基于机器学习的古代玻璃文物成分分析及分类

摘要

本文运用机器学习等算法，研究玻璃样品的化学成分含量并对其进行分类鉴别。

首先对数据进行预处理，检验各化学成分的含量数据，剔除掉不满足成分性的样本数据。

针对问题一，首先要寻找文物玻璃基本属性之间的关系：本文运用 **Fisher 精确检验**方法对表面风化与其玻璃类型、纹饰和颜色之间的关系作相关性分析，可得出表面风化与类型的关系显著，而认为表面风化与纹饰和颜色不具有明显相关性。接下来需要对于两种不同类型的玻璃，分别分析表面风化与未风化玻璃样品的化学成分统计规律：本文分别绘制其风化前后各化学成分含量的箱型图将数据可视化，并列出了各项统计学数据分析其统计规律。最后，本文提出**标准化映射模型**，建立每个化学成分风化前后的大小映射关系，预测出风化前的化学成分含量。

针对问题二，首先要分析高钾与铅钡玻璃的分类规律：本文先运用**主成分分析法**对化学成分参数及风化属性这 15 个指标进行降维处理，接下来使用 **Logistic 回归分类**与**随机森林分类**两种机器学习分类算法求解高钾与铅钡玻璃的分类规律，两种分类的预测准确度都接近甚至达到 100%，分类效果好，从结果来看，本题更适合使用 Logistic 回归分类。接下来进行亚类划分：首先使用**熵权法**，对于高钾玻璃和铅钡玻璃都选择了氧化纳、氧化锡、二氧化硫 3 个化学成分作为分类指标，而后使用 **Kmeans++ 聚类算法**将高钾玻璃分为 3 个亚类，铅钡玻璃分为 4 个亚类，并将分类方式可视化，其中可以看出分类效果较好，不同类之间的特征差异明显。最后进行合理性和敏感性的分析：通过召回率、精确度和 F1 分数 3 个二分类指标评估分类效果，引入**轮廓系数**的指标评价亚分类的合理性，并采用对原数据增加随机扰动的方式得出**分类准确率-扰动大小**的关系，得到高钾-铅钡玻璃的分类准确率受扰动影响较小，敏感性低；亚分类的准确率受扰动影响较大，敏感性较高。

针对问题三，采用问题二中的分类与聚类模型对未知玻璃样品进行分类，得出分类结果，并同样对原数据增加扰动分析其敏感性，得出高钾-铅钡玻璃的分类准确率受扰动影响较小，敏感性较低。

针对问题四，为了探究不同类别化学成分之间的关联关系，本文首先绘制**皮尔逊相关系数矩阵**的热力图，将化学成分间两两的线性相关性大小可视化，接着进行**因子分析**，进一步分析其内在的关联，得出了化学成分之间关联关系，并从化学与历史的角度解释了结果。

关键字： Fisher 精确检验　标准化映射模型　机器学习　Kmeans++ 聚类算法

一、 问题的提出

1.1 背景

在文物鉴定及考古研究中，对文物的成分分析是辨识文物真伪、鉴定文物价值的重要手段之一。古代玻璃制品的化学成分分析和鉴定，对于研究中国古代玻璃的起源与中国古代丝绸之路的贸易历史有重要意义。

本题中，给出了一批玻璃制品的基本信息及化学成分数据，我们希望通过数学建模以及化学领域的相关知识，对这批古代玻璃制品进行成分分析与分类鉴别。

1.2 重述

1. 分析玻璃的基本信息与化学成分数据，得出表面风化与其玻璃类型、纹饰和颜色的关系；对于两种不同类型的玻璃，统计玻璃样品表面风化与未风化的化学成分统计规律；并根据风化点的玻璃类型及化学成分数据，预测其风化前的化学成分数据；
2. 依照给定的玻璃相关数据，分析高钾玻璃、铅钡玻璃的分类规律；对于这两种不同类型，选定合适的化学成分对其进行亚类划分，并给出划分结果；分析分类结果的合理性并进行敏感性分析；
3. 给出一批未知类别玻璃文物的化学成分数据，基于问题二分析其所属类型，并对分类结果进行敏感性分析；
4. 对于每种类别的玻璃样品，分析其 14 种化学成分之间的关联关系，并比较两种类别玻璃关联关系的差异。

二、 基本假设

1. 同一类别的玻璃，风化过程是相似的；
2. 14 种化学成分之间存在相关性；
3. 由检测手段等原因导致的表单 2 数据误差影响较小；
4. 表单 2 中未注明风化情况的点的风化情况与该文物一致。

三、 符号说明

符号	符号含义
成分 0-13	14 种化学成分
$\boldsymbol{P}$	主成分分析矩阵
S	聚类轮廓系数

四、 问题的分析

问题 1 的分析

问题 1 首先要求分析玻璃表面风化与玻璃类型、纹饰和颜色三个属性之间的关系。由于数据不是很直观，我们首先根据风化属性与各属性的列联表分别绘制频数分布热力图，将其可视化。为了评估表面风化与各属性之间的关联关系，我们采用 Fisher 精确检验对其进行分类变量-分类变量的相关性分析。

接下来需要分析玻璃样品表面风化与未风化的化学成分统计规律。首先将数据可视化，分别绘制不同种类玻璃样品风化前后各个化学成分含量的箱型图。并将化学成分数据的各项统计学参数列成表格，进行分析。

最后需要预测所有风化点在风化前的化学成分含量。根据“同一类别的玻璃，风化过程是相似的”假设，提出标准化映射模型，将所有样本风化前后的化学成分含量用两种方法作标准化处理，接着将需要预测的风化后的数据均匀映射到风化前的标准化位置，最后进行逆标准化，即可得出预测的化合物含量。

问题 2 的分析

分析两种不同类型的玻璃的分类规律，意味着要找出玻璃分类与 14 种化学成分参数加上一个风化参数共 15 种参数之间的关系。由于参数较多，首先进行主成分分析，将 15 种参数降维至 3 个参数。接下来使用机器学习分类算法找出分类规律，本文运用两种分类算法，logistic 回归分类和随机森林分类，求出高钾与铅钡玻璃的分类规律。

接下来需要寻找合适的化学成分对其进行亚类划分。我们首先使用熵权法得出权重较大的化学成分，在其中进行选择，然后运用 Kmeans++ 聚类算法完成亚类划分的任务。

最后还要进行合理性与敏感性的分析。对于高钾-铅钡二分类任务，采用召回率、精确度和 F1 分数的指标来评估分类的准确性，说明其合理性；对于亚分类任务，采取轮

廓系数这一指标评估其聚类的合理性。采用对原数据增加扰动的方式检验其准确率，评估其敏感性。

问题 3 的分析

分析未知玻璃样品鉴别其类型与亚类，只需按照第二问的分类与聚类结果进行划分即可。与问题 2 相同，采用对原数据增加扰动的方式检验其准确率，评估其敏感性。

问题 4 的分析

问题 4 要求对两种类型的玻璃分别分析其化学成分之间的关联关系，并比较高钾与铅钡玻璃关联关系之间的差异。

首先列出皮尔逊相关系数矩阵，在这一步使用热力图将其可视化。评估出各化学成分间的相关性后，使用因子分析来分析各化学成分间的内在关联。分别对两种玻璃进行因子分析后，比对其中的差异即可。

五、 数据预处理

5.1 化学成分数据的成分性检验

表单 2 中的化学成分数据具有成分性，也就是各化学成分比例累加和应为 100%。由于检测误差等原因，导致成分比例累加和不一定为 100%，认为成分比例累加和介于 85%-105% 的数据为有效数据。

绘制成分比例累加和的频数分布直方图：

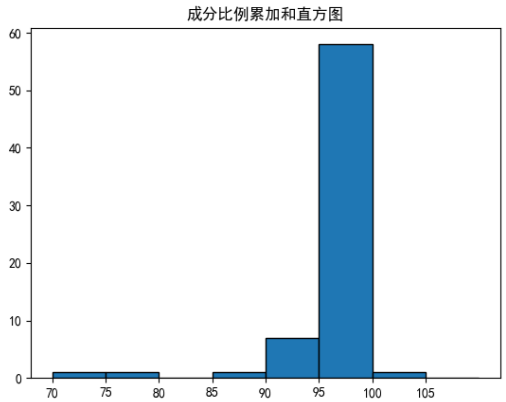

图 1　成分比例累加和频数分布直方图

从图中可以看出，大部分样品的成分数据都为有效数据，仅有 2 个样品不满足成分性（编号为 15,17），数据量较少，作剔除处理。

六、 问题一的模型建立与求解

6.1 文物玻璃基本信息的数据关系分析

6.1.1 模型准备

为了更好地分析玻璃表面风化与类型、纹饰和颜色的关系，分别对于每个属性参量绘制频数分布热力图（列联表）（颜色属性有缺失数据，作剔除处理）。结果如图2-4所示。

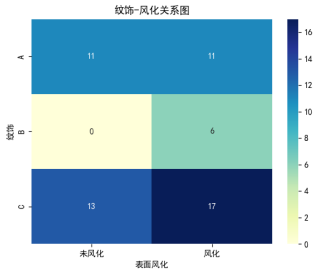

图 2　纹饰-风化关系图

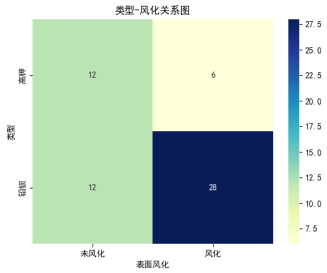

图 3　类型-风化关系图

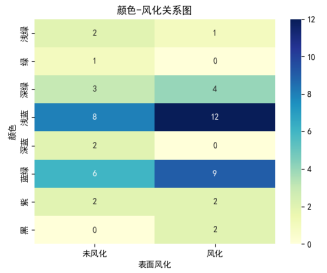

图 4　颜色-风化关系图

由图分析可知，风化与纹饰的关系为：出现纹饰 B 的样品表面均为风化，其余分布较为均衡；风化与类型的关系为：高钾玻璃多为未风化玻璃，铅钡玻璃多为风化玻璃；风化与颜色的关系不明显。

为了更准确地描述表面风化与各属性的关系，建立 Fisher 精确检验模型量化表面风化与类型、纹饰和颜色的关系。

Fisher 精确检验用于列联表的显著性差异检验，适合分析两种分类变量之间的相关性。与卡方检验相比，当列联表的某单元格中的期望频数低于 5 时，不适合用卡方检验，而对于 Fisher 精确检验，无论列联表特征如何，都可以使用此模型。观察图2-4，可以发现许多单元格频数都低于 5，故使用 Fisher 精确检验。

表 1　2*2 列联表示例

	特征 A.1	特征 A.2	行总和
特征 B.1	a	b	$a+b$
特征 B.2	c	d	$c+d$
列总和	$a+c$	$b+d$	n

如表1所示，对于 2*2 列联表，设原假设 (null hypothesis) 为两个分类特征不相关，

则原假设成立的概率为

$$p=\frac{C_a^{a+b}C_c^{c+d}}{C_{a+c}^n}=\frac{C_b^{a+b}C_d^{c+d}}{C_{b+d}^n}=\frac{(a+b)!(c+d)!(a+c)!(b+d)!}{a!b!c!d!n!} \tag{1}$$

其中 a,b,c,d 为（期望）频数。若 $p<\alpha=0.05$，即认为在显著性水平为 0.05（显著）时，我们可拒绝原假设，即两个分类特征不相关。

对于 2*C 列联表，SPSS 中可以使用蒙特卡洛模拟解出 Fisher 精确检验的 p 值.

