基于多种统计方法与机器学习对古代玻璃文物的研究

摘　要

丝绸之路对我们具有重要的历史意义，作为丝绸之路上的物证之一的古代玻璃值得我们的研究。我们通过构建模型让我们对高钾玻璃与铅钡玻璃的认识更加深入，也对风化前后的变化及特征有了更多了解，也能够在一定程度上预测风化后的样本在风化前原本的化学成分。在玻璃文物鉴别上能够有所作用。

问题一中，对于第 1 小问，我们通过使用卡方检验，发现纹饰与表面风化的关系较弱，颜色和 0 玻璃类型与表面类型是中等相关；对于第 2 小问，我们分别对高钾玻璃与铅钡玻璃进行了相关特征值的分析，发现氧化硅的变化量最为明显，其中高钾玻璃中 Na 与 K 流失严重，而 Si，Al，Fe，Cu 等元素有所增加，铅钡玻璃而言的 Pb 含量明显增加，Ba 的含量显著降低，P 含量增加。并且基于化学机理做出了一定的解释；对于第 3 小问，我们采用了分布线性回归的方法进行预测，具体预测结果见文章中相关内容。

问题二中，对于第 1 小问，我们用 CRITIC 权重法分析了风化前后的玻璃文物的数据，找到了一些重要的化学元素，总结出来高钾玻璃主要是 K_2O、CaO、CuO 含量较为丰富，在无风化条件下前两者含量更多，风化条件下后者更多，但总体都位于前列，铅钡玻璃则是 PbO 和 BaO 含量非常丰富，尤其是在风化后 PbO 甚至含量超过了 SiO_2；对于第 2 小问，我们利用前 1 小问已经得到的较为重要的几个化学元素，应用了 Kmeans++等聚类算法对这些样本进行了亚类的划分，具体结果见附录。

问题三中，我们利用了包括 SVM、logistic 回归、RF 等多种分类模型对已有的数据进行了学习，之后将学习所得参数代入未知的附表 3 的玻璃文物进行鉴定，结果非常一致，具体结果见文中相关内容。

问题四中，我们运用了灰色关联度分析的模型得到表格，得到了具体的化学元素之间的关联关系，同时发现铅钡玻璃风化后化学成分关联关系较强，而高钾玻璃风化后化学成分关联关系相对减弱。并且做出了相应的机理解释。

本文通过数据挖掘结合机理分析，充分考虑了化学产品的相关性质，通过挑选重要特征数据，过滤无关数据，提出了可靠模型。

关键词：卡方检验 CRTIC 权重法 多步线性回归 SVM RF 灰色关联度分析

一、 问题重述

丝绸之路是一条具有深远意义的国际通道，是连接中国和西方世界的第一座桥梁。通过这条古道，把古老的中国文化、印度文化、波斯文化、阿拉伯文化和古希腊、古罗马文化联系起来，促进了东西方的文化交流。早期的玻璃就从西亚和埃及地区以饰品的形式传入中国，中国古代人民在吸取工艺经验后制作出了中国本土化玻璃。

玻璃的主要成分是石英砂，主要化学成分是二氧化硅（SiO_2），在玻璃制作过程中常常需要加入助熔剂。古时候人们常用的古时候人们常用的有草木灰（主要为 K_2CO_3）、天然泡碱（主要成分为 Na_2CO_3）、硝石和铅矿石等，并且选择石灰石作为稳定剂。加入的助熔剂不同，会有不同的分类。烧制过程中加入铅矿石作为助熔剂，其氧化铅（PbO）、氧化钡（BaO）的含量较高，便烧制成为铅钡玻璃（主要成分为硅酸铅），加入草木灰作为助熔剂会使得钾含量变高，成为高钾玻璃（K_2O-CaO-SiO_2 与 K_2O- SiO_2）。

流传至今的古代玻璃非常容易被环境影响导致风化。在这一过程中，往往会导致元素的交换，进而造成化学成分比例发生变化，影响我们对其种类的判定。

现在有一批我国古代玻璃制品的相关数据，并且已经被分成了高钾玻璃与铅钡玻璃两种类型。需要解决以下问题：

问题 1 对这些古代玻璃的表面风化与其玻璃类型、纹饰和颜色的关系进行分析；结合玻璃的类型，分析玻璃样品表面有无风化化学成分含量的统计规律；根据风化点检测数据，预测其风化前的化学成分含量。

问题 2 依据附件数据分析两种玻璃的分类规律；对于两种不同类别分别选择合适的化学成分对其进行亚类划分；对亚类分类结果的合理性和敏感性进行分析。

问题 3 对附件表单 3 中未知类别玻璃文物的化学成分进行分析，鉴别其所属类型；对分类结果的敏感性进行分析。

问题 4 针对不同类别的玻璃文物样品，分析其化学成分之间的关联关系；比较两种不同类别之间的化学成分关联关系的差异性。

二、 问题分析

2.1 问题一的分析

第 1 小问：该小问需要我们分析玻璃的表面风化与其三个特征——玻璃类型、纹饰和颜色是否存在某种关系。观察到这三个特征都是无序类型变量，所以选择使用卡方检验来探测这三个特征分别与表面风化的关系。

第 2 小问：该小问需要我们结合玻璃类型，分析玻璃样品表面化学成分含量与其有无风化的统计规律。分别讨论铅钡玻璃与高钾玻璃风化前后的变化差异进行描述性统计分析。

第 3 小问：该小问需要我们结合风化点检测数据，预测其风化前的化学成分含量。我们这里通过分别考察风化前后 SiO_2 的含量变化和其他元素相较于 SiO_2 的改变，最后通过先预测 SiO_2 元素含量改变，再预测其余元素相较于 SiO_2 元素的相对改变量得出占比结果。

2.2 问题二的分析

第 1 小问：我们通过对风化前后的成分分别使用 CRITIC 方法确定出较为重要的化学成分，之后再通过这些重要的化学成分分析确定两种玻璃的分类规律。

第 2 小问：我们考虑继续通过 CRITIC 方法对高钾玻璃和铅钡玻璃风化前后的化

学元素进行分析，同时通过分箱法筛出异常数据，选取重要的 6 大指标化学元素作为高钾和铅钡玻璃的敏感划分元素。且通过等方法，在选取全部元素和 6 大指标元素的情况下，分别在余弦度量和欧式度量下进行聚类。得到了高钾玻璃和铅钡玻璃的亚类划分，并根据其颜色和纹理，化学机理进行合理性分析。并在此基础上进行扰动处理后给出灵敏性分析的结果。

2.3 问题三的分析

这一问需要我们对之前的已知类别的玻璃文物进行总结，找到一个合适的分类方法，对表单 3 中不同化学成分的玻璃文物进行分类。为了保证结果的准确性，我们选择分别用支持向量机(SVM)，贝叶斯分类器(bayes)以及决策树分别对原数据进行划分，得到分界线，最后使用 SVM 与决策树投票来得到对表单 3 的分类结果。

2.4 问题四的分析

这一问要求我们对不同类别的玻璃文物样品，分析其化学成分之间的关联关系并且比较两种不同类别之间的化学成分关联关系的差异性。我们采用了灰色关联分析法，选择占比较大的 SiO2 作为分析的因变量（母序列），其余变量作为自变量（子序列）建立灰色关联分析模型，计算其灰色关联度。将不同类别的玻璃中所计算出的灰色关联系数进行方差分析，通过进行显著性检验观察高钾玻璃与铅钡玻璃之间的差异性。

三、 模型假设

1. 假设除严重风化点的玻璃外其余玻璃风化程度均匀性较好
2. 假设玻璃制品形状对风化严重程度影响小
3. 假设颜色纹饰对风化程度影响小
4. 假设选择合适指标时只考虑了显著性 p 值的影响
5. 假设经筛选后测量结果的数据真实可信
6. 假设古代玻璃制品的成分均匀性优良

四、 符号说明

由于本文有着大量的推导过程，其中存在一些常见字母如x,y等反复使用与不同定义，因此符号含义都写在了文章中相应公式的旁边。

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 第 1 小问模型的建立

该问要求我们分析表面风化与玻璃类型、纹饰和颜色之间的关系。注意到玻璃类型、纹饰与颜色都是无序分类变量，因此我们考虑建立表格对这些分类变量进行卡方检验。

卡方检验是一种用途广泛的假设检验方法，它用于统计样本的实际观测值与理论推断值之间的偏离程度，实际观测值与理论推断值之间的偏离程度就决定卡方值的大小，如果卡方值越大，二者偏差程度越大；反之，二者偏差越小；若两个值完全相等时，卡方值就为 0，表明理论值完全符合。

5.1.2 第 1 小问模型的求解

题目中的分类变量有X{风化,无风化},Y_1{A,B,C},Y_2{高钾,铅钡},Y_3{蓝绿,浅蓝,紫,深绿,深蓝,黑,浅绿,绿}。

(1)我们分别假设X与Y_1,Y_2,Y_3存在联系。

(2)我们把属于风化的Y_i的第j项的个数记作f_{ij},例如f_{33}就表示了属于风化的紫色数量的玻璃文物有多少个。显然我们有$f_{i1}+f_{i2}+f_{i3}+...=n$,其中$n$代表了样本容量的个数。

(3)我们可以计算出总体X的值落入Y_{ij}的概率p_{ij}。

(4)我们引入检验统计量

$$\chi_i^2=\sum_{j=1}^{k}\frac{(f_{ij}-np_{ij})^2}{np_{ij}} \tag{1}$$

其中χ_i^2是用于检验X与Y_i之间关系紧密程度的统计量，并且χ_i^2的值越大，说明“X与Y_i有关系”成立的可能性越大。

用 SPSS 软件我们可以得到如下表所示的结果：

表 1 表面风化卡方检验表

题目	名称	表面风化		总计	χ^2	P
		风化	无风化			
纹饰	A	11	11	22	4.957	0.084
	B	6	0	6		
	C	17	13	30		
合计		34	24	58		
玻璃类型	高钾	6	12	18	6.880	0.009
	铅钡	28	12	40		
合计		34	24	58		
颜色	蓝绿	9	6	15	9.432	0.307
	浅蓝	12	8	20		
	紫	2	2	4		
	深绿	4	3	7		
	深蓝	0	2	2		
	黑	2	0	2		
	浅绿	1	2	3		
	绿	0	1	1		
	不详	4	0	4		
合计		34	24	58		

