max(), data1[col].min()
    max2,min2 = data2[col].max(), data2[col].min()
    x1,x2 = (min1-mu1)/sigma1,(max1-mu1)/sigma1
    y1,y2 = (min2-mu2)/sigma2,(max2-mu2)/sigma2

    imputer_1 = (mu1+min1)/(mu2+min2)
    imputer_2 = function1(mu1,mu2,sigma1,sigma2,-1)
    imputer_3 = mu1/mu2
    imputer_4 = function1(mu1,mu2,sigma1,sigma2,1)
    imputer_5 = (mu1+max1)/(mu2+max2)

    x = np.array([x1,-1,0,1,x2])
    y = np.array([imputer_1,imputer_2,imputer_3,imputer_4,imputer_5])

    plot_line(x,y)
```

```
cols = ['二氧化硅(SiO2)','氧化钠(Na2O)','氧化钙(CaO)','氧化铝(Al2O3)','氧化铁(Fe2O3)','氧化铜(CuO)',
        '氧化铅(PbO)','氧化钡(BaO)','五氧化二磷(P2O5)']
def tol_function2(data1,data2,col):

    mu1,sigma1 = data1[col].mean(), data1[col].std()
    mu2,sigma2 = data2[col].mean(), data2[col].std()

    max1,min1 = data1[col].max(), data1[col].min()
    max2,min2 = data2[col].max(), data2[col].min()
    x1,x2 = (min1-mu1)/sigma1,(max1-mu1)/sigma1
    y1,y2 = (min2-mu2)/sigma2,(max2-mu2)/sigma2

    r = (max1-min1)/(max2-min2)
    imputer_1 = (mu1+min1)/(mu2+min2)
    imputer_2 = function1(mu1,mu2,sigma1,sigma2,-1,r)
    imputer_3 = mu1/mu2
    imputer_4 = function1(mu1,mu2,sigma1,sigma2,1,r)
    imputer_5 = function1(mu1,mu2,sigma1,sigma2,1.5,r)
    imputer_6 = function1(mu1,mu2,sigma1,sigma2,2,r)
    imputer_7 = function1(mu1,mu2,sigma1,sigma2,2.5,r)
    imputer_8 = (mu1+max1)/(mu2+max2)

    x = np.array([x1,-1,0,1,1.5,2,2.5,x2])
    y = np.array([imputer_1,imputer_2,imputer_3,imputer_4,imputer_5,imputer_6,imputer_7,imputer_8])

    data1_val = data1[col].values
    m = (data1_val - mu1)/sigma1

    plot_line(x,y)
    result = np.array(Lagrange(y,x,m))

    index = np.argsort(m)
    plot_line(np.sort(m),result[index])

    return result
tol_function2(data1,data2,cols[1])

def tol_function4(data1,data2,col):

    mu1,sigma1 = data1[col].mean(), data1[col].std()
    mu2,sigma2 = data2[col].mean(), data2[col].std()
```

```
    83.
    84.     max1,min1 = data1[col].max(), data1[col].min()
    85.     max2,min2 = data2[col].max(), data2[col].min()
    86.     x1,x2 = (min1-mu1)/sigma1,(max1-mu1)/sigma1
    87.     y1,y2 = (min2-mu2)/sigma2,(max2-mu2)/sigma2
    88.
    89.     r = (max1-min1)/(max2-min2)
    90.     imputer_1 = (mu1+min1)/(mu2+min2)
    91.     imputer_2 = function1(mu1,mu2,sigma1,sigma2,-1.2)
    92.     imputer_3 = function1(mu1,mu2,sigma1,sigma2,-1)
    93.     imputer_4 = function1(mu1,mu2,sigma1,sigma2,-0.75)
    94.     imputer_5 = function1(mu1,mu2,sigma1,sigma2,-0.5)
    95.     imputer_6 = mu1/mu2
    96.     imputer_7 = function1(mu1,mu2,sigma1,sigma2,1)
    97.     imputer_8 = function1(mu1,mu2,sigma1,sigma2,1.5)
    98.     imputer_9 = function1(mu1,mu2,sigma1,sigma2,2)
    99.     imputer_10 = function1(mu1,mu2,sigma1,sigma2,2.5)
    100.    imputer_11 = (mu1+max1)/(mu2+max2)
    101.
    102.    x = np.array([x1,-1.2,-1,-0.75,-0.5,0,1,1.5,2,2.5,x2])
    103.    y = np.array([imputer_1,imputer_2,imputer_3,imputer_4,imputer_5,impute
r_6,imputer_7,imputer_8,imputer_9,imputer_10,imputer_11])
    104.
    105.    data1_val = data1[col].values
    106.    m = (data1_val - mu1)/sigma1
    107.
    108.    plot_line(x,y)
    109.    result = np.array(Lagrange(y,x,m))
    110.
    111.    index = np.argsort(m)
    112.    plot_line(np.sort(m),result[index])
    113.
    114.    return result
    115. tol_function4(data1,data2,cols[2])
```

Question2-1