6.1.2 Fisher 精确检验模型的建立与求解

分别对风化属性及类型、纹饰和颜色列 2*C 列联表，表格见附录B表10-12，或参考热力图2-4. 将数据导入 SPSS 软件中，得出每一个单元格中的期望频数，并计算出其 Fisher 精确检验的 p 值。结果为：p(类型) $=0.011<0.05$，**说明表面风化与类型的关系显著**；p(纹饰) $=0.089>0.05$，p(颜色) $=0.35>0.05$，说明纹饰和颜色对于表面风化的差异不显著，**认为表面风化与纹饰和颜色不相关**。

6.1.3 文物玻璃基本信息的数据关系结论

由 Fisher 精确检验模型可认为，表面风化与纹饰和颜色不相关；对于表面风化与玻璃类型的关系，由热力图分析可知高钾玻璃多为未风化玻璃，铅钡玻璃多为风化玻璃，关系较为显著。

由热力图可观察出现纹饰 B 的样品表面均为风化这一现象，但数据量较少无法肯定该结论。

6.2 样品各成分数据分析

对于两种不同类型的玻璃样品，分别绘制其风化前后各化学成分含量的箱型图。并使用 Excel 分析，给出化学成分的一些统计学数据。

对于高钾玻璃，结果如图5-6及表2所示。(图中 x 轴序号对应表中 14 种化学成分)

表 2　高钾玻璃统计数据

		二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)	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)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)
未风化	均值	67.98417	0.695	9.330833	5.3325	1.079167	6.62	1.931667	2.4525	0.411667	0.598333	1.4025	0.041667	0.196667	0.101667
	中位数	65.53	0	9.83	6.095	1.165	6.185	2.11	2.345	0.155	0	1.02	0.02	0	0
	方差	70.26411	1.518142	14.08804	8.766469	0.419074	5.690317	2.546381	2.525935	0.317997	0.884147	1.884885	0.002147	0.425456	0.031547
风化	均值	93.96333	0	0.543333	0.87	0.196667	1.93	0.265	1.561667	0	0	0.28	0	0	0
	中位数	93.505	0	0.665	0.83	0	1.72	0.275	1.545	0	0	0.28	0	0	0
	方差	2.504522	0	0.165156	0.198267	0.078189	0.7752	0.004025	0.728247	0	0	0.036733	0	0	0

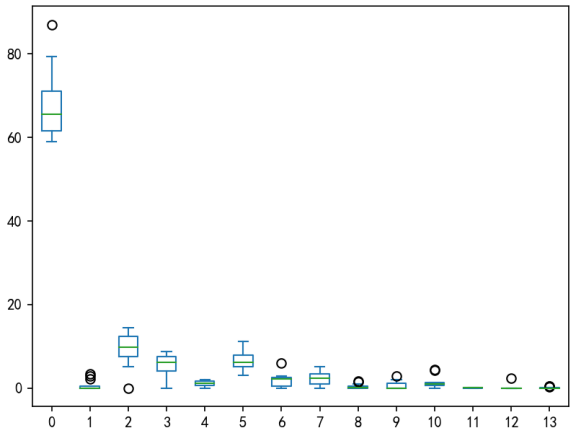

图 5　高钾玻璃未风化成分数据箱型图

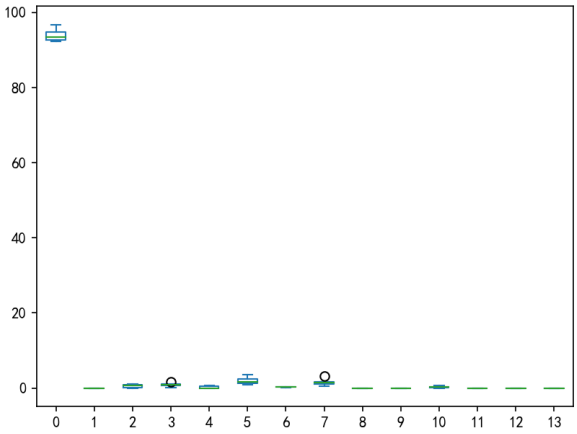

图 6　高钾玻璃风化成分数据箱型图

分析可知高钾玻璃样品化学成分含量具有如下统计规律：

- 高钾玻璃的主要成分为二氧化硅 (SiO2)，未风化的样品二氧化硅占比集中在 60-70%，风化后升至 90% 以上，且分布更为集中；
- 风化前的样品主要化学成分占比由高到低为：二氧化硅 (SiO2)(67.98%)、氧化钾 (K2O)(9.33)、氧化铝 (Al2O3)(6.62%)、氧化钙 (CaO)(5.33%)、氧化铜 (CuO)(2.45%)、氧化铁 (Fe2O3)(1.93%)、五氧化二磷 (P2O5)(1.40%)，括号内为该成分占比均值。其余成分占比均值均在 1% 以下，占比很少；
- 风化后的样品化学成分绝大多数都为二氧化硅，其余大部分成分都急剧降低，仅有氧化铝，氧化铜占比均值均在 1% 以上，其余成分占比均值均在 1% 以下，占比很少。

对于铅钡玻璃，结果如图7-8及表3所示。(图中 x 轴序号对应表中 14 种化学成分)

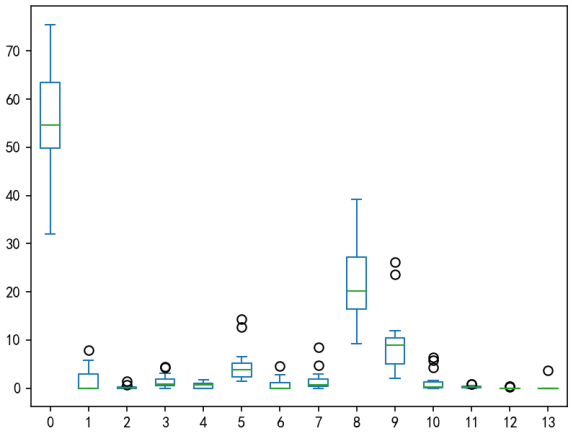

图 7　铅钡玻璃未风化成分数据箱型图

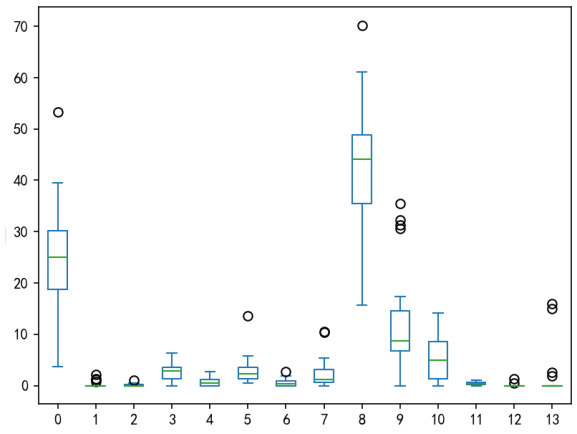

图 8　铅钡玻璃风化成分数据箱型图

表 3　铅钡玻璃统计数据

		二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)	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)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)
未风化	均值	54.65957	1.682609	0.218696	1.320435	0.640435	4.456087	0.736522	1.431739	22.08478	9.001739	1.04913	0.268261	0.046522	0.15913
	中位数	54.61	0	0.15	0.84	0.71	3.86	0	0.65	20.12	8.99	0.19	0.26	0	0
	方差	133.8323	5.380106	0.091959	1.578691	0.285943	10.18081	1.275431	3.711667	64.55436	32.45855	3.263286	0.056693	0.01551	0.557095
风化	均值	24.91269	0.216154	0.133462	2.695385	0.65	2.97	0.584615	2.275769	43.31385	11.80731	5.277308	0.418462	0.068462	1.366154
	中位数	25.015	0	0	2.875	0.57	2.38	0.305	1.145	44.06	8.79	4.975	0.425	0	0
	方差	108.1501	0.297939	0.055369	2.648909	0.479854	6.672469	0.521617	7.649378	143.8254	95.73644	16.9349	0.067444	0.069813	17.0106

分析可知铅钡玻璃样品化学成分含量具有如下统计规律：

- 铅钡玻璃的主要成分为二氧化硅 (SiO2) 和氧化铅 (PbO)；
- 未风化的样品二氧化硅占比集中在 50-65%，个体间差异较大，个别个体的占比可位于 30-80% 之间，风化后降至 20-30%，个别个体的占比可位于 0-55% 之间；
- 未风化的样品氧化铅占比集中在 15-30%，个体间差异较大，个别个体的占比可位于 5-40% 之间，风化后升至 35-50%，个别个体的占比可位于 10-70% 之间；
- 其余的化学成分中，氧化钙 (CaO)、氧化钡 (BaO)、五氧化二磷 (P2O5)、二氧化硫 (SO2) 的占比在风化过程后具有较为明显的上升趋势，而氧化纳 (Na2O)、氧化铝 (Al2O3) 具有较为明显的下降趋势，其余化学成分变化幅度较小。

6.3 预测风化前玻璃样品的化学成分

6.3.1 模型提出

本文提出标准化映射模型，将玻璃样品风化前后视为两个状态，对于两种不同类型的玻璃，分别将风化前与风化后各化学成分数据标准化，这时我们得到了 0-1 区间上的风化前后数据，且数据为 0 对应的样品为在其风化属性中化学成分含量最低的样品，数据为 1 对应的样品为化学成分含量最高的样品。我们认为在一般情况下，同一类型玻璃的风化过程是相似的，于是即可将风化后 0-1 区间上的数据均匀地映射至风化前的 0-1 区间，再逆标准化得到预测的风化前的化学成分含量。对于每个化学成分都作上述处理，即可预测出结果。

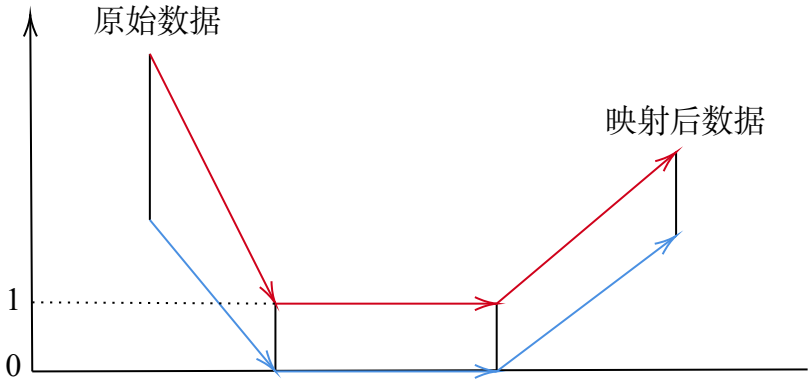

图 9　标准化映射模型原理

标准化映射模型的大致原理如图9所示。

6.3.2 标准化映射模型的建立与求解

本文采取两种方法进行标准化：

- 极差标准化

$$z_i = \frac{b_i - b_{i\min}}{b_{i\max} - b_{i\min}} \tag{2}$$

- z-score 标准化

$$z_i = \frac{b_i - \bar{b}}{\sigma} \tag{3}$$

其中 σ 为样本标准差

在使用 z-score 标准化法预测时，可能会出现预测结果小于 0 的情况，此时将其视为 0 即可。

高钾玻璃的预测数据如图10-11及表4所示。

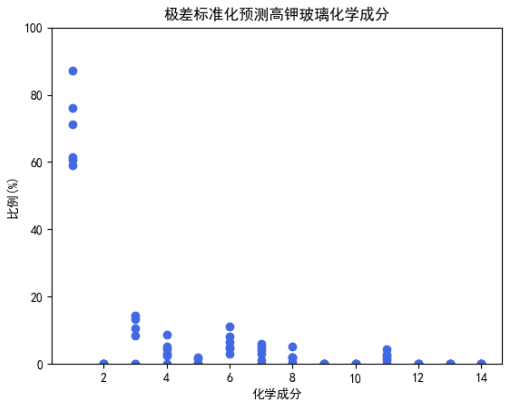

图 10　极差标准化法预测高钾玻璃

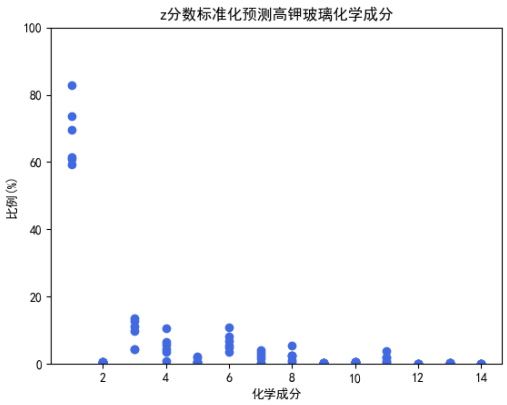

图 11　z-score 标准化法预测高钾玻璃

表 4　高钾玻璃预测结果数据

文物采样点	二氧化硅(SiO2)	氧化钠(Na2O)	氧化钾(K2O)	氧化钙(CaO)	氧化镁(MgO)	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)	氧化锶(SrO)	氧化锡(SnO2)	二氧化硫(SO2)
极差标准化法														
7	60.79	0	0	5.16	0	6.57	0	5.09	0	0	4.5	0	0	0
9	75.95	0	8.48	2.46	0	4.59	5.03	1.89	0	0	2.58	0	0	0
10	87.05	0	13.23	0	0	3.05	3.02	0.55	0	0	0	0	0	0
12	71.32	0	14.52	3.06	0	5.01	4.03	2.08	0	0	1.11	0	0	0
22	59.01	0	10.64	8.7	1.98	11.15	6.04	0	0	0	1.55	0	0	0
27	61.36	0	0	4.38	1.67	8.17	1.01	1.87	0	0	2.66	0	0	0
z-score 标准化法														
7	60.92	0.69	4.31	6.66	0.62	6.76	0	5.58	0.41	0.6	3.77	0.04	0.2	0.1
9	73.58	0.69	9.76	3.67	0.62	4.97	3.32	2.43	0.41	0.6	1.9	0.04	0.2	0.1
10	82.85	0.69	12.81	0.94	0.62	3.59	1.81	1.11	0.41	0.6	0	0.04	0.2	0.1
12	69.71	0.69	13.64	4.34	0.62	5.35	2.56	2.62	0.41	0.6	0.47	0.04	0.2	0.1
22	59.44	0.69	11.15	10.59	2.11	10.87	4.07	0.57	0.41	0.6	0.9	0.04	0.2	0.1
27	61.4	0.69	4.31	5.8	1.87	8.19	0.3	2.41	0.41	0.6	1.98	0.04	0.2	0.1

铅钡玻璃的预测数据如图12-13所示，详细数据表格数据量过大，见附录C表13-14。

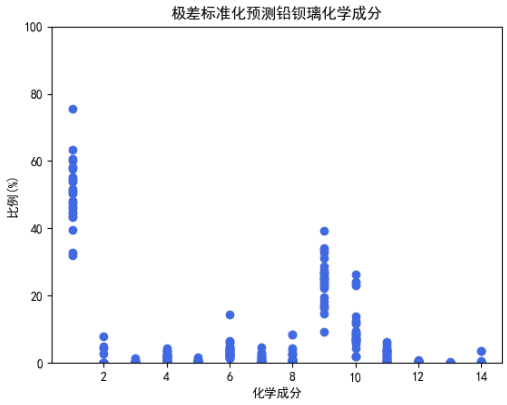

图 12　极差标准化法预测铅钡玻璃

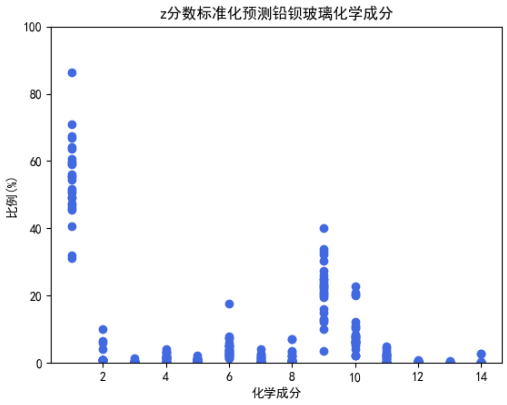

图 13　z-score 标准化法预测铅钡玻璃

观察预测结果，发现各化学成分的预测值符合未风化样品的化学成分统计规律，且极差标准化法与 z-score 标准化法预测的结果相差不大，但是 z-score 标准化法预测的结果极差更大，数据更为分散，我们在后文中均选用极差标准化方法进行标准化映射。

七、 问题二的模型建立与求解

7.1 分析高钾玻璃与铅钡玻璃的分类规律

7.1.1 使用主成分分析进行特征抽取

主成分分析方法 (Principal Component Analysis, PCA)，是一种数据降维算法，核心思想是将 n 维的特征数据映射到 k 维上，这 k 维特征被称为主成分，且两两正交。降维后的数据保留了大部分信息，简化了数据的分析与处理。

本题中，主成分分析的大致步骤如下：

设有 m 个 n 维样本 $x = (x_1, x_2, \ldots, x_m)$，目标为将数据将到 k 维.