由上表 1 表面风化卡方检验表可得出，表面风化和纹饰，显著性 P 值为 0.084，接受原假设，因此不存在显著性差异；表面风化和玻璃类型，显著性 P 值为 0.009，拒绝原假设，存在显著性差异；表面风化和颜色，显著性为 0.307，接受原假设，不存在显著性差异。

在此基础上进行效应量化分析，包括 phi、Crammer’sV、列联系数，用于分析表面风化与其余三个变量的量化程度，这三个指标含义如下：

(1) **phi 系数**：表示两因素之间的关联程度。当 phi 值小于 0.3 时，表示相关较弱；当 phi 值大于 0.6 时，表示相关较强；当 phi 值位于两者之间时，表示相关中等。

(2) **Crammer'sV**：是 phi 系数的修正值。

(3) **列联系数**：简称 c 系数，当列联表中两个变量相互独立的时候，系数 c=0，但它不可能大于 1，其可能的最大值依赖于列联表的行数和列数，且随着行数列数的增大而增大。

使用 SPSS 进行操作得到结果如下图所示：

表 2 表面风化效应量化分析表

分析项	phi	Crammer's V	列联系数
纹饰	0.292	0.292	0.281
类型	0.344	0.344	0.326
颜色	0.403	0.403	0.374

由上表 2 表面风化效应量化分析表可得出，纹饰的 phi 值小于 0.3 说明与表面风化的相关性较弱；而玻璃类型和颜色的 phi 值都位于 0.3~0.6 之间，说明相关程度为中等。

5.1.3 第 2 小问模型的建立

首先我们注意到题目中说到“本题中将成分比例累加和介于 85%~105%之间的数据视为有效数据”，因此在解决问题之前，我们需要将附件表单 2 中不合格的数据剔除出去。

在这里我们利用 python 先读入表单的相应数据，再一一计算总的成分含量是否低于 85%或者高于 105%，如果满足则将此数据删除。通过计算我们将文物采样点 15 与 17 这两行数据进行了删除。同样的各成分含量之和应为 100%，但因检测手段等原因加和并非 100%，所以对各成分进行归一化处理，使其成分含量和为 100%。

然后我们将风化和严重风化采样点的产品记为一类，作为风化后产品，未风化采样点产品和未风化产品记为一类，然后对于高钾，铅钡玻璃风化前后化学成分含量进行统计分析。

5.1.4 第 2 小问模型的求解

首先我们对附表 2 中的各类元素进行初步分析，求其最大值等统计相关量，并且将风化前后进行对比，其中高钾玻璃的相关分析结果如下表 3：

表 3 高钾玻璃风化前后统计相关量

变量		样本量	最大值	最小值	平均值	标准差	中位数	方差	峰度	偏度
SiO_2	风化后	6	96.954	92.35	94.331	1.675	94.267	2.806	-0.241	0.515
	风化前	12	87.05	59.01	67.984	8.755	65.53	76.652	0.536	1.158
Na_2O	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	3.38	0	0.695	1.287	0	1.656	0.559	1.497
K_2O	风化后	6	1.014	0	0.545	0.446	0.666	0.199	-1.909	-0.533
	风化前	12	14.52	0	9.331	3.92	9.83	15.369	1.9	-1.203
CaO	风华后	6	1.66	0.21	0.873	0.488	0.837	0.238	0.942	0.477
	风化前	12	8.7	0	5.333	3.092	6.095	9.563	-0.518	-0.875
MgO	风化后	6	0.64	0	0.198	0.308	0	0.095	-1.635	1.008
	风化前	12	1.98	0	1.079	0.676	1.165	0.457	-1.015	-0.434

Al_2O_3	风化后	6	3.5	0.812	1.938	0.967	1.726	0.935	0.068	0.752
	风化前	12	11.15	3.05	6.62	2.492	6.185	6.208	-0.492	0.482
Fe_2O_3	风化后	6	0.35	0.171	0.266	0.069	0.276	0.005	-1.386	-0.316
	风化前	12	6.04	0	1.932	1.667	2.11	2.778	2.568	1.176
CuO	风化后	6	3.25	0.55	1.568	0.938	1.556	0.88	2.196	1.204
	风化前	12	5.09	0	2.452	1.66	2.345	2.756	-1.058	0.101
PbO	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	1.62	0	0.412	0.589	0.155	0.347	0.418	1.374
BaO	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	2.86	0	0.598	0.982	0	0.965	1.238	1.493
P_2O_5	风化后	6	0.612	0	0.281	0.211	0.28	0.044	0.334	0.39
	风化前	12	4.5	0	1.402	1.434	1.02	2.056	1.876	1.678
SrO	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	0.12	0	0.042	0.048	0.02	0.002	-1.452	0.571
SnO_2	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	2.36	0	0.197	0.681	0	0.464	12	3.464
SO_2	风化后	6	0	0	0	0	0	0	0	0
	风化前	12	0.47	0	0.102	0.186	0	0.034	0.055	1.396
总含量	风化后	6	100	98.81	99.61	0.417	99.735	0.173	3.869	-1.827
	风化前	12	100	96.06	98.178	1.132	98.465	1.282	-0.107	-0.559

我们可以看到对于铅钡玻璃与高钾玻璃风化前后其氧化硅的含量变化是较为明显的。而在学界对于类似问题的研究也同样呈现相同的趋势。

对于高钾玻璃而言，可以发现 Na 元素和 K 元素的流失很严重。机理分析是水解氢化反应的结果。与之相反，Si，Al，Fe，Cu 等元素在风化侵蚀后的浓度是增加的，考虑机理分析，可能是由于风化表面形成了富硅层，吸附了土壤中的 Al，Fe 等离子的原因。

而对于铅钡玻璃而言，Pb 的含量明显增加，Ba 的含量显著降低，P 含量增加。结合机理分析，由于水解氢化反应，玻璃表面会析出碳酸铅，$Pb_5(PO_4)_3OH$ 和 $5PbO \cdot P_2O_5 \cdot SiO_2$ 晶体。

5.1.5 第 3 小问模型的建立

考虑到硅氧四面体的化学稳定性较高，不容易被腐蚀，并且古代玻璃体系以硅氧四面体作为主要成分，故我们可以以 SiO_2 作为预测基准，分别考察风化前后 SiO_2 的含量变化和其他元素相较于 SiO_2 的改变，最后通过先预测 SiO_2 元素含量改变，再预测其余元素相较于 SiO_2 元素的相对改变量得出占比结果，并且由于各成分含量之和应为 100%，所以我们对预测的结果进行归一化处理，使其成分含量和为 100%，得到最终结果。

根据前文分析可以得知，Si 在古代玻璃的分类中不起主导地位的，我们需要更多地去关注其余化学元素的占比变化。考虑到 SiO_2 的数据不稀疏，且 SiO_2 本身化学性质稳定，我们可以考虑分别建立 SiO_2 在风化前后占比含量变化的映射和其他化学成分相较于 SiO_2 而言的化学成分含量的变化映射。

我们先关注严重风化点的玻璃，分别是 08 号铅钡文物，26 号铅钡文物，54 号铅钡文物。根据观察我们可以发现他们的 SiO_2 占比都显著低于均值，PbO 占比都显著高于均值。为了使得风化模型回归效果更好，我们选择在根据铅钡玻璃寻找风化前后复原时去掉严重风化点的数据。（高钾类玻璃不受此影响）

通过散点图图 1，图 2（横坐标为 SiO_2 的百分占比，纵坐标为 0，1）不难发现其具有良好的线性关系，我们不妨使用线性回归（最小二乘法）对 SiO_2 在风化前后占比含量变化建立线性映射。

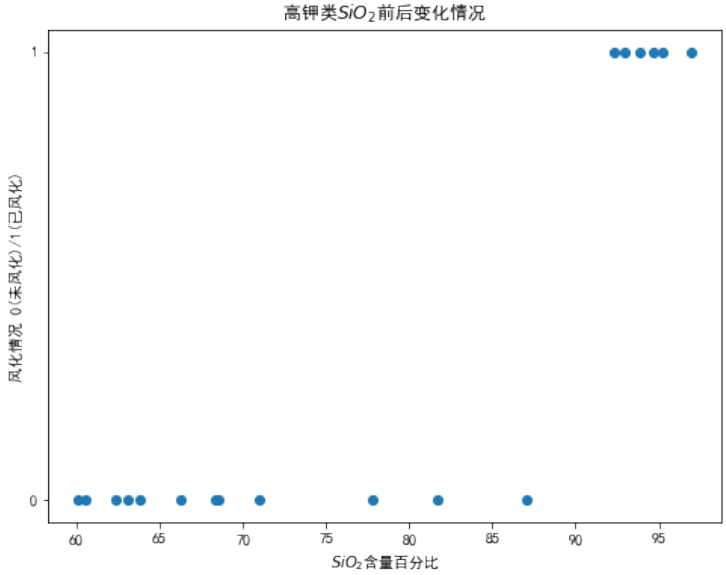

图 1 高钾类 SiO_2 前后变化情况

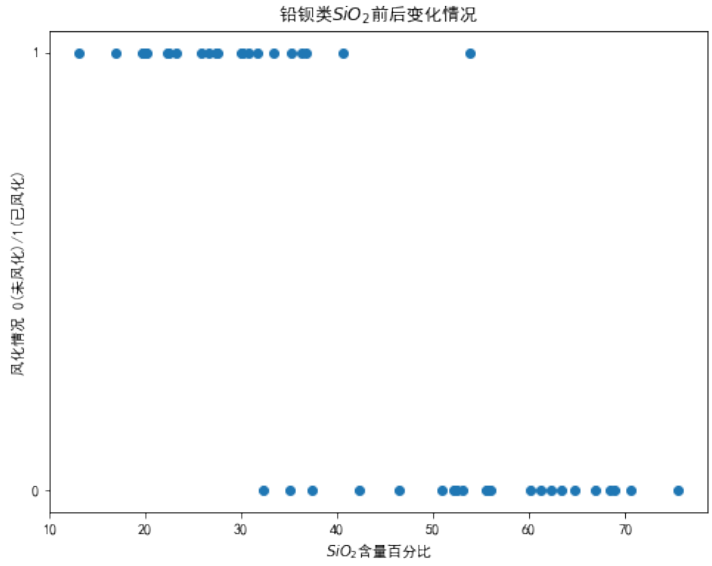

图 2 铅钡类 SiO_2 前后变化情况

5.1.6 第 3 小问模型的求解

高钾的线性回归分析结果表 4：

表 4 线性回归分析结果表

线性回归分析结果 n=18									
	非标准化系数		标准化系数	t	p	VIF	R^2	调整 R^2	F
	B	标准误	Beta						
常数	-1.987	0.342	-	-5.811	0.000***	-			
二氧化硅(SiO2)	0.03	0.004	0.865	6.89	0.000***	1.000	0.748	0.732	F=47.477 P=0.000***
因变量：风化情况									