```
    1.  import pandas as pd
    2.  import numpy as np
    3.  import matplotlib.pylab as plt
    4.  import seaborn as sns
    5.  import warnings
    6.  warnings.filterwarnings('ignore')
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def plot_con_cat(df,j,cols=None):
    if cols == None:
        n = np.ceil(len(df.columns))
        data = df.copy()
        cols = df.columns.tolist()
    else:
        n = np.ceil(len(cols))
        data = df.loc[:,cols]


    plt.figure(figsize=(20,12),dpi=200)
    colors = ['#0f4c81',"#3f64a5"]
    species = ['高钾玻璃','铅钡玻璃']
    alphas = [1,0.5]

    ax1 = plt.subplot()
    for i in range(len(df['类型'].unique())):
        sns.kdeplot(df[df['类型
'] == i][cols[j]], color=colors[i], shade=True, label=species[i],alpha=alphas[i])

    plt.legend()
    ax1.set_ylabel('')

    ax1.set_ylabel('')
    ax1.grid(which='both', axis='y', zorder=0, color='black', linestyle=':
', dashes=(2,7))
    for direction in ['top','right','left']:
        ax1.spines[direction].set_visible(False)

    plt.legend()
    plt.show()
data = pd.read_excel('3problem.xlsx')
index = range(2,len(data.columns))
for j in index:
    plot_con_cat(data, j)
```

Question 2-2-1

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pyplot import MultipleLocator
from sklearn.preprocessing import StandardScaler
%matplotlib inline
import matplotlib as mpl
from matplotlib.font_manager import fontManager
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
data_p3 = pd.read_excel('3problem.xlsx')
data_p3.head()
# 先取出来每类数据，然后PCA+聚类
group = data_p3.groupby("类型")
kk = group.get_group(0)
kk = kk.drop(['类型','表面风化'], axis = 1)
kk_cluster = kk.drop(['氧化铅(PbO)','氧化钡(BaO)','氧化锡(SnO2)'], axis = 1)
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import pandas as pd


# Remove the grain species from the DataFrame, save for later
varieties = list([0,2,3,4,5,6,7,8,11,12,14,15,16,17,18,21,22,28])

# Extract the measurements as a NumPy array

sc = StandardScaler()
norm_kk = sc.fit_transform(kk_cluster)

samples = norm_kk

"""
Perform hierarchical clustering on samples using the
linkage() function with the method='complete' keyword argument.
Assign the result to mergings.
"""
mergings = linkage(samples, method='complete')
plt.figure(figsize=(12,10),dpi=200)
ax1 = plt.subplot()
"""
Plot a dendrogram using the dendrogram() function on mergings,
```

```
    specifying the keyword arguments labels=varieties, leaf_rotation=90,
    and leaf_font_size=6.
    """
    dendrogram(mergings,labels=varieties,leaf_font_size=6,)

    ax1.grid(which='both', axis='y', zorder=0, color='black', linestyle=':', dashes=(2,7))
    for direction in ['top','right','left']:
        ax1.spines[direction].set_visible(False)

    plt.show()
```