1. 计算样本的协方差矩阵 $C = \frac{1}{m} X X^T$
2. 进行特征值分解或奇异值分解，求出协方差矩阵对应的特征向量和特征值
3. 将特征向量按特征值从大到小按行排成矩阵，取前 k 行得到主成分分析矩阵 $\boldsymbol{P}$.

将原 14 种化学成分与风化属性（未风化取 0，风化取 1）共 15 维的参数作为原始数据（使用 1-15 表示），使用 Python 的 sklearn 库进行主成分分析，得出解释方差比率选择主成分数量的关系如图14所示。

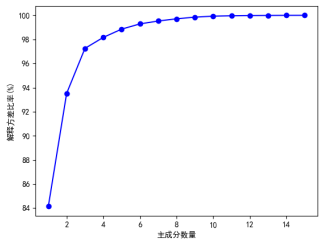

图 14　主成分数量选择

主成分个数为 3 时，解释方差比率达到 97.257%，超过 95%，已经包含了大部分原数据的信息。选取主成分个数为 3 作为各样本的维数。

同时，求解出主成分分析矩阵 P, 其解释各指标的信息载荷大小如图15-17所示。

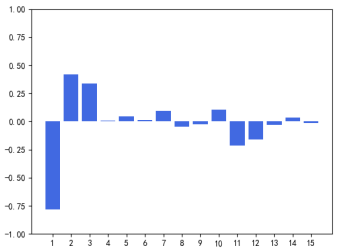

图 15　第一主成分载荷

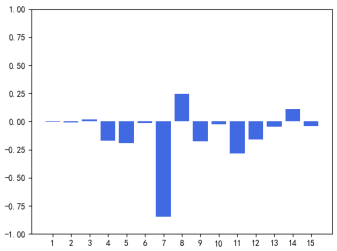

图 16　第二主成分载荷

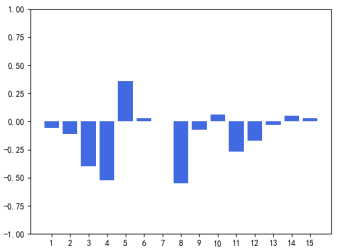

图 17　第三主成分载荷

由图可知，第一主成分中，载荷较高的化学成分有二氧化硅、氧化钾、氧化纳，可概括为 (SiO2,K2O,Na2O) 主成分；第二主成分中，载荷较高的化学成分有氧化铁、氧化铜、五氧化二磷，可概括为 (Fe2O3,CuO,P2O5) 主成分；第三主成分中，载荷较高的化学成分有氧化钾、氧化钙、氧化铜，可概括为 (K2O,CaO,CuO) 主成分。

7.1.2 运用机器学习分类算法求解分类规律

我们使用机器学习分类算法完成高钾/铅钡玻璃的二分类任务。本文采用两种方法对玻璃样本进行分类：

1. Logistic 回归分类算法.

   logistic 回归分类模型适合数值型为二值型的的分类。设 $L(\boldsymbol{x}, \boldsymbol{w}, b)$ 为 Logistic 回归分类函数，其值为 0 时，代表高钾玻璃，其值为 1 时，代表铅钡玻璃。$\boldsymbol{x}$ 为主成分分析后的三个玻璃样品参数. $\boldsymbol{w}, b$ 为所求的权重及偏差。

   所求分类规律可表示为

   $$L(\boldsymbol{P}\boldsymbol{x_0}, \boldsymbol{w}, b) \tag{4}$$

   运用 Python 的 sklearn 库中的 Logistic 回归分类算法分类，对 sklearn 库 Logistic 回归分类算法中影响最大的两个参数：正则化方式和正则化强度 C 进行调节，在 $(0, 2)$ 的范围内遍历正则化强度，调整参数，最终选择表5中效果较好的两种参数。结果如表5所示，其中 Weight 及 bias 即为权重及偏差 $\boldsymbol{w}, b$.

表 5　Logistic 回归分类结果

	训练集占比	正则化	正则化强度	结果
参数 1	0.7	l1	1.75	Weight=[-0.44504609 -0.15976295 -0.32174045] bias=[11.16641578] auc: 1.0 预测准确度: 1.0 1.0 precision recall f1-score support 0 1.00 1.00 1.00 18 1 1.00 1.00 1.00 49 accuracy 1.00 67 macro avg 1.00 1.00 1.00 67 weighted avg 1.00 1.00 1.00 67
参数 2	0.5	l1	1.75	Weight=[-0.45423507 -0.16312751 -0.35876258] bias=[11.2509573] auc: 1.0 预测准确度: 1.0 1.0 precision recall f1-score support 0 1.00 1.00 1.00 18 1 1.00 1.00 1.00 49 accuracy 1.00 67 macro avg 1.00 1.00 1.00 67 weighted avg 1.00 1.00 1.00 67

一般情况下，训练集占比参数为 0.7，为了检验其是否过拟合，调整参数值为 0.5，发现预测仍然良好，说明过拟合现象不明显。由结果可知，两种参数下的预测准确度都达到了 100%，说明 Logistic 回归分类效果好。

2. 随机森林分类算法.

运用 Python 的 sklearn 库中的随机森林分类算法分类，对 sklearn 库随机森林分类算法中影响最大的参数：决策树个数进行调节，选取效果较好的两种参数进行训练，结果如表6所示。当决策树个数为 1 时，模型预测准确度为 100%；当决策树个数为 5 时，模型预测准确度为 98.5%，结果如表5所示，可知两种参数下的预测准确度都接近 100%，说明随机森林分类效果较好，但不如 logistic 回归分类。

表 6　随机森林分类结果

	树的个数	结果
参数 1	1	auc: 1.0 预测准确度: 1.0 precision recall f1-score support 0 1.00 1.00 1.00 18 1 1.00 1.00 1.00 49 accuracy 1.00 67 macro avg 1.00 1.00 1.00 67 weighted avg 1.00 1.00 1.00 67
参数 2	5	auc: 0.9897959183673469 预测准确度: 0.9850746268656716 precision recall f1-score support 0 0.95 1.00 0.97 18 1 1.00 0.98 0.99 49 accuracy 0.99 67 macro avg 0.97 0.99 0.98 67 weighted avg 0.99 0.99 0.99 67

7.2 选择化学成分进行亚类划分

7.2.1 运用熵权法选择合适的化学成分

首先要选择合适的化学成分，这一步我们使用熵权法。

熵权法的基本思路是根据指标变异性的大小来确定客观权重，本题中，熵权法的具体的步骤为：

1. 对数据进行极差标准化；
2. 求每个化学成分下各样本含量所占的比重；

3. 求各指标的信息熵；

4. 通过信息熵计算各化学成分的权重。

由于风化前后的玻璃样品化学成分含量相差很大，我们想要去除这一影响，于是运用问题 1 中预测样品风化前化学成分含量的方法——极差标准化映射法将风化的样品各参数值还原回风化前的各参数值，设该映射函数为 $Y(x)$

计算各化学成分权重，结果如表7所示。

表 7　熵权法结果

	二氧化硅(SiO2)	氧化钠(Na2O)	氧化钾(K2O)	氧化钙(CaO)	氧化镁(MgO)	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)	氧化锶(SrO)	氧化锡(SnO2)	二氧化硫(SO2)
高钾	0.04	**0.14**	0.02	0.02	0.04	0.02	0.03	0.02	0.10	0.12	0.03	0.09	**0.22**	**0.14**
铅钡	0.01	**0.12**	0.08	0.03	0.05	0.04	0.07	0.05	0.01	0.03	0.05	0.03	**0.20**	**0.22**

对于高钾玻璃和铅钡玻璃，都选择氧化纳、氧化锡、二氧化硫 3 个权重较大的化学成分，在表7中加粗表示。

7.2.2 运用 Kmeans++ 聚类算法划分亚类

Kmens++ 聚类是 Kmeans 聚类的改进算法，其在选取初始化簇中心时做出了改进，使得距离上一个聚类簇中心点越远的样本点有更大的可能被选为下一个聚类簇中心。

运用 Python 的 sklearn 库中 Kmeans++ 聚类算法，以上文熵权法确定的化学成分数据为参数，将高钾玻璃分为 3 个亚类，用 0-2 表示，将铅钡玻璃分为 4 个亚类，用 0-3 表示。划分方式如图18-19所示，划分结果如表8所示。

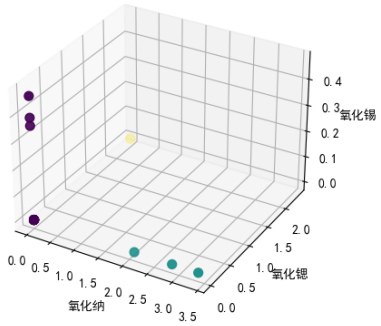

图 18　高钾玻璃亚分类划分方式

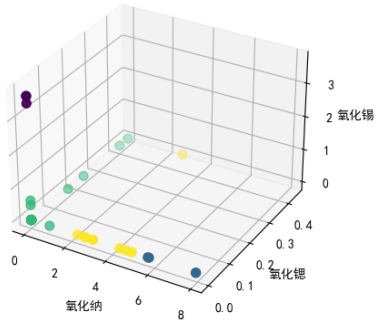

图 19　铅钡玻璃亚分类划分方式

表 8　亚类划分结果

高钾玻璃				铅钡玻璃							
未风化		风化		未风化				风化			
文物采样点	分类结果	文物采样点	分类结果	文物采样点	分类结果	文物采样点	分类结果	文物采样点	分类结果	文物采样点	分类结果
1	0	7	0	20	0	42-1	1	2	0	43-1	0
3	0	9	0	23	1	42-2	1	08*	0	43-2	0
3	0	10	0	24	0	44	2	8	3	48	2
4	0	12	0	25	2	45	2	11	0	49	0
5	0	22	0	28	0	46	0	19	0	50	0
6	0	27	0	29	0	47	2	26	0	51-1	0
6	0			30-1	0	49	0	26*	3	51-2	0
13	2			30-2	0	50	0	34	0	52	2
14	2			31	0	53	2	36	1	54	0
16	2			32	0	55	2	38	2	54*	0
18	1			33	0			39	0	56	0
21	0			35	0			40	0	57	0
				37	3			41	0	58	0

7.3 分析合理性及敏感性

7.3.1 高钾-铅钡玻璃分类准确度及合理性分析

二分类混淆矩阵一共有四个一级指标：真正类 TP(True Positive)、假负类 FN(False Negative)、假正类 FP(False Positive)、真负类 TN(True Negative)。在本题中，它们按顺序分别为：被预测为高钾玻璃的样本且实际为高钾玻璃的样本数量、被预测为铅钡玻璃的样本且实际为高钾玻璃的样本数量、被预测为高钾玻璃的样本且实际为铅钡玻璃的样本数量、被预测为铅钡玻璃的样本且实际为铅钡玻璃的样本数量。

表5中的二分类结果以二级指标召回率、精确度和三级指标 F1 分数作为评判分类预测效果的依据。

精确度 (Precision), 又称查准率，简记为 P，表示在被预测为正的样本中实际为正的样本比例：

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{5}$$

召回率 (recall)，又称查全率，简记为 R，表示在实际为正的样本中被预测为正样本的概率：

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6}$$

F1 分数 (F1 Score)，简记为 F_1

$$F_1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{\mathrm{TP}}{\mathrm{TP}+\frac{\mathrm{FN}+\mathrm{FP}}{2}} \tag{7}$$

这三个指标的取值范围都在 0 到 1 之间，越接近 1，说明模型的预测效果越好。

由表6可知，使用随机森林分类的三个指标接近 1，模型预测效果较好，使用 Logistic 回归分类的三个指标都为 1，模型的预测效果极好，说明结果具有合理性。

7.3.2 亚分类合理性分析

引入轮廓系数的指标，评价聚类结果的内聚度与分离度优劣。

轮廓系数的公式为：

$$S = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i - a_i}{max\{a_i, b_i\}} \tag{8}$$