注：***、**、*分别代表 1%、5%、10%的显著性水平

从 F 检验的结果分析可以得到，显著性 P 值为 0.000***，水平上呈现显著性，拒绝回归系数为 0 的原假设， 因此模型基本满足要求 对于变量共线性表现，VIF 全部小于 10，因此模型没有多重共线性问题，模型构建良好 模型的公式如下:y=-1.987+0.03*二氧化硅(SiO2)

铅钡同理

之后我们构建其他化学成分相较于 SiO2 的改变映射。为了排除部分矿物的偶然性，我们先进行相关性的验证：

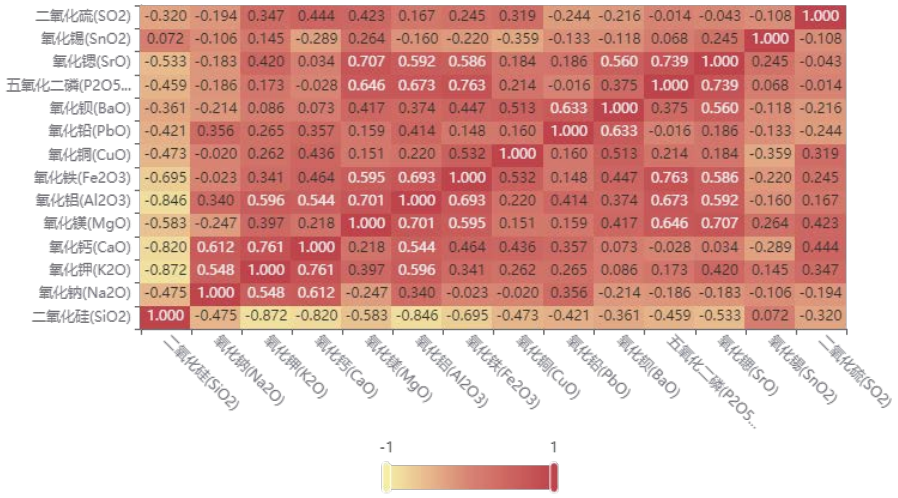

图 3 化学成分相关性验证

挑选其中与 SiO_2 相关系数>0.3 的指标（即除了 SO_2 和 SnO_2 之外的指标）作为回归的因变量，SiO_2 作为自变量，进行线性回归（最小二乘）。

然后最后通过相关系数进一步证实第二小问得到 K，Ca，Fe，Pb，P 这几个元素是风化中变化较大的元素。

最后我们通过回归方程求出原先风化前的百分比。

结果展示在附录中

5.2 问题二模型的建立与求解

5.2.1 第 1 小问模型的建立

这一问需要我们对两种玻璃的分类规律进行分析，我们考虑使用 CRTIC 权重法先对风化前后的数据进行分析。然后根据权重法得到的较为重要的元素来对其分类规律进行解析。

CRITIC 权重法是一种基于数据波动性的客观赋权法。其思想在于两项指标，分别是波动性（对比强度）和冲突性（相关性）指标。对比强度使用标准差进行表示，如果数据标准差越大说明波动越大，权重会越高；冲突性使用相关系数进行表示，如果指标之间的相关系数值越大，说明冲突性越小，那么其权重也就越低。权重计算时，对比强度与冲突性指标相乘，并且进行归一化处理，即得到最终的权重。

5.2.2 第 1 小问模型的求解

以高钾为例，假设其中有效的数据有m项，有n个化学元素作为评价指标，

1.就构成了原始数据矩阵

$$X=\begin{pmatrix} x_{11}\cdots x_{1n} \\ \vdots \ddots \vdots \\ x_{m1}\cdots x_{mn} \end{pmatrix} \tag{2}$$

2.接下来我们需要数据标准化

对于正向指标有：

$$x_{ij}^{'}=\frac{x_{ij}-\min(x_j)}{\max(x_j)-\min(x_j)} \tag{3}$$

对于逆向指标有：

$$x_{ij}^{'}=\frac{\max(x_j)-x_{ij}}{\max(x_j)-\min(x_j)} \tag{4}$$

3.计算信息承载量

波动性：

$$S_j=\sqrt{\frac{\Sigma_{i=1}^{m}(x_{ij}-\overline{x}_j)^2}{n-1}} \tag{5}$$

冲突性：

$$R=\frac{\Sigma_{j,k=1}^{n}(x_{ij}-\overline{x}_j)(x_{ik}-\overline{x}_k)}{\sqrt{\Sigma_{j=1}^{n}(x_{ij}-\overline{x}_j)^2\Sigma_{k=1}^{n}(x_{ik}-\overline{x}_k)^2}} \tag{6}$$

又有

$$A_j=\sum_{i=1}^{n}(1-r_{ij}) \tag{7}$$

其中r_{ij}即表示矩阵R的i行j列。

信息量：

$$C_j=S_j\times A_j \tag{8}$$

4.计算权重

$$W_j=\frac{C_j}{\Sigma_{j=1}^{n}C_j} \tag{9}$$

我们将数据代入可得到如下表 5 表 6 结果

表 5 风化玻璃文物的权重

项	指标变异性	指标冲突性	信息量	权重
二氧化硅(SiO2)	29	11.165	323.797	0.201
氧化钠(Na2O)	0.852	13.634	11.61	0.007
氧化钾(K2O)	6.599	11.97	78.994	0.049
氧化钙(CaO)	5.039	11.227	56.572	0.035
氧化镁(MgO)	2.164	11.996	25.955	0.016
氧化铝(Al2O3)	12.719	10.715	136.293	0.085
氧化铁(Fe2O3)	2.173	11.287	24.524	0.015
氧化铜(CuO)	11.674	11.973	139.769	0.087
氧化铅(PbO)	28.155	17.088	481.098	0.299
氧化钡(BaO)	12.186	15.414	187.841	0.117
五氧化二磷(P2O5)	5.471	13.934	76.227	0.047
氧化锶(SrO)	0.368	15.18	5.582	0.003
氧化锡(SnO2)	0.515	12.494	6.434	0.004
二氧化硫(SO2)	4.029	13.976	56.315	0.035

表 6 无风化玻璃文物的权重

项	指标变异性	指标冲突性	信息量	权重
二氧化硅(SiO2)	12.51	12.19	152.507	0.101
氧化钠(Na2O)	4.899	14.495	71.009	0.047
氧化钾(K2O)	16.692	12.067	201.414	0.133
氧化钙(CaO)	8.548	12.659	108.207	0.071
氧化镁(MgO)	2.161	11.633	25.138	0.017
氧化铝(Al2O3)	9.865	12.083	119.203	0.079

氧化铁(Fe2O3)	4.122	12.355	50.923	0.034
氧化铜(CuO)	4.574	12.393	56.678	0.037
氧化铅(PbO)	26.613	16.66	443.392	0.292
氧化钡(BaO)	11.734	15.6	183.049	0.121
五氧化二磷(P2O5)	4.012	12.527	50.259	0.033
氧化锶(SrO)	0.445	14.183	6.305	0.004
氧化锡(SnO2)	2.241	12.709	28.478	0.019
二氧化硫(SO2)	1.577	13.115	20.681	0.014

这两表分别得到对应图 4 图 5 的权重排序图

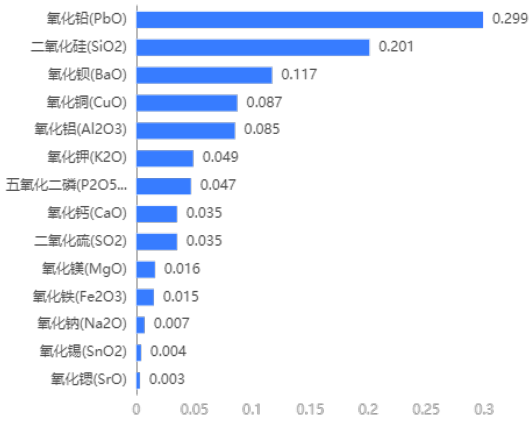

图 4 风化元素权重排序图

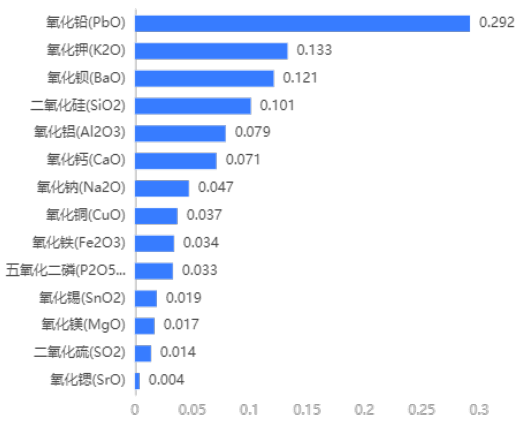

图 5 无风化元素权重排序图

5. 2. 3 第 2 小问模型的建立

这一问需要我们对高钾玻璃与铅钡玻璃进行亚类上的划分，我们考虑继续使用 CRTIC 权重法分别对高钾风化前后与铅钡风化前后的数据进行分析，再这一步分别确定对高钾玻璃和铅钡玻璃来说较为重要的化学成分组成，再依据这些化学成分组成通过 Kmeans++，DBSCAN，T-SNE 聚类算法进行亚类上的划分。