Question2 和 Question3 的详细代码可以查看支撑材料中的“2022C 数学建模.ipynb”文件

Question4-1

```
from tqdm import tqdm
import time
import pandas as pd


def show_confidence(rule):
    index = 1
    for item in rule:
        s = " {:<4d}  {:.3f}        {}=>{}".format(index, item[2], str(list(item[0])), str(list(item[1])))
        index += 1
        print(s)


class Node:
    def __init__(self, node_name, count, parentNode):
        self.name = node_name
        self.count = count
        self.nodeLink = None  # 根据nideLink可以找到整棵树中所有nodename一样的节点
        self.parent = parentNode  # 父亲节点
        self.children = {}  # 子节点{节点名字:节点地址}

```

```
22.
23.  class Fp_growth_plus():
24.
25.      def data_compress(self, data_set):
26.          data_dic = {}
27.          for i in data_set:
28.              if frozenset(i) not in data_dic:
29.                  data_dic[frozenset(i)] = 1
30.              else:
31.                  data_dic[frozenset(i)] += 1
32.          return data_dic
33.
34.      def update_header(self, node, targetNode):  # 更新headertable 中的node
节点形成的链表
35.          while node.nodeLink != None:
36.              node = node.nodeLink
37.          node.nodeLink = targetNode
38.
39.      def update_fptree(self, items, count, node, headerTable):  # 用于更新
fptree
40.          if items[0] in node.children:
41.              # 判断items 的第一个结点是否已作为子结点
42.              node.children[items[0]].count += count
43.          else:
44.              # 创建新的分支
45.              node.children[items[0]] = Node(items[0], count, node)
46.              # 更新相应频繁项集的链表，往后添加
47.              if headerTable[items[0]][1] == None:
48.                  headerTable[items[0]][1] = node.children[items[0]]
49.              else:
50.                  self.update_header(headerTable[items[0]][1], node.children
[items[0]])
51.          # 递归
52.          if len(items) > 1:
53.              self.update_fptree(items[1:], count, node.children[items[0]],
headerTable)
54.
55.      def create_fptree(self, data_dic, min_support, flag=False):  # 建树主函
数
56.          item_count = {}  # 统计各项出现次数
57.          for t in data_dic:  # 第一次遍历，得到频繁一项集
58.              for item in t:
59.                  if item not in item_count:
```

```
60.                     item_count[item] = data_dic[t]
61.                 else:
62.                     item_count[item] += data_dic[t]
63.         headerTable = {}
64.         for k in item_count:  # 剔除不满足最小支持度的项
65.             if item_count[k] >= min_support:
66.                 headerTable[k] = item_count[k]
67.
68.         freqItemSet = set(headerTable.keys())  # 满足最小支持度的频繁项集
69.         if len(freqItemSet) == 0:
70.             return None, None
71.         for k in headerTable:
72.             headerTable[k] = [headerTable[k], None]  # element: [count, node]
73.         tree_header = Node('head node', 1, None)
74.         if flag:
75.             ite = tqdm(data_dic)
76.         else:
77.             ite = data_dic
78.         for t in ite:  # 第二次遍历，建树
79.             localD = {}
80.             for item in t:
81.                 if item in freqItemSet:  # 过滤，只取该样本中满足最小支持度的频繁项
82.                     localD[item] = headerTable[item][0]  # element : count
83.             if len(localD) > 0:
84.                 # 根据全局频数从大到小对单样本排序
85.                 order_item = [v[0] for v in sorted(localD.items(), key=lambda x: x[1], reverse=True)]
86.                 # 用过滤且排序后的样本更新树
87.                 self.update_fptree(order_item, data_dic[t], tree_header, headerTable)
88.         return tree_header, headerTable
89.
90.     def find_path(self, node, nodepath):
91.         '''
92.         递归将 node 的父节点添加到路径
93.         '''
94.         if node.parent != None:
95.             nodepath.append(node.parent.name)
96.             self.find_path(node.parent, nodepath)
97.
98.     def find_cond_pattern_base(self, node_name, headerTable):
```

```
        '''
        根据节点名字，找出所有条件模式基
        '''
        treeNode = headerTable[node_name][1]
        cond_pat_base = {}  # 保存所有条件模式基