其中 a_i 为簇内不相似度，等于同一分类下该点与其他点不相似程度的平均值，体现分类的内聚度，b_i 为簇间不相似度，等于该点到其他分类簇的平均不相似程度的最小值，体现分类的分离度。

轮廓系数的值在-1 与 1 之间，越接近 1，说明聚类效果越好。

对于两种不同类型的玻璃，不同聚类个数对应的轮廓系数如图20-21所示。

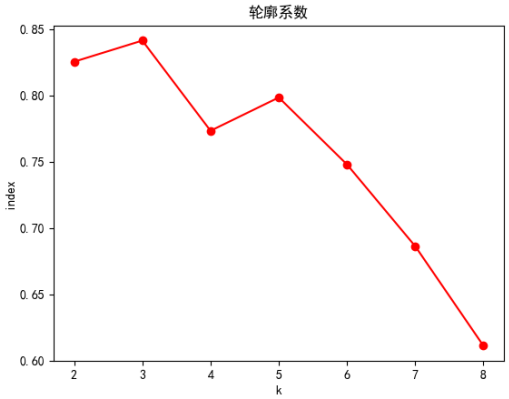

图 20　高钾玻璃聚类轮廓系数

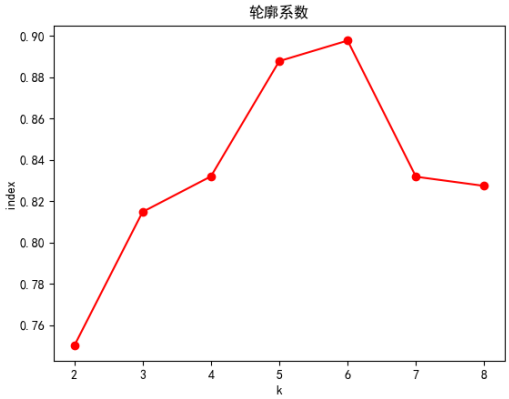

图 21　铅钡玻璃聚类轮廓系数

可见高钾玻璃聚类个数为 3 时，铅钡玻璃聚类个数为 4 时，轮廓系数都较高，聚类效果好，亚分类具有合理性。

7.3.3 分类及亚分类的敏感性分析

分析分类模型的敏感性，即对原数据增加扰动，观察扰动对分类结果的影响程度。由分类结果可知，Logistic 回归分类的效果更佳，对该方法的分类结果进行敏感性分析。

对附件中的原始数据，即化学成分含量百分比随机增加扰动再进行分类，设准确率为增加扰动后的结果与原结果相符的样本占所有样本的比例。

准确率与增加扰动大小的关系如图22-24所示。

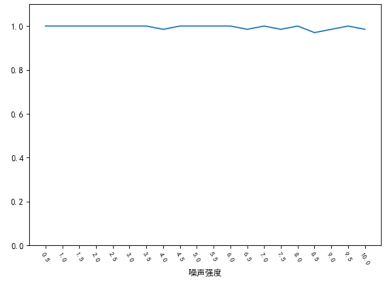

图 22　高钾-铅钡玻璃分类敏感性分析

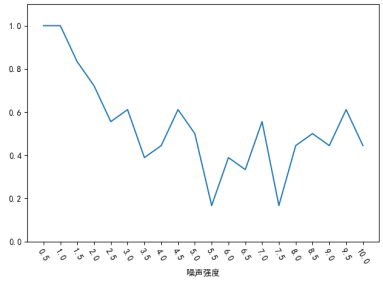

图 23　高钾玻璃亚分类敏感性分析

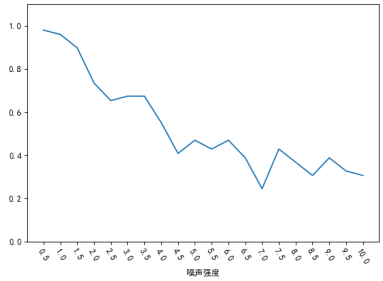

图 24　铅钡玻璃亚分类敏感性分析

由敏感性分析的结果可知，Logistic 回归分类的结果较为稳定，敏感性较低即使对化学成分含量增加 10% 的噪声扰动，也能保持较高的准确率。而 K-means++ 聚类算法对亚类的划分对于噪声扰动较为敏感，随噪声的增加，准确率迅速降低，敏感性较高。

八、 问题三的模型建立与求解

8.1 鉴定未知玻璃文物的类型

运用问题 2 中高钾与铅钡玻璃的分类算法及划分亚类的聚类算法结果对未知玻璃文物进行分类，结果如表9所示.

表 9　分类结果

文物编号	分类结果					
	Logistic 回归分类		随机森林分类		亚类划分	最终结果
	参数 1	参数 2	参数 1	参数 2		
A1	高钾	高钾	高钾	高钾	0	高钾-0
A2	铅钡	铅钡	铅钡	铅钡	0	铅钡-0
A3	铅钡	铅钡	铅钡	铅钡	0	铅钡-0
A4	铅钡	铅钡	高钾	铅钡	0	铅钡-0
A5	铅钡	铅钡	铅钡	铅钡	0	铅钡-0
A6	高钾	高钾	高钾	高钾	0	高钾-0
A7	高钾	高钾	高钾	高钾	0	高钾-0
A8	铅钡	铅钡	铅钡	铅钡	3	铅钡-3

8.2 分析合理性及敏感性

同样地，对原化学成分数据随机增加扰动，使用 Logistic 回归分类计算其受扰动后预测的准确度，结果如图25所示。

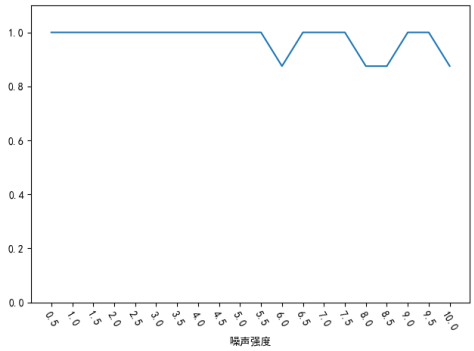

图 25　问题 3 高钾-铅钡玻璃敏感性分析

由图可知，高钾-铅钡玻璃的分类准确率受扰动影响较小，分类的结果较为稳定，敏感性较低。

九、 问题四的模型建立与求解

为了探究不同类别化学成分之间的关联关系，首先绘制皮尔逊相关系数矩阵热力图，如图26-27所示，其中红色代表正相关，蓝色代表负相关。

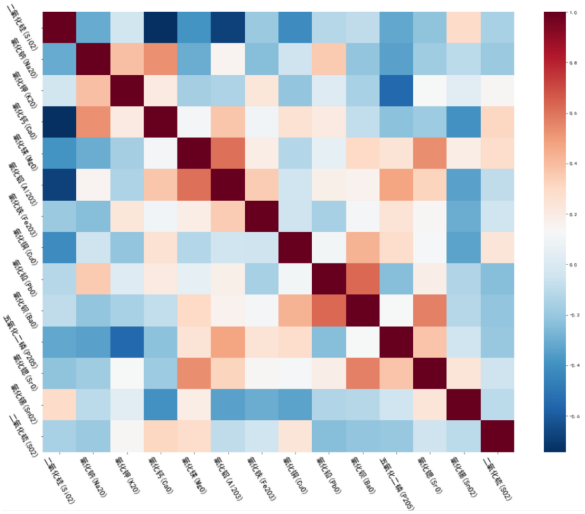

图 26　高钾玻璃相关系数矩阵热力图

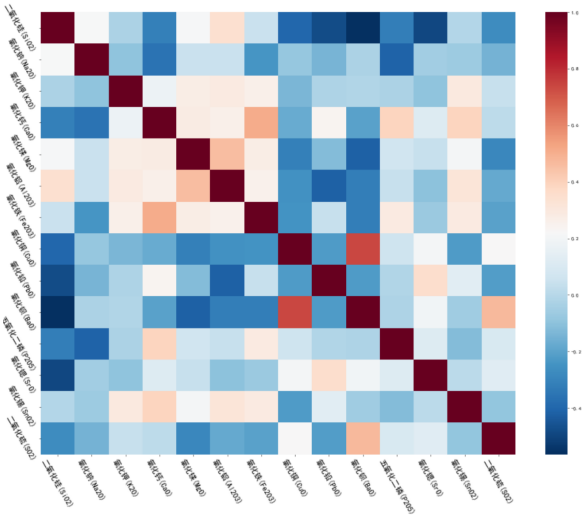

图 27　铅钡玻璃相关系数矩阵热力图

由图可知，无论是高钾玻璃还是铅钡玻璃，都有部分化学成分之间存在较强的正相关或负相关关系。因此有必要进行进一步的分析。

因子分析是从一个变量群中提取共性潜藏因子的技术，可以进一步利用 SPSS 软件对不同类别所有化学成分分别以相同设定进行因子分析：

Bartlett 球型度可用于检验相关阵是否为单位阵，即检验各个变量是否各自独立，若其 p 值小于 0.05，则说明适合作因子分析。

对于高钾玻璃，Bartlett 球型度 p 值为 0.00<0.05，公因子方差提取成分显示该方法对每一种化学成分的信息提取均高于 70%，因此可以继续分析。观察旋转成分矩阵，以 0.6 作为阈值，6 个主成分的代表成分分别为（氧化钙、氧化钠、氧化铝）、（氧化镁、氧化锶）、（氧化钾）、（氧化铅、氧化钡）、（氧化铁）、（二氧化硫）。

对于铅钡玻璃，同样通过了 Bartlett 球型度检验。观察旋转成分矩阵，以 0.6 作为阈值，5 个主成分的代表成分分别为（氧化钡、氧化铜）、（五氧化二磷、氧化钙）、（氧化锶）、（氧化铝、氧化镁）、（氧化钾、氧化锡）。属于同一主成分的化学成分之间具有较强的内在关联，而属于不同主成分的化学成分之间关联较弱。

对比高钾和铅钡两种不同类型玻璃的分析结果，可以看出高钾玻璃与铅钡玻璃化学成分之间关联有很大不同。比如在高钾玻璃中，氧化钙与氧化钠具有较高的关联性，而在铅钡玻璃中则不然。

从化学与历史的角度解释，以氧化钙与氧化钠的关系为例，东南亚出土的古代高钾玻璃中，氧化钙与氧化钠的相关关系较明显。而在中国边缘地区出土的铅钡玻璃中，钠含量几乎检测不到，而氧化钙含量仍有明显波动，二者无明显相关关系 [1]，由此一定程度上佐证了我们的研究结果。

十、 模型的评价与推广

10.1 模型的优点

1. 本文在建模时进行了多方案对比分析，并且选取了较优的 Fisher 精确检验、logistic 回归等模型和方法，说服力较强；
2. 熵权法指导 Kmeans++ 聚类，极大减少了选取化学成分划分亚类时的主观性，使结果更加贴近实际情况；
3. 本文使用较多可视化结果，使结论清晰明显，易于分析；
4. 在分类模型中，使用主成分分析对数据进行降维处理，从而简化处理过程，并使结果更准确。

10.2 模型的缺点

1. 机器学习过程中，一些影响不显著的参数未进行调节，结果可能存在偏差；
2. 模型有一定的超参数，如控制值参数等，具有一定的主观性。

10.3 模型的推广

如果原始数据的表达形式是含量多少而不是占比，本模型同样可以进行求解，此外如果数据量足够大，结果也将更加精确。

参考文献

[1] 干福熹, 承焕生, 李青会. 中国古代玻璃的起源——中国最早的古代玻璃研究 [J]. 中国科学：E 辑,2007,37(3):382-391

附录 A　支撑材料文件列表

国赛文件

python文件

（问题1相关代码及数据）

数据预处理.py
问题一数据分析.py
问题一预测.py

b1.xlsx 附件表一的数据
b2.xlsx 附件表二的数据
b3.xlsx 附件表三的数据
直方图.png 成分比例累加和直方图
combine_data.xlsx 经预处理的数据
new_b2.xlsx 删去异常项的表二数据
纹饰 - 风化.png 纹饰 - 风化热力图
类型 - 风化.png 类型 - 风化热力图
颜色 - 风化.png 颜色 - 风化热力图

问题一合并数据.xlsx 附件表一、二提取的数据
C1.xlsx 未风化高钾数据
C2.xlsx 风化高钾数据
C3.xlsx 未风化铅钡数据
C4.xlsx 风化铅钡数据

z_c2.xlsx 高钾Z分数映射数据
z_c4.xlsx 铅钡Z分数映射数据
m_c2.xlsx 高钾极差标准化映射数据
m_c4.xlsx 铅钡极差标准化映射数据
m_c2.png 高钾极差标准化映射可视化图
z_c2.png 铅钡极差标准化映射可视化图
m_c4.png 高钾Z分数映射可视化图
z_c4.png 铅钡Z分数映可视化图

（问题二、三相关代码及数据）

问题二、三分类.py
问题二、三亚类划分.py

主成分.xlsx 主成分分析所得数据
主成分分析参数选择.png

问题二logistic敏感性分析.png
问题三logistic敏感性分析.png
问题二高钾数据.xlsx
问题二铅钡数据.xlsx
问题三表三数据.xlsx
F1贡献.png 主成分分析贡献度直方图
F2贡献.png
F3贡献.png
ROC曲线.png 分类ROC曲线图
高钾聚类图.png
铅钡聚类图.png
高钾手肘.png 高钾聚类手肘法图
高钾轮廓.png 高钾聚类轮廓系数图
铅钡手肘.png 铅钡聚类手肘法图
铅钡轮廓.png 铅钡聚类轮廓系数图
问题二高钾亚类敏感性分析.png
问题二铅钡亚类敏感性分析.png

