# 基于聚类分析方法的古玻璃制品成分综合分析评价模型


## 摘要

对古代玻璃制品成分的分析和类型鉴别在实际历史研究中具有广泛的使用场景和重要意义。本文主要通过研究一批古代玻璃文物的类型、颜色、成分组成等数据，分析找出相应的分类等统计规律，主要建立了成分综合分析评价模型，并使用**假设检验**、**关联分析**、**聚类分析**等研究方法对玻璃制品数据进行分析，通过敏感度、轮廓系数等指标对分析结果进行评价，利用 pycharm、matlab、SPSS 等软件进行求解。

**针对问题一**，首先根据玻璃纹饰、类型、颜色与是否风化的关系建立关联分析模型，利用支持度和置信度指标对集合覆盖过程进行筛选加速，利用 pycharm 软件进行求解，得到强关联关系，发现玻璃**类型为“铅钡”和纹饰类型为“B”**的玻璃在置信度为 1 的情况下会形成风化结果，而在三个因素共同影响下，**“B”+“高钾”+“蓝绿”和“C”+“铅钡”+“浅蓝”两个组合**都能在置信度为 1 的情况下确定玻璃“风化”。利用**两独立样本** $t$ **检验**分析方法分别对两类玻璃进行分析，根据检验结果和风化前后同一成分含量均值比较发现高钾玻璃中二氧化硅（$SiO_2$）在风化后含量明显增加，而铅钡玻璃中风化后二氧化硅（$SiO_2$）含量显著减少，氧化铅（$PbO$）含量增加，依据这些风化前后有显著变化的成分，建立相应风化前后含量差值对风化点风化前的含量进行预测。

**针对问题二**，首先使用 $t$ 检验分析找到两种类别的玻璃有显著差异的成分，两种玻璃成分分类规律主要是二氧化硅和氧化钾含量多的检测玻璃文物划分到高钾玻璃一组，而氧化钡和氧化铅含量相对多的玻璃文物划分到铅钡玻璃一组。采用**聚类分析**的方法，利用 pycharm 软件求解，重新将检测的玻璃样本划分，其中**高钾组细化分为两种亚类，铅钡组细分为四种亚类**。通过轮廓系数法对分类结果进行检验，高钾组和铅钡组聚类结果的轮廓系数分别为 0.6515 和 0.5406。最后通过人为加入噪音，比较加入噪音前后的分类结果及对应点的轮廓系数，发现在加入 5 组干扰数据时，分类结果保证了 95% 以上的准确性。

**针对问题三**，将未知玻璃类别的玻璃文物的检测数据与已完成聚类的玻璃样本进行混合，重新进行聚类，二次聚类结果就是未知类别玻璃最后的划分。逐渐改变未知类别样本的某一成分的含量，得到了主要成分二氧化硅和氧化铅分别变化了原始数值的 20.88% 和 57.31% 时会开始使分类结果发生改变的结果。

**针对问题四**，在满足显著性水平为 0.05 的情况下计算玻璃成分之间的相关系数，高钾玻璃二氧化硅与其他成分之间有不同程度的负相关关系或基本无相关关系，与氧化铅和氧化钡的相关系数分别为-0.618 和-0.649，铅钡玻璃氧化钡和氧化铅负相关关系较强，相关系数为-0.59，而与氧化铜具有较强正相关关系，相关系数为 0.699，从相关系数矩阵热力图总体而言高钾玻璃成分之间的相关性强于铅钡玻璃，差异性主要体现在氧化铅和氧化钡的正负相关性上。



**关键字：** 成分分析与鉴别　关联分析　聚类划分　噪音　敏感性分析

# 一、问题重述

## 1.1 问题的背景

丝绸之路是古代东西方文明交往的通道，其中玻璃是早期贸易交流的重要实物资料。对于古代玻璃制品成分的准确鉴别，有利于我们进一步了解古人制作玻璃的技艺，并能作为辅助信息确定玻璃制品的所属年代。但是由于年代久远，玻璃风化等因素导致玻璃成分有所变化，从对大量古玻璃制品的成分分析中找到鉴别玻璃类型的规律对于文物成分和其历史的研究具有重大意义。

## 1.2 问题的重述

基于上述背景与附件中的相关数据，分析并建立数学模型解决以下问题：

(1) 分析玻璃表面风化与玻璃类型、纹饰和颜色的关系；结合玻璃的类型，找到文物表面有无风化化学成分含量的统计规律；预测风化点风化前的化学成分含量。

(2) 分析高钾与铅钡两种类型的玻璃的分类规律；并且分别选择合适的化学成分进行亚类划分，说明具体的划分依据及划分结果，同时，对亚类的划分结果进行合理性与敏感性的分析。