5. 2. 4 第 2 小问模型的求解

这里我们就可以通过选择亚类对应的最重要几个变量进行聚类划分了，然后可视化的坐标可以选择相关性强的两两组合来便于显示分类效果好。距离可以余弦和欧式都用，最后再讨论数据波动后划分结果变化。

我们分别将高钾玻璃风化前后与铅钡玻璃风化前后的数据进行 CRTIC 权重法分析，得到如下图 6 图 7 图 8 图 9 的元素权重排序图

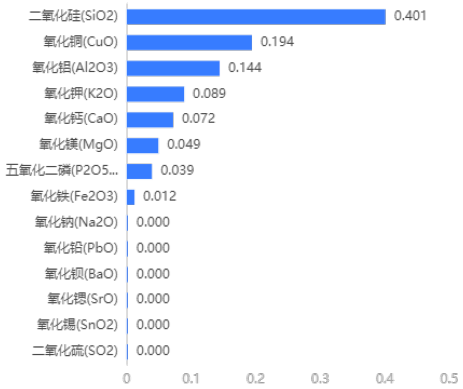

图 6 风化高钾玻璃元素权重排序图

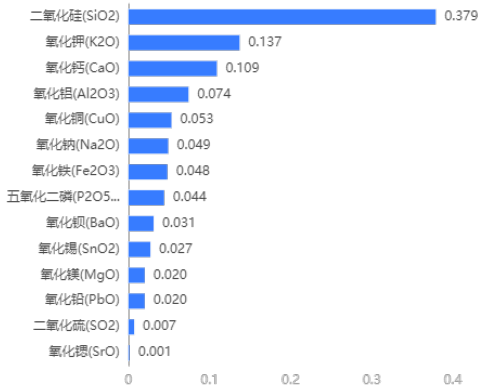

图 7 无风化高钾玻璃元素权重排序图

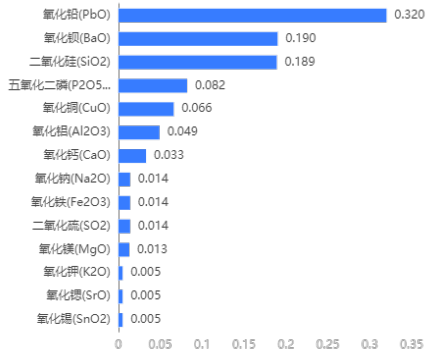

图 8 风化铅钡玻璃元素权重排序图

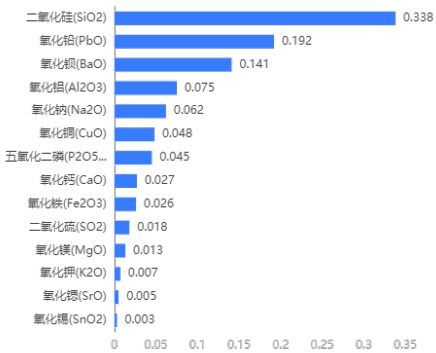

图 9 无风化铅钡玻璃元素权重排序图

通过观察上表，我们可以知道对高钾来说重要的元素有 K，Ca，Cu，Fe，Al，Mg，对铅钡来说重要的元素是 Ca，Al，Cu，P，Ba，Pb。我们将风化前后的高钾玻璃与铅钡玻璃都视为不同的类型，在第一次分类的情况下再对相应类型的玻璃进行进一步的聚类分析，如下表 7 给出的未风化高钾玻璃的再次聚类。

表 7 未风化高钾玻璃亚类分类表

样本类别	样本编号
第 1 类	1、4
第 2 类	3、5、6、13、14、15、16、17、18、21

其中聚类中心点为

表 8 聚类中心点

类种类	中心值_二氧化硅(SiO2)	中心值_氧化钠(Na2O)	中心值_氧化钾(K2O)	中心值_氧化钙(CaO)	中心值_氧化铝(Al2O3)	中心值_氧化铁(Fe2O3)
	65.00486111040446	0.940252787881922	11.04872015740095	6.509479395624389	7.495866328313897	2.360912364639384
	82.196284392035555	0	4.9587917737789216	2.2635850588712954	4.493078337457787	0.801867641088104

其余结果见附件。

之后我们分别对其中数据进行 1%，2%，5%，10%，20%的扰动，发现聚类结果不变，说明该系统具有一定的鲁棒性。

5.3 问题三模型的建立与求解

5.3.1 问题三模型的建立

该问要求我们对表单 3 的玻璃文物的化学成分进行分析，鉴别所属类型并且最后对其分类结果的敏感性进行分析。

但是又考虑到原本的数据样例较小，只使用一种分类算法可能会导致其效果不是很好，于是我们选择分别采用支持向量机(SVM)，贝叶斯岭回归(bayes)、随机森林(RF)、logistic 回归以及 SVM 与 RF 的集成学习对原有的数据进行分析，最后根据这些分类算法给出的结果进行进一步的分析。

支持向量机(SVM)是一类按监督学习方式对数据进行二元分类的广义线性分类器。其决策边界是对学习样本求解的最大边距超平面。往往在计算过程中会使用核函数，将低维的数据映射至高维空间，使得原本在低维空间线性不可分的结果在高维空间线性可分。这样我们就可以求解出一个使得两类样本距离分界超平面最远的分界面，使得之后在判断未知数据的时候拥有更高的准确度。

贝叶斯概率理论体系在机器学习中有着举足轻重的地位。其实很多时候，我们机器学习的算法从本质上来看，就是一种统计学习方法。所以，贝叶斯概率学派的很多思想，是理解机器学习的关键所在。

贝叶斯回归显然是贝叶斯理论在线性回归的一个应用。sklearn 一上来就给出了一条很重要的性质：在贝叶斯概率模型中，我们用参数的概率分布（参数本身具有分布的形式），取代了常规正则化。

贝叶斯岭回归(bayes)是贝叶斯回归的一种经典形式。它在贝叶斯回归的基础上，对参数ω的分布做了进一步的假设，即满足高斯球分布，而先验参数一般是服从 gamma 分布，这个分布与高斯成共轭先验关系。

决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。Entropy = 系统的凌乱程度，使用算法 ID3, C4.5 和 C5.0 生成树算法使用熵。这一度量是基于信息学理论中熵的概念。

logistic 回归又称 logistic 回归分析，主要在流行病学中应用较多，它属于广义线性回归模型，因变量呈现二项分布，即属于高钾玻璃还是铅钡玻璃。通过运用最大似然估计来得到预测结果。

5.3.2 问题三模型的求解

SVM 的求解

因为玻璃类型只有高钾与铅钡两种类型，我们假设高钾类型为+1，此时铅钡为-1。这样的情况下我们的训练样本可以写成$\{x_i, y_i | i = 1, 2, ..., N\}$，其中$x_i$即玻璃的特征向量，具体体现为每种化学元素含量的占比组成的向量，y_i则表示其所属的类型+1 或者-1，N即为要用来训练的样本个数。

此时样本满足

$$\begin{cases} w^T\phi(x_i)+b>0 \ \text{当} y_i=+1 \\ w^T\phi(x_i)+b<0 \ \text{当} y_i=-1 \end{cases} \tag{10}$$

这里的w和b都分别是一组向量，由它们一起确立一个超平面$y=w^Tx+b$作为分界依据，而$\phi(\ )$是我们选择的 rb 核函数，它可以将低维的数据映射至高维空间，使得我们可以上原本在低维空间线性不可分的数据在高维空间可分以满足 SVM 模型的要求。

正规化有

$$\begin{cases} w^T\phi(x_i)+b\geq 1 \ \text{当} y_i=+1 \\ w^T\phi(x_i)+b\leq -1 \ \text{当} y_i=-1 \end{cases} \tag{11}$$

如下图 10 所示

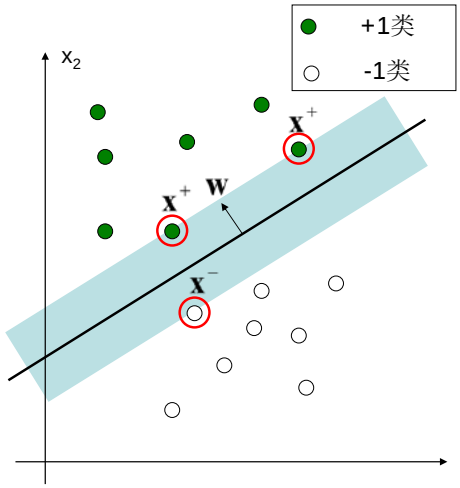

图 10 SVM 分类示意图

在这个时候我们可求得两线之间的间隔为$\frac{2}{||w||}$

我们要求解的是在满足上述条件下最大的间隔即

$$\max \frac{2}{||w||}$$

$$\begin{cases} w^T\phi(x_i)+b\geq 1 \ \text{当} y_i=+1 \\ w^T\phi(x_i)+b\leq -1 \ \text{当} y_i=-1 \end{cases} \tag{12}$$

我们将附表 2 对应的数据放入该模型利用 python 进行求解，在 k 折交叉验证的情况下我们得到其准确率为 88%。

贝叶斯岭回归的求解

在此模型下，我们依旧使用上文的$y=w^Tx+b$来表示我们确立的超平面。

我们假设存在$b\sim N(0,\sigma_0^2)$

由于$y=w^Tx+b$，所以此时的y也符合高斯分布，即：

$$y\sim N(w^Tx,\sigma_0^2)$$

存在先验分布，即参数w的分布，我们同样假设其分布符合高斯分布，即：

$$w\sim N(0,\sigma_1^2)$$

所以这样就会获得两者的概率密度分布公式：

$$P(w)=\frac{1}{\sqrt{2\pi}\,\sigma_1}\exp\left(-\frac{w^T w}{2\sigma_1^2}\right) \tag{13}$$

$$P(Y|w,X)=\frac{1}{\sqrt{2\pi}\,\sigma_0}\exp\left(-\frac{(Y-w^T X)^T(Y-w^T X)}{2\sigma_0^2}\right) \tag{14}$$