(问题四相关代码及数据)

问题四相关性分析.py
c1m2.xlsx 问题四高钾数据
c3mc4.xlsx 问题四铅钡数据
高钾相关系数.xlsx
铅钡相关系数.xlsx
高钾相关系数.png 高钾相关系数热力图
铅钡相关系数.png 铅钡相关系数热力图

spss文件

问题1 第一部分代码
问题4高钾
问题4铅钡

附件.xlsx
c1m2.xlsx 问题四高钾数据
c3mc4.xlsx 问题四铅钡数据

注：spss中读取excel数据使用相对路径'C:/Users/admin/Desktop',
请将文件放在桌面上运行或修改路径再运行。

纯代码（不含数据的代码，取自python文件与spss文件，需与数据在同一目录下才能运行）

图片（论文中的所有图片）

附录 B　问题 1 Fisher 精确检验列联表

表 10　纹饰-风化列联表

		表面风化		合计
		风化	无风化	
A	计数	11	11	22
	期望的计数	12.9	9.1	22.0
B	计数	6	0	6
	期望的计数	3.5	2.5	6.0
C	计数	17	13	30
	期望的计数	17.6	12.4	30.0
合计	计数	34	24	58
	期望的计数	34.0	24.0	58.0
Fisher 精确检验	**0.089**			

表 11　类型-风化列联表

		表面风化		合计
		风化	无风化	
高钾	计数	6	12	18
	期望的计数	10.6	7.4	18.0
铅钡	计数	28	12	40
	期望的计数	23.4	16.6	40.0
合计	计数	34	24	58
	期望的计数	34.0	24.0	58.0
Fisher 精确检验	**0.011**			

表 12　颜色-风化列联表

		表面风化		合计
		风化	无风化	
未知数据	计数	4	0	4
	期望的计数	2.3	1.7	4.0
黑	计数	2	0	2
	期望的计数	1.2	.8	2.0
蓝绿	计数	9	6	15
	期望的计数	8.8	6.2	15.0
绿	计数	0	1	1
	期望的计数	.6	.4	1.0
浅蓝	计数	12	8	20
	期望的计数	11.7	8.3	20.0
浅绿	计数	1	2	3
	期望的计数	1.8	1.2	3.0
深蓝	计数	0	2	2
	期望的计数	1.2	.8	2.0
深绿	计数	4	3	7
	期望的计数	4.1	2.9	7.0
紫	计数	2	2	4
	期望的计数	2.3	1.7	4.0
合计	计数	34	24	58
	期望的计数	34.0	24.0	58.0
Fisher 精确检验	0.35			

附录 C　问题 1 铅钡玻璃预测结果数据

表 13　铅钡玻璃预测结果数据 (一)

文物采样点[1]	二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)	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)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)
极差标准化法														
2	60.54	0	1.41	1.64	0.72	6.59	3.12	0.21	26.71	2.03	1.6	0.15	0	0
08*	46.36	0	0	1.04	0	2.29	0	8.33	16.42	23.35	1.61	0.3	0	0.59
8	32.72	0	0	2.24	0	2.07	0	2.51	18.49	22.93	3.39	0.43	0	3.45
11	58.17	0	0.28	2.46	0.43	3.61	0	3.95	14.61	12	4.21	0.3	0	0
19	54.7	0	0	2.06	0.36	4.47	2.23	2.81	24.18	5.68	3.96	0.15	0	0
26	46.05	0	0	1.01	0	1.66	0	8.46	16.89	24.05	1.4	0.37	0	0.45
26*	31.94	0	0.54	2.11	0	2.13	0	2.88	17.1	26.23	2.71	0.5	0	3.66
34	60.1	0	0.34	0.55	0	2.57	0.79	1.21	26.23	8.86	0.15	0.18	0	0
36	63.43	7.92	0.19	0.26	0	2.55	0.54	0.54	23.52	9.42	0.03	0.18	0	0
38	57.59	4.92	0	0.48	0	3.5	0.49	0.58	27.75	8.71	0.22	0.33	0	0
39	51.73	0	0	0.78	0	1.47	0	0.7	34.18	6.96	0.52	0.5	0	0
40	43.35	0	0	1.31	0	1.42	0.32	0	39.22	6.6	0.79	0.55	0	0
41	44.89	0	0.59	3.48	1.67	4.24	3	0.15	24.9	8.69	3.35	0.38	0	0
43-1	39.57	0	0	3.68	0.54	3.18	1.27	4.28	33.53	7.01	0	0.52	0	0
43-2	47.73	0	0	4.49	0.58	4.32	2.33	1.21	25.24	4.26	5.76	0.38	0	0
48	75.51	2.85	0.43	1.98	0.94	14.34	1.73	0	9.3	7.02	0.49	0.2	0.44	0
49	53.96	0	0	3.21	0.9	6.25	4.59	0.56	19.44	6.19	4.98	0.37	0	0
50	44.46	0	0	2.24	0.29	2.81	0.55	0.9	24.83	11.72	2.84	0.54	0	0
51-1	50.29	0	0	2.51	0.73	6.12	1.99	1.1	22.77	8.13	3.63	0.32	0.16	0
51-2	47.42	0	0	3.6	0.89	3.44	0.7	0.6	28.86	2.03	3.93	0	0	0
52	51.28	4.35	0	1.59	0.34	2.11	0.39	0.56	26.71	7.93	2.56	0.36	0	0
54	48.24	0	0.43	2.24	0.78	5.04	0	0.66	31.12	6.84	1.9	0.72	0	0
54*	43.7	0	0	0	0.68	4.55	0	1.07	32.77	2.03	6.34	0.91	0	0
56	54.27	0	0	0.85	0	2.79	0	0.63	23.32	12.58	1.14	0	0	0
57	51	0	0	0.92	0	3.11	0	0.93	25.43	13.84	0	0	0	0
58	55.36	0	0.46	2.45	0.48	4.42	1.44	2.51	22.28	7.26	4.03	0.2	0	0

[1] 标星号 (*) 的采样点为严重风化点，采样点后的”-1”,”-2” 表示部位 1,2.

表 14　铅钡玻璃预测结果数据 (二)

文物采样点	二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)	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)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)
z-score 标准化法														
2	67.3	0.76	1.4	1.05	1.05	7.87	2.73	0.03	24.84	2.13	0.3	0.06	0.01	0
08*	49.35	0.76	0.05	0.38	0.14	2.44	0	7.1	12.28	20.31	0.31	0.22	0.01	0.38
8	32.07	0.76	0.05	1.7	0.14	2.16	0	2.03	14.81	19.96	2.05	0.37	0.01	2.63
11	64.31	0.76	0.32	1.95	0.69	4.11	0	3.28	10.08	10.63	2.85	0.22	0.01	0
19	59.92	0.76	0.05	1.5	0.59	5.2	1.9	2.29	21.75	5.24	2.61	0.06	0.01	0
26	48.96	0.76	0.05	0.35	0.14	1.65	0	7.21	12.85	20.9	0.11	0.3	0.01	0.27
26*	31.08	0.76	0.56	1.56	0.14	2.25	0	2.35	13.11	22.77	1.38	0.45	0.01	2.8
34	66.75	0.76	0.37	0	0.14	2.79	0.56	0.9	24.25	7.95	0	0.09	0.01	0
36	70.96	10.2	0.23	0	0.14	2.76	0.32	0.32	20.94	8.43	0	0.09	0.01	0
38	63.58	6.63	0.05	0	0.14	3.96	0.28	0.35	26.1	7.83	0	0.26	0.01	0
39	56.15	0.76	0.05	0.1	0.14	1.41	0	0.46	33.95	6.33	0	0.44	0.01	0
40	45.53	0.76	0.05	0.68	0.14	1.34	0.12	0	40.1	6.02	0	0.51	0.01	0
41	47.48	0.76	0.61	3.07	2.25	4.9	2.62	0	22.62	7.81	2.01	0.32	0.01	0
43-1	40.75	0.76	0.05	3.28	0.83	3.57	1.01	3.57	33.16	6.37	0	0.47	0.01	0
43-2	51.09	0.76	0.05	4.18	0.87	5	2	0.9	23.05	4.02	4.36	0.32	0.01	0
48	86.27	4.16	0.46	1.42	1.33	17.65	1.43	0	3.59	6.38	0	0.11	0.63	0
49	58.97	0.76	0.05	2.78	1.27	7.43	4.11	0.33	15.97	5.68	3.61	0.31	0.01	0
50	46.95	0.76	0.05	1.7	0.5	3.1	0.34	0.63	22.54	10.39	1.52	0.49	0.01	0
51-1	54.32	0.76	0.05	2	1.06	7.27	1.68	0.8	20.03	7.33	2.29	0.24	0.24	0
51-2	50.7	0.76	0.05	3.2	1.26	3.89	0.48	0.37	27.46	2.13	2.57	0	0.01	0
52	55.58	5.95	0.05	0.99	0.56	2.22	0.18	0.33	24.84	7.16	1.24	0.29	0.01	0
54	51.73	0.76	0.46	1.7	1.13	5.91	0	0.42	30.22	6.23	0.59	0.69	0.01	0
54*	45.98	0.76	0.05	0	1	5.3	0	0.78	32.23	2.13	4.94	0.91	0.01	0
56	59.37	0.76	0.05	0.17	0.14	3.07	0	0.4	20.7	11.12	0	0	0.01	0
57	55.22	0.76	0.05	0.25	0.14	3.48	0	0.65	23.28	12.2	0	0	0.01	0
58	60.75	0.76	0.48	1.93	0.75	5.14	1.17	2.03	19.43	6.59	2.68	0.1	0.01	0

附录 D　数据预处理 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

if __name__ == "__main__":
    #####数据读取#####
    b1 = pd.read_excel("b1.xlsx")
    b2 = pd.read_excel("b2.xlsx")
    b3 = pd.read_excel("b3.xlsx")


    #####表一的扩大#####
    copy1 = np.array(b1.iloc[:,:5])
    copy2 = np.array(b1.iloc[:,:5])
    for i in  range(499):
        copy1 = np.vstack((copy1,copy2))
    Saveexcel(copy1,"copy.xlsx")

    b1 = np.array(b1.iloc[:,5:10],dtype= int) #表一数据
    #####筛选表2中无效数据#####
    b2 = np.array(b2.iloc[:,:],dtype= float) #表二数据
    print(b2)

    score = np.zeros([ len(b2),1]) #记录总和
    for i in  range( len(b2)):
         sum = 0
        for j in  range(1, len(b2[i])-1):
            if b2[i,j]>0:
                 sum =  sum + b2[i,j]
        score[i,0] =  sum
    #print(score)

    #####记录无效数据#####
```

```
del_num = []
k = 0 #记录无效个数
for i in  range( len(score)):
    if score[i,0] < 85 or score[i,0]> 105:
        k = k + 1
        print("无效的行数为:",i)
        del_num.append(i)
print("共有无效检测次数: ",k)

#####可视化#####
# 直方图
plt.figure()
nums, bins, patches = plt.hist(score, bins=(110 - 70) // 5,  range=(70, 110), edgecolor='k')
plt.xticks(np.arange(70, 110, 5))
plt.title("成分比例累加和直方图")
plt.savefig("直方图.png")

#####读取删改后的数据#####
b2 = pd.read_excel("new_b2.xlsx")
b2 = np.array(b2.iloc[:, :], dtype= float)  # 表二数据
print(b2)
for i in  range( len(b2)): #标签修改
    print(b2[i,b2.shape[1]-1])
    if b2[i,b2.shape[1]-1] <= 1.5:
        b2[i, b2.shape[1] - 1] = 0
    elif b2[i,b2.shape[1]-1] <= 2.5:
        b2[i, b2.shape[1] - 1] = 1
    else:
        b2[i, b2.shape[1] - 1] = 2
print(b2)

#####还原数据百分比#####
for i in  range( len(b2)):
    for j in  range(1,  len(b2[i])-1):
        if b2[i,j] > 0:
            b2[i,j] = (b2[i,j] / score[i,0]) * 100

print(b2)

#####合并数据#####
com = np.zeros([ len(b2),b2.shape[1] + b1.shape[1]])
for i in  range( len(b2)):
    for j in  range( len(b1)):
        if b2[i,0] == b1[j,0]:
            for k in  range(b2.shape[1] + b1.shape[1]):
                if k < b2.shape[1] and b2[i,k] > 0:
                    com[i,k] = b2[i,k]
```