(3) 对所给的未知类别玻璃文物的化学成分进行分析，鉴别其所属类型，并对分类结果的敏感性进行分析。

(4) 分析不同类别的玻璃文物样品化学成分之间的关联关系，并比较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

问题一要求对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。但从数据上来看，仅能对一种变量的值与是否风化的频数进行分析，分析的结论过于狭隘。因此建立关联分析模型，从单个变量到多个变量组合来分析其对玻璃风化的影响，并且选择合适的支持度和置信度加快 Apriori 算法的筛选速度。另外，我们建立两独立样本的 $t$ 检验模型分别确定两种类型的玻璃风化前后，含量发生改变的化学成分。并且在已知风化点的化学成分数据的情况下，依据这些样本化学成分含量平均值的变化差值来预测该玻璃风化前的化学成分。

### 2.2 问题二

问题二可以通过数据可视化对不同类别玻璃成分进行比较，可以较为直观看到玻璃类别划分主要根据各成分含量的多少及相互关系进行，通过 $t$ 检验的方法可以得到高钾玻璃和铅钡玻璃有显著含量差异的成分，并结合各成分含量的热力图作为划分的主要依据。使用聚类分析的方法可以将文物玻璃样本划分成不同的亚类，敏感性可以通过增加人为加入噪音并改变噪音强弱，观察分类的变化情况，并根据轮廓系数等指标综合分析评价。

### 2.3 问题三

问题三依据聚类结果具有的一定稳定性，将待分类的玻璃样本与原数据混合重新进行聚类分析，最后的划分结果即可视作是未知类别玻璃的分类结果；通过不断改变该玻璃样本某一成分含量值，重新进行类别鉴定，寻找使分类结果发生改变的临界值，通过分析得到相对应成分含量变化对分类结果的敏感性。

### 2.4 问题四

问题四分析成分之间的关联关系，先对原始数据进行标准化处理，再分别求出两种玻璃各成分之间的相关系数矩阵，最后根据相关系数大小定性分析成分之间相关关系的强弱。

## 三、模型假设

为了便于模型的建立与求解，现作如下假设：

1. 假设文物样品中未检测到的化学成分比例均视为 0，即视为样本不含有该化学成分。
2. 假设文物样品的化学成分数据服从正态分布，使样品数据符合 $t$ 检验的前提。
3. 假设样本含量检测准确，使样品数据更具有可信度。
4. 假设样本风化前后成分含量只受风化影响，且风化过程花费的时间的长短不影响化学成分的变化，即化学成分的变化只取决于样本是否风化。

# 四、符号说明

| 符号 | 含义 | 单位 |
|---|---|---|
| $\sigma_i$ | 第 $i$ 个样本总体的标准差 | / |
| $S_i^2$ | 第 $i$ 组样本的方差 | / |
| $\nu$ | 自由度 | / |
| $L_{ij}$ | 编号为 $i$ 的未分类玻璃文物成分 $j$ 含量 | % |
| $a(i)$ | 样本 $i$ 到同簇其他样本的平均距离，簇内不相似度 | / |
| $b(i)$ | 样本 $i$ 到其他某簇 $C_j$ 的所有样本的平均距离，簇间不相似度 | / |
| $\overline{P_{\cdot j}}$ | 不改变聚类分析结果的 $j$ 成分平均最大变化百分比 | / |
| $S(i)$ | 第 $i$ 个样本的轮廓系数 | / |

注：其他符号在文中说明。

# 五、模型的建立与求解

## 5.1 数据预处理

由于题目提供的数据存在缺失值，且有可能存在无效数据的情况，我们需要对数据进行清洗和整理，对数据做了以下处理：

(1) 由于成分比例累加和介于之间的数据视为有效数据，表单二中编号 15 和 17（累加值分别为 79.47% 和 71.89%）的数据不满足此条件，将其视为异常值，剔除掉。

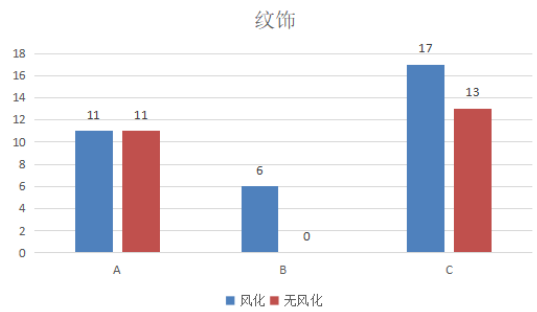
(2) 对于表单中单元格数据出现的的空缺，用 0 填补，以方便后面的数据使用。

(3) 在问题求解过程中多次需要将玻璃类型和是否分化情况分开考虑，所以依照玻璃类型和是否风化的情况将数据分为四组，分别为高钾风化组、高钾无风化组、铅钡风化组和铅钡无风化组。