为了获得最优解参数，使用最大后验概率估计，即：

$$P(w|Y)=\frac{P(w)P(Y|w)}{P(Y)} \tag{15}$$

在这样的情况下我们直接使用贝叶斯分类器的准则即可得到结果，即选择属于类别更高的一方。

我们同样将附表 2 对应的数据放入该模型利用 python 进行求解，在 k 折交叉验证的情况下我们得到其准确率为 87%。

随机森林的求解

一棵决策树的生成过程主要分为以下 3 个部分:

1.特征选择：特征选择是指从训练数据中众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准标准，对本问而言就是选择某一个化学元素，从而衍生出不同的决策树算法。

2.决策树生成：根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。

3.剪枝：决策树容易过拟合，一般来需要剪枝，缩小树结构规模、缓解过拟合。剪枝技术有预剪枝和后剪枝两种。

而随机森林采用了多个决策树的投票机制来改善决策树，多次采样并代入决策树模型建立起多个决策树模型，之后再利用多个决策树通过投票来最终确定划分的类别。

我们同样将附表 2 对应的数据放入该模型利用 python 进行求解，在 k 折交叉验证的情况下我们得到其准确率为 89%。

logistic 回归的求解

我们通过以下的公式进行回归分析

$$\begin{aligned} logit\ P &\equiv \ln\left(\frac{P}{1-P}\right) \\ &= b+w_1x_1+w_2x_2+...+w_kx_k \end{aligned} \tag{16}$$

其中w_i表示系数w的第i项，x_i代表特征向量x的第i项。

我们同样求一个超平面$y=w^T x+b$。

我们同样将附表 2 对应的数据放入该模型利用 python 进行求解，在 k 折交叉验证的情况下我们得到其准确率为 86%。

SVM 与决策树集成学习的求解

我们将上文提到的支持向量机(SVM)与决策树联合到一个模型中，分别进行目标类别的预测，最终通过投票来选择出更合适的一种判断。

我们同样将附表 2 对应的数据放入该模型利用 python 进行求解，在 k 折交叉验证的情况下我们得到其准确率为 90%。

汇总以上预测有下表 9 所示结果

表 9 训练集上 k 折交叉准确度

	SVM	Bayes	RF	logistic	SVM+RF
准确度	88%	87%	89%	86%	90%

最终我们使用每一种模型对附表 3 的玻璃文物进行类别鉴定，得到如下表 12 所示的结果。

表 10 不同模型对附表 3 的鉴定结果

	SVM	Bayes	RF	logistic	SVM+RF
A1	高钾	高钾	高钾	高钾	高钾
A2	铅钡	铅钡	铅钡	铅钡	铅钡
A3	铅钡	铅钡	铅钡	铅钡	铅钡
A4	铅钡	铅钡	铅钡	铅钡	铅钡
A5	铅钡	铅钡	铅钡	铅钡	铅钡
A6	高钾	高钾	高钾	高钾	高钾
A7	高钾	高钾	高钾	高钾	高钾
A8	铅钡	铅钡	铅钡	铅钡	铅钡

最终结果非常一致，均得到了同样的结果，除 1、6、7 号为高钾玻璃外其余均为铅钡玻璃。

之后我们分别对其中数据进行 1%，2%，5%，10%，20%的扰动，发现分类结果不变，说明该系统具有一定的鲁棒性。

5.4 问题四模型的建立与求解

5.4.1 问题四模型的建立

该问要求我们分别对高钾玻璃和铅钡玻璃分析它们化学成分之间的关联关系，并且对两者不同的关联关系的差异性进行分析。

我们考虑使用灰色关联度分析法对相应数据进行分析。

灰色关联度分析法中的灰色是相对于白色系统和黑色系统而言的，这个概念最早由邓聚龙教授提出，其中白色代表信息充足，比如一个简单的力学系统，我们是很清楚其元素之间的关系的，反之黑色就代表信息缺失，比如一个复杂的混沌系统，我们并不清楚其元素之间的关系。而灰色介于两者之间，表示我们只对系统的部分有所了解。

现在我们想要了解在高钾玻璃与铅钡玻璃这两个灰色系统中各元素之间的关联强弱，我们通过假设某一个指标可能与其他因素有较高的相关性，那么我们就针对这个指标将其余的因素排列一下，总结出一个排列反映哪些因素会与该指标相关性更高。

5.4.2 问题四模型的求解

以高钾为例

由标准差分布，选择 SiO_2 作为参考序列

$$z_0 = (z_0(1)...z_0(n))^T \tag{17}$$

其中 z_0 代表 SiO_2 这一向量序列，每个值对应着相应的 SiO_2 的含量，n 表示总共

的高钾玻璃样本数。

其余的元素作为子序列列为$z_i=(z_i(1)...z_i(n))^T$。

设α为两极极差，β为最大差，有

$$\alpha=\min_i\min_k|z_0(k)-z_i(k)| \tag{18}$$

$$\beta=\max_i\max_k|z_0(k)-z_i(k)| \tag{19}$$

关联系数ν为

$$\nu=\frac{\alpha+\beta^*\rho}{|y_0(k)x_i(k)|+\beta^*\rho} \tag{20}$$

其中ρ是我们自己取的系数，通常将其取为 0.5。

关联度η为

$$\eta=\frac{1}{n}\sum_{k=1}^{n}\nu \tag{21}$$

我们将附表中的数据代入模型通过 python 实现就有如下表 11 表 12 表 13 表 14 结果。

表 11 高钾玻璃风化后关联度分析

评价项	关联度	排名
氧化铁(Fe_2O_3)	0.872	1
氧化铜(CuO)	0.821	2
氧化铝(Al_2O_3)	0.791	3
氧化钙(CaO)	0.786	4
五氧化二磷(P_2O_5)	0.728	5
氧化钾(K_2O)	0.695	6
氧化镁(MgO)	0.518	7
氧化钠(Na_2O)	0	8
氧化铅(PbO)	0	9
氧化钡(BaO)	0	10
氧化锶(SrO)	0	11
氧化锡(SnO_2)	0	12
二氧化硫(SO_2)	0	13

表 12 高钾玻璃未风化关联度分析

评价项	关联度	排名
氧化铝(Al_2O_3)	0.942	1
氧化钾(K_2O)	0.938	2
氧化镁(MgO)	0.916	3
氧化钙(CaO)	0.911	4
氧化铜(CuO)	0.909	5
氧化铁(Fe_2O_3)	0.9	6
五氧化二磷(P_2O_5)	0.893	7
氧化锶(SrO)	0.846	8

氧化铅(PbO)	0.837	9
氧化钡(BaO)	0.81	10
氧化锡(SnO_2)	0.804	11
二氧化硫(SO_2)	0.794	12
氧化钠(Na_2O)	0.791	13

表 13 铅钡玻璃风化后关联度分析

评价项	关联度	排名
氧化铅(PbO)	0.955	1
氧化铝(Al_2O_3)	0.938	2
氧化钡(BaO)	0.929	3
氧化钙(CaO)	0.927	4
氧化锶(SrO)	0.917	5
五氧化二磷(P_2O_5)	0.907	6
氧化镁(MgO)	0.904	7
氧化铁(Fe_2O_3)	0.895	8
氧化铜(CuO)	0.891	9
氧化钾(K_2O)	0.875	10
氧化钠(Na_2O)	0.85	11
氧化锡(SnO_2)	0.849	12
二氧化硫(SO_2)	0.839	13

表 14 铅钡玻璃未风化关联度分析

评价项	关联度	排名
氧化铅(PbO)	0.962	1
氧化铝(Al_2O_3)	0.956	2
氧化钡(BaO)	0.954	3
氧化镁(MgO)	0.943	4
氧化锶(SrO)	0.942	5
氧化钙(CaO)	0.936	6
氧化钾(K_2O)	0.932	7
氧化铜(CuO)	0.919	8
氧化铁(Fe_2O_3)	0.908	9
氧化钠(Na_2O)	0.907	10
五氧化二磷(P_2O_5)	0.906	11
二氧化硫(SO_2)	0.892	12
氧化锡(SnO_2)	0.877	13

以上表格说明铅钡玻璃风化后化学成分关联关系较强，而高钾玻璃风化后化学成分关联关系相对减弱。

其中高钾玻璃风化前后差异性较大，在风化前主要与 Al、K、Mg、Ca、Cu 关系更紧密，但风化后与 Fe、Cu、Al、Ca、P 关系更加紧密，排名也有较大变化，这说明高钾玻璃在风化中更容易丢失相关元素。而铅钡玻璃始终化学成分关联度都比较高，特别是 Pb、Al、Ba 始终位于前三关联度，说明其在风化中不易丢失相关元素。

总的来说，高钾玻璃关联度较大的元素为 Al、K、Cu、Mg，而铅钡玻璃关联度较大的是 Pb、Al、Ba、Mg。

可以注意到高钾风化后，Fe_2O_3，CuO 的关联度获得了很大的提升，分别到了第一和第二，结合机理，确实符合在风华表面形成了富硅层，吸附了土壤中的 Fe，Cu 等离子的原因。

六、 模型的评价与改进

6.1 模型的优点

我们的模型是经过了化学机理的验证，能够更加有效地反映出不同化学成分之间的统计规律。

我们在意识到在当前不均衡数据集下贝叶斯估计效果较弱，采用了多模型融合来弥补这一缺陷。

本体的样本较小，在解决第四题的时候我们通过灰色关联分析弥补了采用数理统计方法做系统分析被小样本影响所导致的缺陷。计算量小，不会出现量化结果与定性结果不符的情况。

CRTIC 是比熵权法和标准离差法更好的客观赋权法。对于本题，考虑化学成分变异性大小的同时兼顾化学成分之间的相关性，完全利用数据自身的客观属性进行科学评价。符合人类对于化学制品的普遍划分规律：先划分差异性大的成分，再划分差异性小的。

6.2 模型的缺点

我们在解决问题的过程中忽略了严重风化与普通风化均一性不一致。

6.3 模型的改进

在时间允许的情况下，我们可以将更多的化学机理相关知识融入到我们的模型中来，并且将原本我们为了简化题目而忽略的一些因素加入考虑。

七、 参考文献

[1] 王婕,李沫,马清林,张治国,章梅芳,王菊琳.一件战国时期八棱柱状铅钡玻璃器的风化研究[J].玻璃与搪瓷,2014,42(02):6-13.DOI:10.13588/j.cnki.g.e.1000-2871.2014.02.002

[2] 李曼.郑州地区出土战国蜻蜓眼珠饰的无损分析及制作工艺初探[J].中原文物,2022,(03):126-135.