        while treeNode != None:
            nodepath = []
            self.find_path(treeNode, nodepath)
            if len(nodepath) > 1:
                cond_pat_base[frozenset(nodepath[:-1])] = treeNode.count
            treeNode = treeNode.nodeLink
        return cond_pat_base

    def create_cond_fptree(self, headerTable, min_support, temp, freq_items, support_data):
        # 最开始的频繁项集是 headerTable 中的各元素
        freqs = [v[0] for v in sorted(headerTable.items(), key=lambda p: p[1][0])]  # 根据频繁项的总频次排序
        for freq in freqs:  # 对每个频繁项
            freq_set = temp.copy()
            freq_set.add(freq)
            freq_items.add(frozenset(freq_set))
            if frozenset(freq_set) not in support_data:  # 检查该频繁项是否在 support_data 中
                support_data[frozenset(freq_set)] = headerTable[freq][0]
            else:
                support_data[frozenset(freq_set)] += headerTable[freq][0]

            cond_pat_base = self.find_cond_pattern_base(freq, headerTable)  # 寻找到所有条件模式基
            # 创建条件模式树
            cond_tree, cur_headtable = self.create_fptree(cond_pat_base, min_support)
            if cur_headtable != None:
                self.create_cond_fptree(cur_headtable, min_support, freq_set, freq_items, support_data)  # 递归挖掘条件 FP 树

    def generate_L(self, data_set, min_support):
        data_dic = self.data_compress(data_set)
        freqItemSet = set()
        support_data = {}
        tree_header, headerTable = self.create_fptree(data_dic, min_support, flag=True)  # 创建数据集的 fptree
```

```
        # 创建各频繁一项的fptree，并挖掘频繁项并保存支持度计数
        self.create_cond_fptree(headerTable, min_support, set(), freqItemSet, support_data)

        max_l = 0
        for i in freqItemSet:  # 将频繁项根据大小保存到指定的容器L中
            if len(i) > max_l: max_l = len(i)
        L = [set() for _ in range(max_l)]
        for i in freqItemSet:
            L[len(i) - 1].add(i)
        for i in range(len(L)):
            print("frequent item {}:{}".format(i + 1, L[i]))
        return L, support_data

    def generate_R(self, data_set, min_support, min_conf):
        L, support_data = self.generate_L(data_set, min_support)
        rule_list = []
        sub_set_list = []
        for i in range(0, len(L)):
            for freq_set in L[i]:
                for sub_set in sub_set_list:
                    if sub_set.issubset(
                            freq_set) and freq_set - sub_set in support_data:  # and freq_set-sub_set in support_data
                        conf = support_data[freq_set] / support_data[freq_set - sub_set]
                        big_rule = (freq_set - sub_set, sub_set, conf)
                        if conf >= min_conf and big_rule not in rule_list:
                            rule_list.append(big_rule)
                sub_set_list.append(freq_set)
        rule_list = sorted(rule_list, key=lambda x: (x[2]), reverse=True)
        return rule_list


if __name__ == "__main__":
    begin_time = time.time()
    min_support = 2  # 最小支持度
    min_conf = 0.75  # 最小置信度
    data_set1 = pd.read_excel("Q4风化高钾分级结果.xlsx").astype(str)
    data_set1 = data_set1.drop(columns=['ttt'])
    data_set1 = data_set1.drop(columns=['二氧化硅(SiO2)分级'])

    for col in data_set1.columns:
```

```
        temp_data = data_set1[col].values
        temp_data = [col + i for i in temp_data]
        data_set1[col] = temp_data

    data_set = data_set1.values.tolist()
    for i in range(len(data_set)):
        count = 0
        for j in range(len(data_set[i])):
            if data_set[i][count] == (data_set1.columns[j] + str(0)):
                data_set[i].pop(count)
            else:
                count += 1
    fp = Fp_growth_plus()
    rule_list = fp.generate_R(data_set, min_support, min_conf)
    L , suport = fp.generate_L(data_set, 2)
    print("confidence:")
    show_confidence(rule_list)
    print(suport)
    end_time = time.time()
    timeget = end_time - begin_time
    print("程序开始时间:", begin_time)
    print("程序结束时间：", end_time)
    print("程序花费时间：", timeget)
```