```
                elif k < b2.shape[1]:
                    com[i, k] = 0
                else:
                    com[i,k] = b1[j,k-b2.shape[1]]
    print(com)
    Saveexcel(com,"combine_data.xlsx")
```

附录 E　问题一数据分析 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

def combine(x,y):
    '''
        合并数据
    '''
    res = np.zeros([ len(x),2])
    for i in  range( len(x)):
        res[i,0] = x[i]
        res[i,1] = y[i]
    return res

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

def construct(c,num,com):
    '''
        构建数据矩阵
    '''
    for i in  range( len(c)):
        for j in  range(c.shape[1]):
            c[i,j] = com[num[i],j]
    return c
```

```
if __name__ == "__main__":
    #####数据读取#####
    b1 = pd.read_excel("b1.xlsx")
    b1 = np.array(b1.iloc[:, 5:10], dtype= int)  # 表一数据

    b2 = pd.read_excel("new_b2.xlsx")
    b2 = np.array(b2.iloc[:, :], dtype= float)  # 表二数据
    for i in  range( len(b2)): #标签修改
        if b2[i, b2.shape[1]-1] <= 1.5:
            b2[i, b2.shape[1]-1] = 0
        elif b2[i, b2.shape[1]-1] <= 2.5:
            b2[i, b2.shape[1]-1] = 1
        else:
            b2[i, b2.shape[1]-1] = 2

    #print(b1)

    #####热力图#####
    pattern = b1[:,1] #纹饰
    kind = b1[:,2] #玻璃类型
    color = b1[:,3] #颜色
    wind = b1[:,4] #风化

    #纹饰 - 风化
    pic1 = np.zeros([3,2])
    data1 = combine(pattern,wind)
    for i in  range( len(data1)):
        u =  int(data1[i,0])
        v =  int(data1[i,1])
        pic1[u,v] = pic1[u,v] + 1
    plt.figure()
    ax = sns.heatmap(pic1,cmap="YlGnBu",annot=True)
    plt.title("纹饰-风化关系图")
    plt.xlabel("表面风化")
    plt.xticks([0.5,1.5],["未风化","风化"])
    plt.ylabel("纹饰")
    plt.yticks([0.5,1.5,2.5],["A","B","C"])
    plt.savefig("纹饰 - 风化.png")

    #类型 - 风化
    pic2 = np.zeros([2,2])
    data2 = combine(kind,wind)
    for i in  range( len(data2)):
        u =  int(data2[i,0])
        v =  int(data2[i,1])
        pic2[u,v] = pic2[u,v] + 1
    plt.figure()
```

```
ax = sns.heatmap(pic2, cmap="YlGnBu", annot=True)
plt.title("类型-风化关系图")
plt.xlabel("表面风化")
plt.xticks([0.5, 1.5], ["未风化", "风化"])
plt.ylabel("类型")
plt.yticks([0.5, 1.5], ["高钾", "铅钡"])
plt.savefig("类型 - 风化.png")

#颜色 - 风化
pic3 = np.zeros([8,2])
data3 = combine(color,wind)
for i in  range( len(data3)):
    u =  int(data3[i,0])
    if u == 0: #删去未知的颜色
        continue
    else:
        u = u - 1
    v =  int(data3[i,1])
    pic3[u,v] = pic3[u,v] + 1
plt.figure()
plt.title("颜色-风化关系图")
ax = sns.heatmap(pic3, cmap="YlGnBu", annot=True)
plt.xlabel("表面风化")
plt.xticks([0.5, 1.5], ["未风化", "风化"])
plt.ylabel("颜色")
plt.yticks([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5],
     ["浅绿","绿","深绿","浅蓝","深蓝","蓝绿","紫","黑"])
plt.savefig("颜色 - 风化.png")

#####统计规律分析#####
print(b2)
glass = np.zeros([ len(b2),1]) #类型
for i in  range( len(b2)):
    glass[i] = kind[ int(b2[i,0]) - 1]

label = np.zeros([ len(b2),1]) #风化
for i in  range( len(b2)):
    label[i] = b2[i,b2.shape[1] - 1]

com = np.zeros([ len(b2), b2.shape[1] + 1])
for i in  range( len(b2)):
    for j in  range(b2.shape[1] + 1):
        if j < b2.shape[1] and b2[i, j] > 0:
            com[i, j] = b2[i, j]
        elif j < b2.shape[1]:
            com[i, j] = 0
        else:
```

```
            com[i, j] = glass[i]
print(com)
Saveexcel(com,"问题一合并数据.xlsx")

#记录各分类数据
num_1 = []
num_2 = []
num_3 = []
num_4 = []
for i in  range( len(b2)):
    if glass[i] == 0 and label[i] == 0:
        num_1.append(i)
    elif glass[i] == 0 and label[i] >= 1:
        num_2.append(i)
    elif glass[i] == 1 and label[i] == 0:
        num_3.append(i)
    else:
        num_4.append(i)
print("类1个数：", len(num_1))
print("类2个数：", len(num_2))
print("类3个数：", len(num_3))
print("类4个数：", len(num_4))

#构建数据矩阵
c1 = np.zeros([ len(num_1),com.shape[1]])
c2 = np.zeros([ len(num_2),com.shape[1]])
c3 = np.zeros([ len(num_3),com.shape[1]])
c4 = np.zeros([ len(num_4),com.shape[1]])

c1 = construct(c1, num_1, com)
c2 = construct(c2, num_2, com)
c3 = construct(c3, num_3, com)
c4 = construct(c4, num_4, com)

Saveexcel(c1, "C1.xlsx")
Saveexcel(c2, "C2.xlsx")
Saveexcel(c3, "C3.xlsx")
Saveexcel(c4, "C4.xlsx")

#####绘制箱形图#####

# c1
df = pd.DataFrame(c1[:, 1:-2])
df.plot.box()
plt.title("c1")
plt.show()
```

```
    # c2
    df = pd.DataFrame(c2[:, 1:-2])
    df.plot.box()
    plt.title("c2")
    plt.show()

    # c3
    df = pd.DataFrame(c3[:, 1:-2])
    df.plot.box()
    plt.title("c3")
    plt.show()

    # c4
    df = pd.DataFrame(c4[:, 1:-2])
    df.plot.box()
    plt.title("c4")
    plt.show()
```

附录 F　问题一预测 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

if __name__ == "__main__":
    #####数据读取#####
    c1 = pd.read_excel("C1.xlsx")
    c2 = pd.read_excel("C2.xlsx")
    c3 = pd.read_excel("C3.xlsx")
    c4 = pd.read_excel("C4.xlsx")
```

```
c1 = np.array(c1.iloc[:, 2:-2])
c2 = np.array(c2.iloc[:, 2:-2])
c3 = np.array(c3.iloc[:, 2:-2])
c4 = np.array(c4.iloc[:, 2:-2])

#####预测#####
#z_score
z_c2 = np.zeros([ len(c2),c2.shape[1]])
for j in  range(c2.shape[1]):
    data1 = np.array(c2[:,j])
    data2 = np.array(c1[:,j])
    scaler1 = StandardScaler()  # 实例化
    scaler2 = StandardScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1,1))  # z_score标准化
    res2 = scaler2.fit_transform(data2.reshape(-1,1))
    result = scaler2.inverse_transform(res1)
    for i in  range( len(c2)):
        if result[i] > 0:
            z_c2[i,j] = result[i]
Saveexcel(z_c2,"z_c2.xlsx")

z_c4 = np.zeros([ len(c4), c4.shape[1]])
for j in  range(c4.shape[1]):
    data1 = np.array(c4[:, j])
    data2 = np.array(c3[:, j])
    scaler1 = StandardScaler()  # 实例化
    scaler2 = StandardScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1, 1))  # z_score标准化
    res2 = scaler2.fit_transform(data2.reshape(-1, 1))
    result = scaler2.inverse_transform(res1)
    for i in  range( len(c4)):
        if result[i] > 0:
            z_c4[i, j] = result[i]
Saveexcel(z_c4, "z_c4.xlsx")
#max_min
m_c2 = np.zeros([ len(c2), c2.shape[1]])
for j in  range(c2.shape[1]):
    data1 = np.array(c2[:, j])
    data2 = np.array(c1[:, j])
    scaler1 = MinMaxScaler()  # 实例化
    scaler2 = MinMaxScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1, 1))  # maxmin标准化
    res2 = scaler2.fit_transform(data2.reshape(-1, 1))
    result = scaler2.inverse_transform(res1)
    for i in  range( len(c2)):
        if result[i] > 0:
            m_c2[i, j] = result[i]
```

```
Saveexcel(m_c2, "m_c2.xlsx")

m_c4 = np.zeros([ len(c4), c4.shape[1]])
for j in  range(c4.shape[1]):
    data1 = np.array(c4[:, j])
    data2 = np.array(c3[:, j])
    scaler1 = MinMaxScaler()  # 实例化
    scaler2 = MinMaxScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1, 1))  # maxmin标准化
    res2 = scaler2.fit_transform(data2.reshape(-1, 1))
    result = scaler2.inverse_transform(res1)
    for i in  range( len(c4)):
        if result[i] > 0:
            m_c4[i, j] = result[i]
Saveexcel(m_c4, "m_c4.xlsx")

#####绘制散点图#####
plt.figure()
for i in m_c2:
    plt.scatter( range(1,15),i,c='royalblue')
plt.title("极差标准化预测高钾玻璃化学成分")
plt.xlabel("化学成分")
plt.ylabel("比例(%)")
plt.ylim(0,100)
plt.savefig("m_c2.png")

plt.figure()
for i in z_c2:
    plt.scatter( range(1, 15), i, c='royalblue')
plt.title("z分数标准化预测高钾玻璃化学成分")
plt.xlabel("化学成分")
plt.ylabel("比例(%)")
plt.ylim(0, 100)
plt.savefig("z_c2.png")

plt.figure()
for i in m_c4:
    plt.scatter( range(1, 15), i, c='royalblue')
plt.title("极差标准化预测铅钡璃化学成分")
plt.xlabel("化学成分")
plt.ylabel("比例(%)")
plt.ylim(0, 100)
plt.savefig("m_c4.png")