[3] 赵志强.新疆巴里坤石人子沟遗址群出土玻璃珠的成分体系与制作工艺研究[D].西北大学,2016.

[4] 余庆.基于多变量样本图方法的古陶瓷分类研究[D].景德镇陶瓷学院,2012.

[5] 宁发艳.基于多元统计和智能算法的古陶瓷分类研究[D].景德镇陶瓷大学,2019

[6] 赵华玲.淄博地区古代玻璃历史发展的研究[D].苏州大学,2008.

[7] 谭学瑞, 邓聚龙. 灰色关联分析;多因素统计分析新方法[J]. 统计研究, 1995, 000(003):46-48.

[8] 郭为民.灰色理论和加权平均法组合预测家用空调产品销售量[J].内蒙古科技与经济,2021(15):61-62

[9] 冯丽明,孙虹.基于回归分析的大学女生体脂百分比模型优选[J].安康学院学报,2015,27(06):85-88+93.DOI:10.16858/j.issn.1674-0092.2015.06.021

[10] 高庚.最小二乘法和总体最小二乘法线性回归中的估值漂移及其判定[D].太原理工大学,2018.

[11] 冯百龄.中国出土古代玻璃珠数据库建设与应用[D].西北大学,2021.

[12] 陈姝聿.我国古代玻璃的起源和发展[J].文物鉴定与鉴赏,2019,(04):44-45.

[13] 李盈理.10 至 12 世纪中国出土玻璃器研究[D].辽宁师范大学,2021.

[14] 温睿,赵志强,马健,王建新.新疆巴里坤石人子沟遗址群出土玻璃珠的成分分析[J].光谱学与光谱分析,2016,36(09):2961-2965.

附录

附录 1
介绍：支撑材料的文件列表
我们的支撑材料文件名为“支撑材料.zip”，在“支撑材料.zip”中有“程序”与“附件”两个文件夹，在“程序”文件夹中分别有“Q1”、“Q2”、“Q3”、“Q4”四个文件夹，这些文件夹内有着对应着相应题号的代码文件，在“附件”文件夹中有“图片”与“数据”两个文件夹，“图片”文件夹中包含了在解题过程中产生的辅助的图片，“数据”则存放了在问题中反复用到的数据。

附录
第一问第 3 小问的答案展示 单位%

	纹饰	类型	颜色	表面风化	文物采样点	二氧化硅(Si02)	氧化钠(Na20)	氧化钾(K20)	氧化钙(Ca0)	氧化镁(Mg0)
0	B	高钾	蓝绿	风化	7	73.94	0.00	0.00	8.22	0.00
1	B	高钾	蓝绿	风化	27	73.33	0.00	0.00	6.14	5.19
2	B	高钾	蓝绿	风化	9	72.41	0.00	12.04	3.99	0.00
3	B	高钾	蓝绿	风化	22	59.69	0.00	12.21	8.63	4.90
4	B	高钾	蓝绿	风化	12	66.89	0.00	18.32	4.96	0.00
5	B	高钾	蓝绿	风化	10	74.48	0.00	18.10	1.30	0.00
6	C	铅钡	浅蓝	风化	54 严重风化点	49.65	0.00	0.00	0.00	1.52
7	C	铅钡	浅蓝	风化	43 部位 2	55.76	0.00	0.00	3.23	1.17
8	A	铅钡	黑	风化	49	62.24	0.00	0.00	1.94	1.52
9	C	铅钡	浅蓝	风化	11	67.80	0.00	0.37	1.39	0.69
10	C	铅钡	0	风化	58	62.91	0.00	0.62	1.42	0.78
11	A	铅钡	0	风化	19	63.56	0.00	0.00	1.23	0.61
12	C	铅钡	浅蓝	风化	51 部位 2	57.38	0.00	0.00	2.71	1.87
13	C	铅钡	浅蓝	风化	51 部位 1	56.75	0.00	0.00	1.62	1.31
14	C	铅钡	紫	风化	08 严重风化点	20.03	0.00	0.00	2.38	0.00
15	C	铅钡	浅绿	风化	41	48.22	0.00	1.01	2.54	3.42
16	A	铅钡	黑	风化	50	49.96	0.00	0.00	1.74	0.62
17	C	铅钡	紫	风化	26 严重风化点	14.10	0.00	1.33	2.24	0.00
18	C	铅钡	浅蓝	风化	52	53.79	12.94	0.00	0.93	0.55
19	C	铅钡	浅蓝	风化	54	51.42	0.00	0.68	1.54	1.49
20	C	铅钡	紫	风化	8	47.89	0.00	0.00	0.69	0.00
21	A	铅钡	浅蓝	风化	2	67.48	0.00	1.69	0.84	1.04
22	C	铅钡	紫	风化	26	47.47	0.00	0.00	0.67	0.00
23	C	铅钡	蓝绿	风化	56	62.35	0.00	0.00	0.54	0.00
24	C	铅钡	0	风化	40	46.71	0.00	0.00	1.03	0.00
25	C	铅钡	深绿	风化	39	60.44	0.00	0.00	0.50	0.00
26	A	铅钡	0	风化	48	70.26	5.29	0.38	0.75	1.00
27	C	铅钡	深绿	风化	38	57.83	13.49	0.00	0.19	0.00
28	C	铅钡	深绿	风化	34	68.93	0.00	0.42	0.30	0.00

29	C	铅钡	深绿	风化	36	60.16	16.50	0.19	0.11	0.00
30	C	铅钡	浅蓝	风化	43 部位 1	35.7%	0	0	3.2%	1.3%
31	C	铅钡	蓝绿	风化	57	58.9%	0	0	0.6%	0

氧化铝(A1203)	氧化铁(Fe203)	氧化铜(Cu0)	氧化铅(Pb0)	氧化钡(Ba0)	五氧化二磷(P205)	氧化锶(Sr0)	氧化锡(Sn02)	二氧化硫(S02)	sum
7.72	1.31	5.83	0.00	0.00	2.99	0.00	0.00	0.00	97.38
9.46	1.49	2.68	0.00	0.00	1.71	0.00	0.00	0.00	98.19
4.90	2.34	2.66	0.00	0.00	1.66	0.00	0.00	0.00	96.67
10.94	2.07	0.76	0.00	0.00	0.79	0.00	0.00	0.00	100.52
4.82	1.89	2.51	0.00	0.00	0.62	0.00	0.00	0.00	100.85
2.90	1.83	1.39	0.00	0.00	0.00	0.00	0.00	0.00	100.00
7.12	0.00	1.04	37.18	0.00	2.60	0.88	0.00	0.00	100.70
5.98	1.81	1.06	25.56	2.99	2.12	0.33	0.00	0.00	100.00
7.93	3.01	0.41	16.42	4.71	1.54	0.27	0.00	0.00	100.00
3.70	0.00	2.71	11.39	10.53	1.22	0.20	0.00	0.00	100.00
5.91	0.90	1.76	18.71	5.66	1.20	0.14	0.00	0.00	101.57
5.22	1.50	2.05	20.41	4.10	1.22	0.11	0.00	0.00	100.05
4.60	0.57	0.70	30.67	0.00	1.51	0.00	0.00	0.00	100.15
8.26	1.39	0.86	20.62	7.36	1.20	0.25	0.37	0.00	100.00
2.88	0.00	3.25	27.45	41.60	1.85	0.55	0.00	2.70	99.85
6.00	2.37	0.14	25.62	9.10	1.25	0.34	0.00	0.00	100.06
3.54	0.46	0.85	27.13	14.06	1.13	0.50	0.00	0.00	100.08
3.05	0.00	3.72	25.21	47.97	1.74	0.64	0.00	2.60	97.55
1.65	0.24	0.40	22.03	6.45	0.77	0.25	0.00	0.00	100.47
6.90	0.00	0.55	30.03	6.12	0.66	0.59	0.00	0.00	97.23
2.17	0.00	6.74	15.16	26.51	0.60	0.24	0.00	0.28	99.77
7.18	1.74	0.13	19.38	0.00	0.42	0.10	0.00	0.00	100.78
1.13	0.00	6.79	15.49	27.69	0.47	0.29	0.00	0.21	101.11
2.87	0.00	0.49	20.84	12.54	0.37	0.00	0.00	0.00	96.23
0.86	0.27	0.00	43.62	6.68	0.32	0.52	0.00	0.00	100.00
0.79	0.00	0.55	31.23	5.93	0.17	0.38	0.00	0.00	100.00
12.58	0.71	0.00	4.72	3.53	0.10	0.09	0.60	0.00	98.21
2.99	0.25	0.34	18.69	5.96	0.05	0.19	0.00	0.00	103.75
2.13	0.46	0.79	19.93	6.88	0.04	0.12	0.00	0.00	100.00
1.66	0.25	0.28	14.88	5.88	0.01	0.09	0.00	0.00	100.82
4.70	1.18	4.46	40.76	8.17	0.00	0.54	0.00	0.00	97.95
3.34	0.00	0.71	22.54	13.89	0.00	0.00	0.00	0.00	101.80