Question4-2

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn import preprocessing
warnings.filterwarnings('ignore')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def plot_heatmap(data,low,high):
    plt.figure(figsize=(16, 16),dpi=200)
    ax = plt.subplot()
    sns.heatmap(data,square=True, cmap="Blues", annot=True, annot_kws={"fo
ntsize":18},fmt='0.2f',vmin=low,vmax=high,cbar=False)
    labels = ['$SiO_2$','Na_2O','$K_2O$','$CaO$','Mg_2O3','$Al_2O_3$','$Fe
_2O_3$',
```

```
                '$CuO$','PbO','BaO','$P_2O_5$','SrO']
        ax.set_yticks(np.arange(12)+0.5)
        ax.set_yticklabels(labels)

        ax.set_xticks(np.arange(12)+0.5)
        ax.set_xticklabels(labels)

        plt.show()


data00 = pd.read_excel("未风化高钾.xlsx")
data00 = data00.drop(columns=['文物采样点','是否风化','玻璃类别'])
data00 = data00.fillna(0)
data00.columns = ['二氧化硅','氧化钠','氧化钾','氧化钙','氧化镁','氧化铝','氧化铁','氧化铜','氧化铅','氧化钡','五氧化二磷','氧化锶','氧化锡','二氧化硫']
data00 = data00.drop(columns=['二氧化硫','氧化锡'])
count = 0
for i in range(len(data00)):
    if 85 <= data00.iloc[count,:].sum() <=105:
        count += 1
    else:
        data00 = data00.drop(index=count)
for column in data00.columns:
    if data00[column].sum()<0.1:
        data00 = data00.drop(columns=column)
data = np.array(data00.values)
p = 0.3
scaler = preprocessing.StandardScaler().fit(data)
data = scaler.transform(data)
MeanCors = np.zeros([len(data[0]),len(data[0])])
for i in range(len(data[0])):
    momCol = data[:,i].copy()
    sonCol = np.delete(data,i,axis=1)
    for col in range(sonCol.shape[1]):
        sonCol[:,col] = abs(sonCol[:,col]-momCol)
    minMin = sonCol.min()
    maxMax = sonCol.max()
    cors = (minMin+p*maxMax)/(sonCol+p*maxMax)
    meanCors = cors.mean(axis=0)
    meanCors = np.insert(meanCors,i,[1])
    MeanCors[i] = meanCors