plt.figure()
for i in z_c4:
    plt.scatter( range(1, 15), i, c='royalblue')
```

```
    plt.title("z分数标准化预测铅钡玻璃化学成分")
    plt.xlabel("化学成分")
    plt.ylabel("比例(%)")
    plt.ylim(0, 100)
    plt.savefig("z_c4.png")
```

附录 G　问题二、三分类 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.decomposition import PCA

from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_decision_regions
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve,auc
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

class _PCA_():
    '''
        主成分分析
        X:样本特征
        y:样本标签
    '''
    def __init__(self,X,y = None):
        self.X = X
        self.y = y

    def n_choice(self):
        pca = PCA(n_components=None).fit(X)
        evr = pca.explained_variance_ratio_ * 100
        cov_matritx = pca.get_covariance()
        eigenvalue, featurevector = np.linalg.eig(cov_matritx)
        print("特征值: ", eigenvalue)
        print("特征向量: ", featurevector)
        # 绘制权重图
        feat = featurevector[:3,:]
        for i in feat:
```

```
            x =  range(1,16)
            plt.figure()
            plt.bar(x,i,color = 'royalblue')
            plt.ylim(-1,1)
            plt.xticks( range(1,16))
            plt.show()

        # 查看累计解释方差比率与主成分个数的关系
        fig, ax = plt.subplots()
        ax.plot(np.arange(1,  len(evr) + 1), np.cumsum(evr), "-bo")
        ax.set_xlabel("主成分数量")
        ax.set_ylabel("解释方差比率(%)")
        plt.savefig("主成分分析参数选择.png")
        num = 0
        time = 0
        for i in np.cumsum(evr):
            num = num + 1
            if i >= 85 and time == 0:
                print("主成分个数为{}，解释方差比率达到%{}". format(num,i))
                time = time + 1
            if i >= 90 and time == 1:
                print("主成分个数为{}，解释方差比率达到%{}". format(num, i))
                time = time + 1
            if i >= 95 and time == 2:
                print("主成分个数为{}，解释方差比率达到%{}". format(num, i))
                break

    def PCA(self,target = 0.95):
        ppp = PCA(n_components = target).fit(self.X)
        res = ppp.transform(self.X)
        self.pr = ppp
        print()
        print("原始矩阵大小：",self.X.shape)
        print("转换矩阵大小",res.shape)
        cov_matritx = ppp.get_covariance()
        Saveexcel(cov_matritx,"协方差矩阵.xlsx")
        print()
        return res

    def transor(self,X):
        res = self.pr.transform(X)
        return res

class logistics_classfier():
    '''
        logistics分类
        X:特征样本
```

```
        y:特征标签
        L:正则化方式，可选'l1','l2'
        C:惩罚参数,数值越小正则化越强
    '''
    def __init__(self,X,y,l = 'l2',C = 2):
        self.X = X
        self.y = y
        self.l = l
        self.C = C

    def __parameter__(self):
        '''选择参数'''
        #评价拟合效果
        l1_tr = []
        l1_te = []
        l2_tr = []
        l2_te = []
        Xtrain, Xtest, Ytrain, Ytest = train_test_split(self.X, self.y, test_size=0.3,
            random_state=20)
        #表现
        for i in np.linspace(0.05, 2, 38):
            l1_lr = LogisticRegression(penalty='l1',
                C=i,solver='liblinear',max_iter=1000).fit(Xtrain, Ytrain)
            l2_lr = LogisticRegression(penalty='l2',
                C=i,solver='liblinear',max_iter=1000).fit(Xtrain, Ytrain)
            l1_tr.append(accuracy_score(l1_lr.predict(Xtrain), Ytrain))
            l1_te.append(accuracy_score(l1_lr.predict(Xtest), Ytest))
            l2_tr.append(accuracy_score(l2_lr.predict(Xtrain), Ytrain))
            l2_te.append(accuracy_score(l2_lr.predict(Xtest), Ytest))
        g = [l1_tr, l2_tr, l1_te, l2_te]
        c = ['green', 'black', 'lightgreen', 'gray']
        lab = ['l1_train', 'l2_train', 'l1_test', 'l2_test']
        plt.figure()
        for i in  range( len(g)):
            plt.plot(np.linspace(0.05, 2, 38), g[i], c[i], label=lab[i])
        plt.legend(loc=4)
        plt.savefig("logistics参数选择.png")

    def __logistics__(self,test_size = 0.3):
        '''
            logistcs回归
            test_size: 测试集占比
        '''
        #划分测试集
        X_train, X_test, y_train, y_test = train_test_split(self.X, self.y, test_size=test_size,
            random_state=0)
        X_combined = np.vstack((X_train, X_test))
```

```
        y_combined = np.hstack((y_train, y_test))
        #分类
        if self.l == 'l1':
            solver = 'liblinear'
        else:
            solver = 'lbfgs'
        lr = LogisticRegression(penalty=self.l, C=self.C,solver=solver).fit(X_train, y_train)
        self.lr = lr
        #可能的画图
        NX = np.array(self.X)
        if NX.shape[1]<=2:
            plot_decision_regions(X_combined, y_combined, clf=lr)
            plt.legend(loc='upper left')
            plt.show()
        #参数
        w, b = lr.coef_[0], lr.intercept_
        print('Weight={}\nbias={}'. format(w, b))
        return w,b

    def __predict__(self,X):
        '''结果预测'''
        X = np.array(X)
        Y = self.lr.predict(X)
        print("预测值y:",Y)
        return Y

    def __evaluate__(self):
        '''结果评价'''
        y_pred = self.lr.predict(self.X)
        acc = accuracy_score(y_true=self.y, y_pred=y_pred)
        report = classification_report(self.y,y_pred)
        fpr, tpr, thresholds = roc_curve(self.y, y_pred)
        print("auc:",auc(fpr,tpr))
        print('预测准确度:',acc)
        plt.figure()
        roc_auc = auc(fpr,tpr)
        plt.title('ROC曲线')
        plt.plot(fpr, tpr, label=u'AUC = %0.3f' % roc_auc)
        plt.legend(loc='lower right')
        plt.plot([0, 1], [0, 1], 'r--')
        plt.xlim([-0.1, 1.1])
        plt.ylim([-0.1, 1.1])
        plt.ylabel('TP Rate')
        plt.xlabel('FP Rate')
        plt.grid(linestyle='-.')
        plt.grid(True)
        plt.savefig("ROC曲线")
```

```
        print(roc_auc)
        print(report)

class RFC_classifier():
    '''
        随机森林
        X:特征样本
        y:特征标签
        n_est:决策树个数
        depth:树的深度
    '''
    def __init__(self,X,y,n_est = 10,depth = 10):
        self.X = X
        self.y = y
        self.n_est = n_est
        self.depth = depth

    def __nstimators__(self,low = 10,up = 100,step = 10):
        '''交叉验证选取树的数量'''
        sco = []
        for i in  range(low, up, step):
            rfc = RandomForestClassifier(n_estimators=i,random_state=20,n_jobs=-1)
            score = cross_val_score(rfc, self.X, self.y, cv=10).mean()
            sco.append(score)
        print("最高准确率为{}". format( max(sco), (sco.index( max(sco)) * step + low)))
        plt.figure()
        plt.plot( range(low, up, step), sco)
        plt.savefig("随机森林参数选择.png")

    def __RFC__(self,test_size = 0.3):
        '''
            随机森林分类
            test_size: 测试集占比
        '''
        #划分测试集
        X_train, X_test, y_train, y_test = train_test_split(self.X, self.y, test_size=test_size,
            random_state=0)
        X_combined = np.vstack((X_train, X_test))
        y_combined = np.hstack((y_train, y_test))
        #分类
        lr =
            RandomForestClassifier(n_estimators=self.n_est,max_depth=self.depth,random_state=20).fit(X_train,
            y_train)
        self.lr = lr
        #可能的画图
        NX = np.array(self.X)
        if NX.shape[1]<=2:
```

```
            plot_decision_regions(X_combined, y_combined, clf=lr)
            plt.legend(loc='upper left')
            plt.show()

    def __predict__(self,X):
        '''结果预测'''
        X = np.array(X)
        Y = self.lr.predict(X)
        print("预测值y:",Y)
        return Y

    def __evaluate__(self):
        '''结果评价'''
        y_pred = self.lr.predict(self.X)
        acc = accuracy_score(y_true=self.y, y_pred=y_pred)
        report = classification_report(self.y, y_pred)
        fpr, tpr, thresholds = roc_curve(self.y, y_pred)
        print("auc:", auc(fpr, tpr))
        print('预测准确度:', acc)
        print(report)

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

def acc(y_true,y_pred):
    right = 0
    wrong = 0
    for i in  range( len(y_true)):
        if y_true[i] == y_pred[i]:
            right = right + 1
        else:
            wrong = wrong + 1
    return right / (right + wrong)

if __name__ == "__main__":
    #####数据读取#####
    data = pd.read_excel("问题一合并数据.xlsx")
    X = np.array(data.iloc[:,2:-1])
    y = np.array(data.iloc[:,-1])
    print(X)
    print(y)
```

```
#####主成分分析#####
dr = _PCA_(X)
dr.n_choice()
new_X = dr.PCA(3)
Saveexcel(new_X,"主成分.xlsx")

#####分类#####
# 随机森林

clf2 = RFC_classifier(new_X, y)
clf2.__nstimators__(1,10,1)
clf2.n_est = 5 #选择树的个数
clf2.__RFC__()
clf2.__evaluate__()

clf4 = RFC_classifier(new_X, y)
clf4.__nstimators__(1, 10, 1)
clf4.n_est = 1  # 选择树的个数
clf4.__RFC__()
clf4.__evaluate__()

#逻辑斯蒂
clf1 = logistics_classfier(new_X, y)
clf1.__parameter__()
clf1.__logistics__(0.7) #训练集占比0.7
clf1.l = 'l1' #选择l1正则化
clf1.C = 1.75
clf1.__evaluate__()

#验证
clf3 = logistics_classfier(new_X, y)
clf3.__logistics__(0.5) #训练集占比0.5
clf3.l = 'l1'
clf3.C = 1.75
clf3.__evaluate__()

#####结果预测#####
b3 = pd.read_excel("b3.xlsx")
b3 = np.array(b3.iloc[:,1:])

#处理数据
for i in  range( len(b3)):
    for j in  range(b3.shape[1]):
        if j == 0:
            if b3[i,j] == "无风化":
                b3[i,j] = 0
```

```
            else:
                b3[i,j] = 1
        else:
            if not b3[i,j] > 0:
                b3[i,j] = 0
print(b3.shape)
test = np.zeros([ len(b3),b3.shape[1]])
for i in  range( len(test)):
    for j in  range(test.shape[1]):
        if j < test.shape[1] - 1:
            test[i,j] = b3[i,j + 1]
        else:
            test[i,j] = b3[i,0]
b3_test = test

#降维
test = dr.transor(test)
print(test.shape)

#随机森林预测
res_rf1 = clf2.__predict__(test)
res_rf2 = clf4.__predict__(test)

#logistics预测
res_lr1 = clf1.__predict__(test)
res_lr2 = clf3.__predict__(test)

#####问题二敏感性分析#####
print("第二题敏感度分析")
y_true = y
loser = []
xser = []
np.random.seed(20)
for i in  range(20):
    u = i + 1
    eps = np.random.uniform(-u/2,u/2,(X.shape))
    epX = X + eps
    # 主成分
    ep_test = dr.transor(epX)
    #逻辑斯蒂
    y_pred = clf1.__predict__(ep_test)
    # 准确率
    loss = acc(y_true, y_pred)
    loser.append(loss)
    xser.append(u/2)
#可视化
plt.figure()
```

```
    plt.plot(xser,loser)
    plt.xticks(xser)
    plt.ylim(0,1.1)
    plt.xticks(rotation=300,fontsize=8)
    plt.xlabel("噪声强度")
    plt.tight_layout()
    plt.savefig("问题二logistic敏感性分析.png")

    #####问题三敏感性分析#####
    print("问题三敏感性分析")
    y_true = [0,1,1,1,1,0,0,1]
    loser = []
    xser = []
    np.random.seed(20)
    for i in  range(20):
        u = i + 1
        eps = np.random.uniform(-u/2,u/2,(b3_test.shape))
        epX = b3_test + eps
        # 主成分
        ep_test = dr.transor(epX)
        # 逻辑斯蒂
        y_pred = clf1.__predict__(ep_test)
        # 准确率
        loss = acc(y_true, y_pred)
        loser.append(loss)
        xser.append(u/2)
    #可视化
    plt.figure()
    plt.plot(xser,loser)
    plt.xticks(xser)
    plt.ylim(0,1.1)
    plt.xticks(rotation=300,fontsize=10)
    plt.xlabel("噪声强度")
    plt.tight_layout()
    plt.savefig("问题三logistic敏感性分析.png")
```

附录 H　问题二、三亚类划分 Python 代码

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

from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

def Vision(kind,data,name):
    draw = np.zeros([ len(kind),14])
    for i in  range( len(kind)):
        for j in  range(14):
            draw[i,j] = data[kind[i],j]
    plt.figure()
    for i in  range( len(kind)):
        plt.scatter( range(1, 15), draw[i], c='royalblue')
    plt.xticks( range(1,15))
    plt.xlabel("化学成分")
    plt.ylabel("占比(%)")
    plt.ylim(0,100)
    plt.savefig("亚类{}". format( str(name)))

class kmeans_cluster():
    '''
        kmeans聚类
        X:特征样本
        k:类别数量
    '''
    def __init__(self,X,k=3):
        self.X = X
        self.k = k

    def __kmeans__(self):
        '''聚类'''
        self.y_pred = KMeans(n_clusters=self.k).fit_predict(self.X)
        return self.y_pred

    def __kmeanspp__(self):
        '''kmeans++聚类'''
        kkk = KMeans(n_clusters=self.k,init='k-means++',random_state=20)
        self.y_pred = kkk.fit_predict(self.X)
```

```
        self.lr = kkk
        return self.y_pred

    def __elbow__(self, min = 2, max = 9):
        '''手肘法则确定聚类个数'''
        SSE = []  # 误差平方和
        index = [] # 轮廓系数
        for t in  range( min, max): #k的迭代范围
            estimator = KMeans(n_clusters=t,random_state=20,init="k-means++")  # 构造聚类器
            estimator.fit(self.X)
            SSE.append(estimator.inertia_) # estimator.inertia_获取聚类准则的总和
            y_est = KMeans(n_clusters=t).fit_predict(self.X)
            value = silhouette_score(self.X, y_est)
            index.append(value)
        t =  range( min, max)
        plt.xlabel('k')
        plt.ylabel('SSE')
        plt.plot(t,SSE,'o-')
        plt.title("手肘法则")
        plt.show()
        plt.xlabel('k')
        plt.ylabel('index')
        plt.plot(t,index,'ro-')
        plt.title("轮廓系数")
        plt.show()

    def __show3D__(self):
        fig2 = plt.figure()
        ax2 = plt.axes(projection='3d')
        ax2.scatter(self.X[:,0],self.X[:,1],self.X[:,2],c=self.y_pred,s=50)
        ax2.set_xlabel("氧化纳")
        ax2.set_zlabel("氧化锡")
        ax2.set_ylabel("氧化锶")

        plt.show()
        print("y_pred:",self.y_pred)

    def __predict__(self,X):
        X = np.array(X)
        y_pred = self.lr.predict(X)
        return y_pred

def E_j(data):
    '''
        计算熵值
    '''
    data = np.array(data)
```

```
    m,n = data.shape
    E = np.zeros([m,n])
    for i in  range(m):
        for j in  range(n):
            if data[i][j] == 0:
                e_ij = 0
            else:
                P_ij = data[i][j] / data. sum(axis=0)[j]  #计算比重
                e_ij = (-1 / np.log(m)) * P_ij * np.log(P_ij)
            E[i][j] = e_ij
    res=E. sum(axis=0)
    return res

def acc(y_true,y_pred):
    right = 0
    wrong = 0
    for i in  range( len(y_true)):
        if y_true[i] == y_pred[i]:
            right = right + 1
        else:
            wrong = wrong + 1
    return right / (right + wrong)

if __name__ == "__main__":
    #####数据读取#####
    c1 = pd.read_excel("C1.xlsx")
    c2 = pd.read_excel("m_c2.xlsx")
    c3 = pd.read_excel("C3.xlsx")
    c4 = pd.read_excel("m_c4.xlsx")

    c1 = np.array(c1.iloc[:, 2:-2])
    c2 = np.array(c2.iloc[:, 1:])
    c3 = np.array(c3.iloc[:, 2:-2])
    c4 = np.array(c4.iloc[:, 1:])

    print(c1.shape,c2.shape,c3.shape,c4.shape)

    #数据合并
    K = np.vstack((c1,c2))
    B = np.vstack((c3,c4))
    print(K.shape,B.shape)

    #####熵权法#####
    #标准化
    scaler1 = MinMaxScaler()  # 实例化
    res1 = scaler1.fit_transform(K)
```

```
scaler2 = MinMaxScaler()
res2 = scaler2.fit_transform(B)
#熵值
E1 = E_j(res1)
E2 = E_j(res2)
#差异系数
G1 = 1 - E1
G2 = 1 - E2
#权重
W1 = G1 /  sum(G1)
W2 = G2 /  sum(G2)

print("高钾权重",W1)
print("铅钡权重",W2)