附录

第二问 2 小问的聚类结果

表 1 风化后高钾玻璃亚类分类表

样本类别	样本编号
第 1 类	7、10、12
第 2 类	22、27
第 3 类	9

表 2 未风化铅钡玻璃亚类分类表

样本类别	样本编号
第 1 类	30、33、45
第 2 类	31、46
第 3 类	24、37
第 4 类	20、32、35、55

表 3 风化后铅钡玻璃亚类分类表

样本类别	样本编号
第 1 类	23、28、40、42、43、51、53
第 2 类	2、11、19、29、41、49、50、52、54、58
第 3 类	8、25、26、34、39、44
第 4 类	36、56

附录

用 python 划分数据集为高钾风化，高钾未风化，高钾，铅钡，铅钡未风化，铅钡风化，风化，未风化

```
# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus']=False

# %%
df_2 = pd.read_excel("/Desktop/附件-GJ.xlsx", sheet_name="表单 2")


df_2

```

```
15.
16.     # %%
17.     df_2.sum(axis=1)
18.     df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
19.
20.     for column in df_2.columns[6:19]:
21.         df_2[column]=df_2[column]/df_2['sum']*100
22.     df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
23.
24.     df=df_2[df_2['风化情况']==0]
25.     df.to_excel("GJNFH.xlsx")
26.
27.     # %%
28.
29.
30.     df1=df_2[df_2['风化情况']==1]
31.     df1.to_excel("GJFH.xlsx")
32.
33.     # %% [markdown]
34.     #
35.
36.
37.     # %%
38.     import pandas as pd
39.     import numpy as np
40.     import matplotlib.pyplot as plt
41.     import seaborn as sns
42.     plt.rcParams['font.sans-serif'] = 'SimHei'
43.     plt.rcParams['axes.unicode_minus']=False
44.
45.     # %%
46.     df_2 = pd.read_excel("/Desktop/fh.xlsx", sheet_name="风化")
47.
48.
49.     df_2
50.
51.
52.     # %%
53.     df_2.sum(axis=1)
54.     df_2['sum']=df_2[df_2.columns[7:20]].sum(axis=1)
55.     for column in df_2.columns[7:20]:
56.         df_2[column]=df_2[column]/df_2['sum']*100
57.     df_2['sum']=df_2[df_2.columns[7:20]].sum(axis=1)
58.     df_2.to_excel("/Desktop/FH.xlsx")
```

```
59.
60.     # %% [markdown]
61.     #
62.
63.     # %%
64.     import pandas as pd
65.     import numpy as np
66.     import matplotlib.pyplot as plt
67.     import seaborn as sns
68.     plt.rcParams['font.sans-serif'] = 'SimHei'
69.     plt.rcParams['axes.unicode_minus']=False
70.
71.     # %%
72.     df_2 = pd.read_excel("/Desktop/附件-QB.xlsx", sheet_name="表单 2")
73.
74.
75.     df_2
76.
77.
78.     # %%
79.     df_2.sum(axis=1)
80.     df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
81.
82.     for column in df_2.columns[6:19]:
83.         df_2[column]=df_2[column]/df_2['sum']*100
84.     df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
85.     df_2
86.
87.     # %% [markdown]
88.     #
89.
90.     import pandas as pd
91.     import numpy as np
92.     import matplotlib.pyplot as plt
93.     import seaborn as sns
94.     plt.rcParams['font.sans-serif'] = 'SimHei'
95.     plt.rcParams['axes.unicode_minus']=False
96.
97.     # %%
98.     df_2 = pd.read_excel("/Desktop/附件-GJ.xlsx", sheet_name="表单 2")
99.
100.
101.    df_2
102.
```

```
103.
104.    # %%
105.    df_2.sum(axis=1)
106.    df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
107.
108.    for column in df_2.columns[6:19]:
109.        df_2[column]=df_2[column]/df_2['sum']*100
110.    df_2['sum']=df_2[df_2.columns[6:19]].sum(axis=1)
111.    df_2
112.
113.    # %% [markdown]
114.    #
```

附录

用 SPSS 实现的问题一中的卡方检验

```
WEIGHT BY 个数.
CROSSTABS
  /TABLES=是否风化 BY 纹饰类型 BY 个数
  /FORMAT=AVALUE TABLES
  /STATISTICS=CHISQ CC PHI LAMBDA
  /CELLS=COUNT
  /COUNT ROUND CELL.

WEIGHT BY 个数.
CROSSTABS
  /TABLES=是否风化 BY 颜色 BY 个数
  /FORMAT=AVALUE TABLES
  /STATISTICS=CHISQ CC PHI LAMBDA
  /CELLS=COUNT
  /COUNT ROUND CELL.

WEIGHT BY 个数.
CROSSTABS
  /TABLES=是否风化 BY 玻璃类型 BY 个数
  /FORMAT=AVALUE TABLES
  /STATISTICS=CHISQ CC PHI LAMBDA
  /CELLS=COUNT
  /COUNT ROUND CELL.
```

附录

用 python 实现问题二中的 CRITIC 权重法

```
1.      from sklearn.cluster import KMeans
2.      from sklearn.cluster import DBSCAN
3.      import pandas as pd
4.      import numpy as np
5.      import math
```

```
import matplotlib.pyplot as plt
import pickle
import statsmodels.api as sm

df = pickle.load(open("../data_logit.pkl","rb"))
df.iloc[:,5:] = df.iloc[:,5:].div(df.iloc[:,5:].sum(axis=1), axis=0) #! 行规一化

df_GJ = df.loc[df["类型"]=="高钾",:]
df_QB = df.loc[df["类型"]=="铅钡",:]

df_GJ_fh = df_GJ.loc[df["风化情况"]==1,:]
df_GJ_no_fh = df_GJ.loc[df["风化情况"]==0,:]
df_QB_fh = df_QB.loc[df["风化情况"]==1,:]
df_QB_no_fh = df_QB.loc[df["风化情况"]==0,:]
# coding=gbk

import pandas as pd
import numpy as np

def critic(df):
    df = df.iloc[:,5:].transpose()
    # print(df)
    std_d=np.std(df,axis=1)
    mean_d=np.mean(df,axis=1)
    cor_d=np.corrcoef(df)
    w_j=(1-cor_d).sum(0) *std_d
    # print(w_j)
    w=(mean_d/(1-mean_d)*w_j)/sum(mean_d/(1-mean_d)*w_j)
    # print(w)
    return dict(w)

sorted(critic(df_GJ_fh),reverse=True)

sorted(critic(df_GJ_no_fh),reverse=True)

sorted(critic(df_QB_fh),reverse=True)

sorted(critic(df_QB_no_fh),reverse=True)
```

附录

用 python 实现第二问对重要变量的选择

```
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import pickle
import statsmodels.api as sm

df = pickle.load(open("../data_logit.pkl","rb"))

df_GJ = df.loc[df["类型"]=="高钾",:]
df_QB = df.loc[df["类型"]=="铅钡",:]

from collections import Counter
def outlier_hunt(df):
    """
    Takes a dataframe df of features and returns a list of the indices
    corresponding to the observations containing more than 2 outliers.
    """
```

```
        outlier_indices = []

        # iterate over features(columns)
        for col in df.columns.tolist():
            # 1st quartile (25%)
            Q1 = np.percentile(df[col], 25)

            # 3rd quartile (75%)
            Q3 = np.percentile(df[col],75)

            # Interquartile rrange (IQR)
            IQR = Q3 - Q1

            # outlier step
            outlier_step = 1.5 * IQR

            # Determine a list of indices of outliers for feature col
            outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + o
utlier_step )].index

            # append the found outlier indices for col to the list of outlier indic
es
            outlier_indices.extend(outlier_list_col)

        # select observations containing more than 2 outliers
        outlier_indices = Counter(outlier_indices)
        multiple_outliers = list( k for k, v in outlier_indices.items() if v > 2 )


        return multiple_outliers

    def rid_of_outliner(df):
        features = df.columns[5:]

        idx = [True if (i not in outlier_hunt(df[features])) else False for i in ra
nge(len(df)) ]
        return df.loc[idx,:]

    df_GJ = rid_of_outliner(df_GJ)
    df_QB = rid_of_outliner(df_QB)

    def get_train_data(df):
        ori_id = {i : df.iloc[i, 0] for i in range(len(df))}
        return np.concatenate((df.loc[:, ['氧化镁(MgO)','氧化锶(SrO)','氧化锡
(SnO2)','氧化铝(Al2O3)','氧化铜(CuO)','氧化铅
(PbO)',]].to_numpy(),df.iloc[:, 15].to_numpy().reshape(-1, 1)), axis=1), ori_id # 对应
的文物编号

    train_GJ, id_GJ = get_train_data(df_GJ)
    train_QB, id_QB = get_train_data(df_QB)
    # train_GJ

    def select_MinPts(data,k):
        k_dist = []
        for i in range(data.shape[0]):
            dist = (((data[i] - data)**2).sum(axis=1)**0.5)
            dist.sort()
            k_dist.append(dist[k])
        return np.array(k_dist)

    def cluster(x_QB, eps, method = "D", metric="euclidean"):
        from sklearn.cluster import DBSCAN
        from sklearn.cluster import OPTICS

        k = 13  # 此处 k 取 2*2 -1
```

```
        k_dist = select_MinPts(x_QB,k)
        k_dist.sort()
        # plt.plot(np.arange(k_dist.shape[0]),k_dist[::-1])

        if method == "D": clustering = DBSCAN(eps=2,min_samples=3, metric=metric).f
it(x_QB)
        if method == "O": clustering = OPTICS(metric=metric).fit(x_QB)
        # show_PCA(x_QB,clustering.fit(x_QB).labels_) #! no PCA = False

        metrics = {}
        from sklearn.metrics import silhouette_score
        from sklearn.metrics import calinski_harabasz_score
        from sklearn.metrics import davies_bouldin_score

        metrics["SC"] = silhouette_score(x_QB, clustering.labels_)
        metrics["CHI"] = calinski_harabasz_score(x_QB, clustering.labels_)
        metrics["DBI"] = davies_bouldin_score(x_QB, clustering.labels_)
        # print(clustering.cluster_centers_)
        return clustering.labels_, metrics

    GJ_result_e = cluster(train_GJ, 1, method = "D") # 欧氏距离
    QB_result_e = cluster(train_QB, 0, method = "O")

    # GJ_result_c = cluster(train_GJ, 1, method = "D",metric="cosine") # 余弦距离
    # QB_result_c = cluster(train_QB, 0, method = "O",metric="cosine")

    def assign_id(x, d):
        y = {d[i]:x[i] for i in range(len(x))}
        return y

    QB_result_e = assign_id(QB_result_e[0], id_QB), QB_result_e[1]
    GJ_result_e = assign_id(GJ_result_e[0], id_GJ), GJ_result_e[1]
    GJ_result_e
    QB_result_e

    # print(id_GJ)
    excel_writer = pd.ExcelWriter('聚类结果 v2.xlsx')
    pd.DataFrame({"文物 id" :GJ_result_e[0].keys(), "类
":GJ_result_e[0].values()}).to_excel(excel_writer, sheet_name="高钾")
    pd.DataFrame({"文物 id" :QB_result_e[0].keys(), "类
":QB_result_e[0].values()}).to_excel(excel_writer, sheet_name="铅钡")
    excel_writer.save()  # 储存文件
    excel_writer.close()


    print("GJ_result:",GJ_result_e[1],"\nQB_result:", QB_result_e[1])

    def show_PCA(x_QB, y_QB, no_PCA=False):
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd
        import matplotlib.pyplot as plt
        import pickle
        import torch
        from sklearn.manifold import TSNE
        from sklearn.decomposition import PCA
        from sklearn.datasets import load_digits


        # 数据需要是 numpy array
        data_X = x_QB
        y = y_QB

        # 选择维度
        pca = PCA(n_components=2)
```

```
        tsne_obj = pca.fit_transform(data_X)


        tsne_df = pd.DataFrame({'X':tsne_obj[:,0],
                                'Y':tsne_obj[:,1],
                                'digit':y})
        print(tsne_df.head())

        if no_PCA: sns.scatterplot(x="X", y="Y",
                    hue="digit",
                    palette=['green','red','yellow','blue',"orange","black"][:len(set(y))],
                    legend='full',
                    data=pd.DataFrame({'X':data_X[:,2],
                                'Y':data_X[:,3],'digit':y}))
        else :sns.scatterplot(x="X", y="Y",
                    hue="digit",
                    palette=['green','red','yellow','blue',"orange","black"][:len(set(y))],
                    legend='full',
                    data=tsne_df)

        plt.show()