plot_heatmap(MeanCors, 0.6,1)
```

描述性统计

风化铅钡玻璃描述性统计

变量名	平均值	标准差	中位数	峰度	偏度	变异系数（CV）
二氧化硅(SiO2)	24.913	10.605	25.015	1.23	0.313	0.426
氧化钠(Na2O)	0.216	0.557	0	6.743	2.666	2.575
氧化钾(K2O)	0.133	0.24	0	7.793	2.512	1.798
氧化钙(CaO)	2.695	1.66	2.875	-0.461	0.383	0.616
氧化镁(MgO)	0.65	0.706	0.57	1.209	1.037	1.087
氧化铝(Al2O3)	2.97	2.634	2.38	10.638	2.829	0.887
氧化铁(Fe2O3)	0.585	0.737	0.305	1.525	1.404	1.260
氧化铜(CuO)	2.276	2.821	1.145	4.154	2.101	1.239
氧化铅(PbO)	43.314	12.23	44.06	0.233	-0.033	0.282
氧化钡(BaO)	11.807	9.978	8.79	0.838	1.278	0.845
五氧化二磷(P2O5)	5.277	4.197	4.975	-0.824	0.4	0.795
氧化锶(SrO)	0.418	0.265	0.425	0.777	0.544	0.633
氧化锡(SnO2)	0.068	0.269	0	19.787	4.365	3.936
二氧化硫(SO2)	1.366	4.206	0	9.634	3.261	3.079

未风化铅钡玻璃描述性统计

变量名	平均值	标准差	中位数	峰度	偏度	变异系数（CV）
二氧化硅(SiO2)	54.66	11.829	54.61	-0.538	-0.371	0.216
氧化钠(Na2O)	1.683	2.372	0	0.758	1.287	1.409
氧化钾(K2O)	0.219	0.31	0.15	10.061	2.883	1.418
氧化钙(CaO)	1.32	1.285	0.84	1.03	1.347	0.973
氧化镁(MgO)	0.64	0.547	0.71	-1.241	0.085	0.854
氧化铝(Al2O3)	4.456	3.262	3.86	4.233	1.988	0.732
氧化铁(Fe2O3)	0.737	1.155	0	4.768	2.072	1.568
氧化铜(CuO)	1.432	1.97	0.65	6.877	2.461	1.376
氧化铅(PbO)	22.085	8.215	20.12	-0.456	0.62	0.372
氧化钡(BaO)	9.002	5.825	8.99	3.758	1.797	0.647
五氧化二磷(P2O5)	1.049	1.847	0.19	3.724	2.167	1.761
氧化锶(SrO)	0.268	0.243	0.26	1.96	1.231	0.908
氧化锡(SnO2)	0.047	0.127	0	5.816	2.634	2.737
二氧化硫(SO2)	0.159	0.763	0	23	4.796	4.796

风化高钾玻璃描述性统计

变量名	标准差	中位数	峰度	偏度	变异系数（CV）
二氧化硅(SiO2)	1.734	93.505	-0.388	0.854	0.018
氧化钠(Na2O)	0	0	0	0	0.000
氧化钾(K2O)	0.445	0.665	-1.913	-0.537	0.819
氧化钙(CaO)	0.488	0.83	0.988	0.504	0.561
氧化镁(MgO)	0.306	0	-1.598	1.014	1.558
氧化铝(Al2O3)	0.964	1.72	0.181	0.779	0.500
氧化铁(Fe2O3)	0.069	0.275	-1.418	-0.3	0.262
氧化铜(CuO)	0.935	1.545	2.231	1.218	0.599
氧化铅(PbO)	0	0	0	0	0.000
氧化钡(BaO)	0	0	0	0	0.000
五氧化二磷(P2O5)	0.21	0.28	0.372	0.399	0.750
氧化锶(SrO)	0	0	0	0	0.000
氧化锡(SnO2)	0	0	0	0	0.000
二氧化硫(SO2)	0	0	0	0	0.000

未风化高钾玻璃描述性统计

变量名	平均值	标准差	中位数	峰度	偏度	变异系数（CV）
二氧化硅(SiO2)	67.984	8.755	65.53	0.536	1.158	0.129
氧化钠(Na2O)	0.695	1.287	0	0.559	1.497	1.852
氧化钾(K2O)	9.331	3.92	9.83	1.9	-1.203	0.420
氧化钙(CaO)	5.333	3.092	6.095	-0.518	-0.875	0.580
氧化镁(MgO)	1.079	0.676	1.165	-1.015	-0.434	0.627
氧化铝(Al2O3)	6.62	2.492	6.185	-0.492	0.482	0.376
氧化铁(Fe2O3)	1.932	1.667	2.11	2.568	1.176	0.863
氧化铜(CuO)	2.452	1.66	2.345	-1.058	0.101	0.677
氧化铅(PbO)	0.412	0.589	0.155	0.418	1.374	1.431
氧化钡(BaO)	0.598	0.982	0	1.238	1.493	1.641
五氧化二磷(P2O5)	1.402	1.434	1.02	1.876	1.678	1.022
氧化锶(SrO)	0.042	0.048	0.02	-1.452	0.571	1.162
氧化锡(SnO2)	0.197	0.681	0	12	3.464	3.464
二氧化硫(SO2)	0.102	0.186	0	0.055	1.396	1.825