#####成分筛选#####
K_choice = []
B_choice = []
for i in  range( len(W1)): #选择的阈值为0.12
    if W1[i] > 0.12:
        K_choice.append(i)
    if W2[i] > 0.11:
        B_choice.append(i)
print(K_choice)
print(B_choice)

data1 = np.zeros([ len(K), len(K_choice)])
data2 = np.zeros([ len(B), len(B_choice)])

for i in  range( len(K)):
    for j in  range( len(K_choice)):
        data1[i,j] = K[i,K_choice[j]]

for i in  range( len(B)):
    for j in  range( len(B_choice)):
        data2[i,j] = B[i,B_choice[j]]

Saveexcel(data1,"问题二高钾数据.xlsx")
Saveexcel(data2,"问题二铅钡数据.xlsx")
q1 = data1
q2 = data2

#####聚类#####
#kmeans++
print("聚类结果: ")
K_cl1 = kmeans_cluster(data1)
K_cl1.__elbow__()
```

```
K_cl1.k = 3 #聚类个数为3
K_res1 = K_cl1.__kmeanspp__()
K_cl1.__show3D__()
print(K_res1)

B_cl1 = kmeans_cluster(data2)
B_cl1.__elbow__()
B_cl1.k = 4
B_res1 = B_cl1.__kmeanspp__()
B_cl1.__show3D__()
print(B_res1)
print()

#####问题三预测#####
b3 = pd.read_excel("问题三表三数据.xlsx")
test = np.array(b3.iloc[:, 2:])
c1 = pd.read_excel("C1.xlsx")
c2 = pd.read_excel("C2.xlsx")
c3 = pd.read_excel("C3.xlsx")
c4 = pd.read_excel("C4.xlsx")

c1 = np.array(c1.iloc[:, 2:-2])
c2 = np.array(c2.iloc[:, 2:-2])
c3 = np.array(c3.iloc[:, 2:-2])
c4 = np.array(c4.iloc[:, 2:-2])
#筛选类别
wind = [] #是否风化
no_wind = []
for i in  range( len(test)):
    if test[i,-2] == 0:
        no_wind.append(i)
    else:
        wind.append(i)
print(no_wind)
print(wind)

kind2 = [] #玻璃类别
kind4 = []
for i in  range( len(test)):
    if test[i,-1] == 0:
        kind2.append(i)
    else:
        kind4.append(i)
print(kind2)
print(kind4)

t2 = [] #组合类别
```

```
t4 = []
for i in wind:
    if i in kind2:
        t2.append(i)
    else:
        t4.append(i)
print(t2)
print(t4)
t3 = [2,3,7]
#分开类别
test1 = np.zeros([1,14])
test2 = np.zeros([ len(t2),14])
test3 = np.zeros([3,14])
test4 = np.zeros([ len(t4),14])
for i in  range(2):
    for j in  range(14):
        if test[t2[i],j] > 0:
            test2[i, j] = test[t2[i],j]
        else:
            test2[i, j] = 0
        if test[t4[i],j] > 0:
            test4[i, j] = test[t4[i],j]
        else:
            test4[i, j] = 0
for j in  range(14):
    if test[0,j] > 0:
        test1[0,j] = test[0,j]
for i in  range(3):
    for j in  range(14):
        if test[t3[i],j] > 0:
            test3[i,j] = test[t3[i],j]
print(test1)
print(test2)
print(test3)
print(test4)

#映射
# max_min
m_c2 = np.zeros([2,14])
for j in  range(c2.shape[1]):
    data1 = np.array(c2[:, j])
    data2 = np.array(c1[:, j])
    data = np.array(test2[:,j])
    scaler1 = MinMaxScaler()  # 实例化
    scaler2 = MinMaxScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1, 1))  # maxmin标准化
    res2 = scaler2.fit_transform(data2.reshape(-1, 1))
```

```
    res = scaler1.transform(data.reshape(-1,1))
    result = scaler2.inverse_transform(res)
    for i in  range(2):
        m_c2[i, j] = result[i]
print(m_c2)

m_c4 = np.zeros([2,14])
for j in  range(c4.shape[1]):
    data1 = np.array(c4[:, j])
    data2 = np.array(c3[:, j])
    data = np.array(test2[:, j])
    scaler1 = MinMaxScaler()  # 实例化
    scaler2 = MinMaxScaler()
    res1 = scaler1.fit_transform(data1.reshape(-1, 1))  # maxmin标准化
    res2 = scaler2.fit_transform(data2.reshape(-1, 1))
    res = scaler1.transform(data.reshape(-1,1))
    result = scaler2.inverse_transform(res)
    for i in  range(2):
        m_c4[i, j] = result[i]
print(m_c4)
#成分筛选
pre1 = np.zeros([1,  len(K_choice)])
pre2 = np.zeros([2,  len(K_choice)])
pre3 = np.zeros([3,  len(B_choice)])
pre4 = np.zeros([2,  len(B_choice)])

for i in  range(1):
    for j in  range( len(K_choice)):
        pre1[i, j] = test1[i, K_choice[j]]

for i in  range(2):
    for j in  range( len(K_choice)):
        pre2[i, j] = test2[i, K_choice[j]]

for i in  range(3):
    for j in  range( len(B_choice)):
        pre3[i, j] = test3[i, B_choice[j]]

for i in  range(2):
    for j in  range( len(B_choice)):
        pre4[i, j] = test4[i, B_choice[j]]

print(pre1)
print(pre2)

#聚类
print("预测结果: ")
```

```
res1 = K_cl1.__predict__(pre1)
print(res1)
res2 = K_cl1.__predict__(pre2)
print(res2)
res3 = B_cl1.__predict__(pre3)
print(res3)
res4 = B_cl1.__predict__(pre4)
print(res4)
'''
#####聚类结果可视化#####
print(K_res1)
print(B_res1)
#高钾
k0 = []
k1 = []
k2 = []
k3 = []
k4 = []
for i in range(len(K_res1)):
    if K_res1[i] == 0:
        k0.append(i)
    elif K_res1[i] == 1:
        k1.append(i)
    elif K_res1[i] == 2:
        k2.append(i)
    elif K_res1[i] == 3:
        k3.append(i)
    elif K_res1[i] == 4:
        k4.append(i)
Vision(k0,K,'k0')
Vision(k1,K,'k1')
Vision(k2,K,'k2')
Vision(k3,K,'k3')
Vision(k4,K,'k4')

#钡铅
k0 = []
k1 = []
k2 = []
k3 = []
k4 = []
k5 = []
for i in range(len(B_res1)):
    if B_res1[i] == 0:
        k0.append(i)
    elif B_res1[i] == 1:
        k1.append(i)
```

```
        elif B_res1[i] == 2:
            k2.append(i)
        elif B_res1[i] == 3:
            k3.append(i)
        elif B_res1[i] == 4:
            k4.append(i)
        else:
            k5.append(i)
    Vision(k0, B, 'b0')
    Vision(k1, B, 'b1')
    Vision(k2, B, 'b2')
    Vision(k3, B, 'b3')
    Vision(k4, B, 'b4')
    Vision(k5, B, 'b5')
    '''
    #####问题二敏感性分析#####
    print("问题二敏感性分析")
    y_true = K_res1
    loser = []
    xser = []
    np.random.seed(20)

    for i in  range(20):
        u = i + 1
        eps = np.random.uniform(-u / 2, u / 2, (q1.shape))
        epX = q1 + eps
        # 聚类
        y_pred = K_cl1.__predict__(epX)
        # 准确度
        loss = acc(y_true, y_pred)

        loser.append(loss)
        xser.append(u / 2)
    # 可视化
    plt.figure()
    plt.plot(xser, loser)
    plt.xticks(xser)
    plt.ylim(0, 1.1)
    plt.xticks(rotation=300, fontsize=10)
    plt.xlabel("噪声强度")
    plt.tight_layout()
    plt.savefig("问题二高钾亚类敏感性分析.png")

    y_true = B_res1
    loser = []
    xser = []
    np.random.seed(20)
```

```
    for i in  range(20):
        u = i + 1
        eps = np.random.uniform(-u / 2, u / 2, (q2.shape))
        epX = q2 + eps
        # 聚类
        y_pred = B_cl1.__predict__(epX)
        # 准确度
        loss = acc(y_true, y_pred)

        loser.append(loss)
        xser.append(u / 2)
    # 可视化
    plt.figure()
    plt.plot(xser, loser)
    plt.xticks(xser)
    plt.ylim(0, 1.1)
    plt.xticks(rotation=300, fontsize=10)
    plt.xlabel("噪声强度")
    plt.tight_layout()
    plt.savefig("问题二铅钡亚类敏感性分析.png")
```

附录 I　问题四相关性分析 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False

def Saveexcel(data,filename):
    '''
        保存excel
    '''
    data = pd.DataFrame(data)
    writer = pd.ExcelWriter(filename)
    data.to_excel(writer, "page_1", float_format='%.2f')
    writer.save()

if __name__ == "__main__":
    #####数据读取#####
    c1c2 = pd.read_excel("c1mc2.xlsx")
    c3c4 = pd.read_excel("c3mc4.xlsx")
```

```
title =  list(c1c2)
title = np.array(title)
print(title)

c1c2 = np.array(c1c2)
c3c4 = np.array(c3c4)

print(c1c2)
print(c3c4.shape)

#####计算相关系数#####
cof1 = np.corrcoef(c1c2.T)
cof2 = np.corrcoef(c3c4.T)
print(cof1.shape)
print(cof2.shape)
Saveexcel(cof1,"高钾相关系数.xlsx")
Saveexcel(cof2,"铅钡相关系数.xlsx")

#####绘制热力图#####
plt.figure(figsize=(20,15),dpi=50)
ax = sns.heatmap(cof1,cmap="RdBu_r")
plt.xticks([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5], title)
plt.yticks([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5], title)
plt.xticks(rotation=300,fontsize=15)
plt.yticks(rotation=300,fontsize=15)
plt.savefig("高钾相关系数.png")

plt.figure(figsize=(20, 15), dpi=50)
ax = sns.heatmap(cof2, cmap="RdBu_r")
plt.xticks([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5], title)
plt.yticks([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5], title)
plt.xticks(rotation=300, fontsize=15)
plt.yticks(rotation=300, fontsize=15)
plt.savefig("铅钡相关系数.png")
```

附录 J　问题 1 第一部分 SPSS 代码

```
GET DATA /TYPE=XLSX
/FILE='C:\Users\admin\Desktop\附件.xlsx'
/SHEET=name '表单1'
/CELLRANGE=full
/READNAMES=on
/ASSUMEDSTRWIDTH=32767.
EXECUTE.
```

```
DATASET NAME 数据集1 WINDOW=FRONT.
CROSSTABS
  /TABLES=纹饰 类型 颜色 BY 表面风化
  /FORMAT=AVALUE TABLES
  /STATISTICS=CORR
  /CELLS=COUNT EXPECTED
  /COUNT ROUND CELL
  /METHOD=MC CIN(99) SAMPLES(10000).
```

附录 K　问题 4 高钾 SPSS 代码

```
  GET DATA /TYPE=XLSX
  /FILE='C:\Users\admin\Desktop\c1mc2.xlsx'
  /SHEET=name 'page_1'
  /CELLRANGE=full
  /READNAMES=on
  /ASSUMEDSTRWIDTH=32767.
EXECUTE.
DATASET NAME 数据集1 WINDOW=FRONT.
FACTOR
  /VARIABLES 二氧化硅SiO2 氧化钠Na2O 氧化钾K2O 氧化钙CaO 氧化镁MgO 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 氧化锶SrO 氧化锡SnO2 二氧化硫SO2
  /MISSING LISTWISE
  /ANALYSIS 二氧化硅SiO2 氧化钠Na2O 氧化钾K2O 氧化钙CaO 氧化镁MgO 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 氧化锶SrO 氧化锡SnO2 二氧化硫SO2
  /PRINT INITIAL CORRELATION KMO EXTRACTION ROTATION FSCORE
  /PLOT EIGEN ROTATION
  /CRITERIA MINEIGEN(1) ITERATE(25)
  /EXTRACTION PC
  /CRITERIA ITERATE(25)
  /ROTATION VARIMAX
  /SAVE REG(ALL)
  /METHOD=CORRELATION.
```

附录 L　问题 4 铅钡 SPSS 代码

```
  GET DATA /TYPE=XLSX
  /FILE='C:\Users\admin\Desktop\c3mc4.xlsx'
  /SHEET=name 'page_1'
  /CELLRANGE=full
  /READNAMES=on
  /ASSUMEDSTRWIDTH=32767.
EXECUTE.
```

```
DATASET NAME 数据集2 WINDOW=FRONT.
DATASET CLOSE 数据集1.
FACTOR
  /VARIABLES 二氧化硅SiO2 氧化钠Na2O 氧化钾K2O 氧化钙CaO 氧化镁MgO 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 氧化锶SrO 氧化锡SnO2 二氧化硫SO2
  /MISSING LISTWISE
  /ANALYSIS 二氧化硅SiO2 氧化钠Na2O 氧化钾K2O 氧化钙CaO 氧化镁MgO 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 氧化锶SrO 氧化锡SnO2 二氧化硫SO2
  /PRINT INITIAL CORRELATION KMO EXTRACTION ROTATION FSCORE
  /PLOT EIGEN ROTATION
  /CRITERIA MINEIGEN(1) ITERATE(25)
  /EXTRACTION PC
  /CRITERIA ITERATE(25)
  /ROTATION VARIMAX
  /SAVE REG(ALL)
  /METHOD=CORRELATION.
```