```

附录

用 python 实现问题二中的数据扰动

```
clc
clear
close all
fName = ["gjfh.xlsx","gjnfh.xlsx","qbfh.xlsx","qbnfh.xlsx"];
dicName = [1,2,5,10,20,30];
T = xlsread("zoresads.xlsx");
P = [[43,a5],[2,3,10],[7,8,9,13],[11]];
B = zeros(4,2,13);
B(1,:,:) = T(1:1:2,2:1:14);%gj 类风化
B(2,:,:) = T(3:1:4,2:1:14);%gj 类无风化
B(3,:,:) = T(5:1:6,2:1:14);%qb 类风化
B(4,:,:) = T(10:1:11,2:1:14);%qb 类无风化
error = 0.000000;
total = 0;
for t=1:4
    p = P(t);
    A = xlsread(fName(t));
    [n,m] = size(A);
    Data = A(:,4:m-1);
    for i=1:n
       t1 = GetNorm(p,Data(i,:),B(t,1.,:));
       t2 = GetNorm(p,Data(i,:),B(t,2,:));
       if t1>t2
           Result(t,i) = 1;
       else
           Result(t,i) = 2;
       end
    end
    for j = 1:length(dicName)
        D =  xlsread( "rjzia\rand" + dicName(j) + "\" + fName(t));
        [n,m] = size(D);
        Data = D(:,4:m-1);
        for ii=1:n
```

```
                t1 = GetNorm(p,Data(ii,:),B(t,1,:));
                t2 = GetNorm(p,Data(ii,:),B(t,2,:));
                if t1>t2
                    res = 1;
                else
                    res = 2;
                end
                if res ~= Result(t,ii)
                    error = error + 1;
                end
                total = total + 1;
            end

            er(t,j) =(1- error/total)*100;
            total = 0;
            error = 0;
        end
    end
    er

    function val = GetNorm(p,a,b)
        val = 0;
        for i=1:length(p)
            val = val + (a(1,p(i))-b(1,1,p(i)))*(a(1,p(i))-b(1,1,p(i)));
        end
    %     for i=1:13
    %         val = val + (a(1,i)-b(1,1,i))*(a(1,i)-b(1,1,i));
    %     end
    end
```

附录

用 python 实现的问题三中的 logistic 回归

```
from operator import index
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import statsmodels.api as sm
df = pd.read_excel("../附件-processedv3.0_not_logit.xlsx", sheet_name="表单 2")
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
import pickle

x_train_nl, y_train_nl = pickle.load(open("../data.pkl", "rb"))

clf = LogisticRegression()
scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=5)
print(scores.mean())
```

附录

问题三中的 SVM、RF、logistic、bayes 对结果的预测

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

```
from copyreg import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn import svm
import pickle
x_train_nl, y_train_nl = pickle.load(open("../data.pkl", "rb"))
x_test = pickle.load(open("../data_test.pkl", "rb"))

clf = svm.SVC()
scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=2)
print(scores.mean())

scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=5 )
print(scores.mean())

scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=10 )
print("交叉验证：",scores.mean())

clf.fit(x_train_nl, y_train_nl)

print("最终预测：",clf.predict(x_test))

from sklearn.model_selection import LeaveOneOut
clf = svm.SVC()
loocv = LeaveOneOut()
scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=loocv)
print(scores.mean())
from copyreg import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn import svm
import pickle

clf = LogisticRegression()
scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=2)
print(scores.mean())

scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=5 )
print(scores.mean())

scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=10 )
print(scores.mean())

```

```
clf.fit(x_train_nl, y_train_nl)

print(scores.mean())

print(clf.predict(x_test))

from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

x_train_nl, y_train_nl = pickle.load(open("../data.pkl", "rb"))

clf = RandomForestClassifier(random_state=0)
scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=5 )
clf.fit(x_train_nl, y_train_nl)

print(scores.mean())

print(clf.predict(x_test))

from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
clf =BernoulliNB()

scores = cross_val_score(clf, x_train_nl, y_train_nl, cv=5)
print("平均正确率:" ,scores.mean())

clf.fit(x_train_nl, y_train_nl)

print(scores.mean())

print(clf.predict(x_test))

```

附录

用 python 语言实现问题四中的灰色关联度分析

```
    # 导入可能要用到的库
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
    data_name = ["QB-fh", "QB-no-fh", "GJ=fh", "GJ-no-fh"]
    data = pd.read_excel("../附件-processedv3.0_not_logit.xlsx", sheet_name="表单
2")
    data.shape

    df = {}

```

```
    df[0] = data.iloc[np.logical_and((data["类型"]=="高钾").to_numpy(), (data["风化情况"] !=0).to_numpy()),:]
    df[1] = data.iloc[np.logical_and((data["类型"]=="高钾").to_numpy(), (data["风化情况"] ==0).to_numpy()),:]
    df[2] = data.iloc[np.logical_and((data["类型"]=="铅钡").to_numpy(), (data["风化情况"] !=0).to_numpy()),:]
    df[3] = data.iloc[np.logical_and((data["类型"]=="铅钡").to_numpy(), (data["风化情况"] ==0).to_numpy()),:]

    f = lambda x : x.iloc[:,6:]
    for i in range(4): df[i] = f(df[i])
    # 无量纲化
    def dimensionlessProcessing(df_values,df_columns):
        from sklearn.preprocessing import StandardScaler
        scaler = StandardScaler()
        res = scaler.fit_transform(df_values)
        return pd.DataFrame(res,columns=df_columns)

    # 求第一列(影响因素)和其它所有列(影响因素)的灰色关联值
    def GRA_ONE(data,m=0): # m 为参考列
        # 标准化
        data = dimensionlessProcessing(data.values,data.columns)
        # 参考数列
        std = data.iloc[:,m]
        # 比较数列
        ce = data.copy()

        n = ce.shape[0]
        m = ce.shape[1]

        # 与参考数列比较，相减
        grap = np.zeros([n,m])
        for i in range(m):
            for j in range(n):
                grap[j,i] = abs(ce.iloc[j,i] - std[j])

        # 取出矩阵中的最大值和最小值
        mmax = np.amax(grap)
        mmin = np.amin(grap)
        ρ = 0.5 # 灰色分辨系数

        # 计算值
        grap = pd.DataFrame(grap).applymap(lambda x:(mmin+ρ*mmax)/(x+ρ*mmax))
        # print(grap)
        # 求均值，得到灰色关联值
        ax = grap.plot(figsize=(18,12), xticks=np.arange(len(grap))) #! plot
        plt.legend(loc=2, bbox_to_anchor=(1.05,1.0),borderaxespad = 0.)
        fig = ax.get_figure()
        fig.savefig('fig%s.png' % id(data))

        RT = grap.mean(axis=0)
        return pd.Series(RT)

    # 调用 GRA_ONE，求得所有因素之间的灰色关联值
    def GRA(data):
        list_columns = np.arange(data.shape[1])
        df_local = pd.DataFrame(columns=list_columns)
        # for i in np.arange(data.shape[1]):
        df_local.iloc[:,0] = GRA_ONE(data,m=0)
        return df_local
        # print(df_local)

    data_gra = {}
```

```
71.
72.     for i in range(4):
73.         data_gra[i] = GRA(df[i])
74.
75.     # data_gra[0][0]
76.     pd.DataFrame({data_name[i]:data_gra[i][0].to_numpy() for i in range(4)}).transp
ose().to_excel("GRA.xlsx")
77.
```