## 5.2 问题一

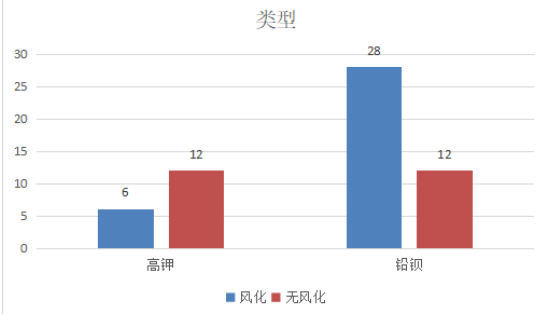
### 数据可视化

首先对单因素影响进行分析，分别做出相应纹饰、玻璃类型及颜色对于风化和无风化结果出现次数的柱状统计图，结果如下:




(a)

图 1　纹饰




(a)

图 2　类型

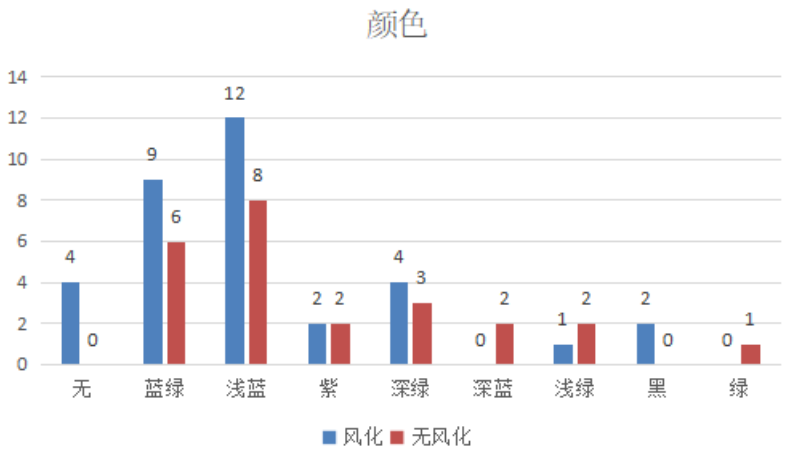



图 3　颜色

从柱状图中大致可以看出单个因素中，纹饰 $B$、铅钡与浅蓝对于玻璃有无风化的影响较大，接下来做进一步分析。

### 关联分析模型的建立

关联分析是从大量数据中发现项集之间有趣的关联和相关联系，从而描述了一个事物中某些属性同时出现的规律和模式。关联关系主要形式之一为代表因果/相关关系的关联规则：关联规则（association rules）暗示两种及两种以上事件之间可能存在很强的关系，它更关注的是事物之间的互相依赖和条件先验关系。它暗示了组内某些属性间不

仅共现，而且还存在明显的相关和因果关系，关联关系一种更强的共现关系 [**?**]。所以从这点上来讲，关联规则是准确率（accuracy）这个指标的一种度量关系。

关联规则形式定义为 $\{x_1, x_2, ...x_n\} \Rightarrow y$，表示 $x_1, x_2, ...x_n$ 事件出现在事务集合时，$y$ 也可能出现在事务集合。据此，此题的关联规则形式定义为 {纹饰，类型，颜色}⇒ 有无风化。

主要衡量指标：

a 支持度——定量评估频繁项集（$k$-项集）的频繁共现度（即覆盖度）的统计量

关联规则的支持度定义如下:

$$support(AB) = P(A \cap B) \tag{1}$$

表示两种事件同时出现的概率。

b 置信度——定量评估一个频繁项集的置信度（准确度）的统计量置信度定义

$$confidence(A, B \Rightarrow C) = P(C|AB) = \frac{support(C \cap AB)}{support(AB)} \tag{2}$$

为了寻找强关联关系，设置一个最小支持度和最小置信度的阈值，以此发现玻璃有无风化和其他因素的关系。

### 关联分析模型求解

寻找元素项间不同的组合是个耗时的任务，为了节约耗时，运用 Apriori 原理减少在数据库上进行检查的集合的数目。其中，Apriori 原理为：如果某个项集是频繁的，那么它的所有子集也是频繁的。我们使用基于 Apriori 原理的 Apriori 算法来求解关联分析模型，Apriori 算法的流程图如图所示:

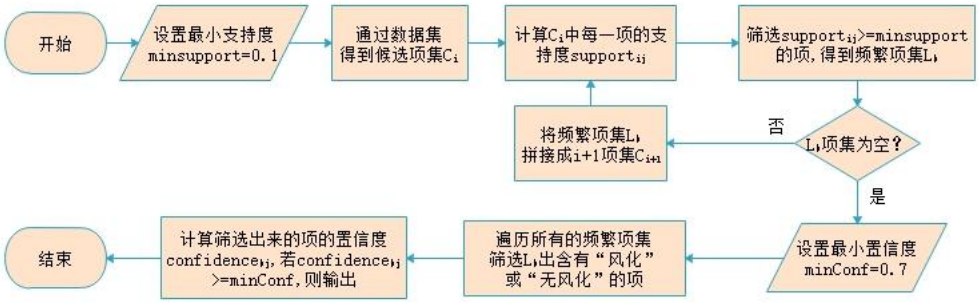



图 **4** **Apriori** 算法流程图

利用 pycharm 软件 Apriori 算法进行求解，设定支持度和置信度阈值分别为 0.1 和 0.7，得到结果。其中满足强关联性的有:

表 **1**　不同条件组合与出现风化结果的关联关系

| 变量集合 | 风化结果 | 置信度 | [支持度，出现次数] |
|---|---|---|---|
| {B} | 风化 | 1.0 | [0.1034, 6] |
| {铅钡} | 风化 | 0.7 | [0.4828, 28] |
| {A，高钾} | 无风化 | 1.0 | [0.1034, 6] |
| {C，高钾} | 无风化 | 1.0 | [0.1034, 6] |
| {B，高钾} | 风化 | 1.0 | [0.1034, 6] |
| {B，蓝绿} | 风化 | 1.0 | [0.1034, 6] |
| {C，铅钡} | 风化 | 0.708 | [0.2931, 17] |
| {铅钡，浅蓝} | 风化 | 0.75 | [0.2068, 12] |
| {B，高钾，蓝绿} | 风化 | 1.0 | [0.1034, 6] |
| {C，铅钡，浅蓝} | 风化 | 1.0 | [0.1034, 6] |

结果解释：

a 单因素对有无风化的影响中，对结果有显著关联影响的为玻璃属性为“铅钡”和纹饰类型为“B”, 在满足阈值的情况下会出现“风化”结果。

b 双因素组合影响中“C”+“高钾”和“A”+“高钾”组合在置信度为 1 的确定结果为“无风化”;“B”+“蓝绿”和“B”+“高钾”组合则是在置信度为 1 的情况下确定结果为“风化”。

c 纹饰，类型，颜色三个因素共同影响下，“B”+“高钾”+“蓝绿”和“C”+“铅钡”+“浅蓝”两个组合都能在置信度为 1 的情况下确定玻璃“风化”。

**两独立样本 $t$ 检验方法**

设两个检验样本 $X(x_1, x_2, ...x_n)$ 和 $Y(y_1, y_2, ...y_m)$，平均值分别为

$$\overline{x} = \frac{x_1 + x_2 + ... + x_n}{n}，\ \overline{y} = \frac{y_1 + y_2 + ... + y_m}{m} \tag{3}$$

对应的样本方差分别为

$$S_1^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2，\ S_2^2 = \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{y})^2 \tag{4}$$

且有 $X \sim (\bar{x}, \sigma_1^2)$，$Y \sim (\bar{y}, \sigma_2^2)$ 其中 $\sigma 1$ 和 $\sigma 2$ 为对应样本总体的标准差，由于 $t$ 检验对数据的非正态性有一定的耐受能力，因此对数据不严格（近似）服从正态分布的情况结果依然稳健。

假设检验：$H_0$：对取定玻璃类型，风化前后该成分无明显差异，即 $\overline{x} = \overline{y}$

$H_1$：对取定玻璃类型，风化前后该成分有差异，即 $\overline{x} \neq \overline{y}$

显著性水平取 $\alpha = 0.05$

接下来需要考虑两个样本总体方差是否相等:

I 两总体方差相等，即 $\sigma_1^2 = \sigma_2^2$，可将两样本方差合并，求二者的共同方差

$$S_c = \frac{(n-1)S_1^2 + (m-1)S_2^2}{n+m-2} \tag{5}$$

两样本 $t$ 检验的检验统计量按照单样本 $t$ 检验统计量计算公式进行计算。

$\mu = \overline{x} - \overline{y} = \mu_0 = 0$ 下，构造检验统计量

$$t = \frac{\bar{X} - \bar{Y} - (\bar{x} - \bar{y})}{s_{\bar{X}-\bar{Y}}} = \frac{\bar{X} - \bar{Y}}{s_{\bar{X}-\bar{Y}}} \quad ，\nu = n + m - 2 \tag{6}$$

即

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{S_c^2(\frac{1}{n} + \frac{1}{m})}} \tag{7}$$

II 两总体方差不相等：

使用 Satterthwaite 近似法，检验统计量

$$t' = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{S_1^2}{n} + \frac{S_2^2}{m}}} \tag{8}$$

的分布比较复杂，需要对自由度进行校正，使 $t'$ 分布近似于 $t$ 分布。自由度校正为

$$\mu = \frac{(\frac{S_1^2}{n} + \frac{S_2^2}{m})^2}{\frac{(\frac{S_1^2}{n})^2}{n-1} - \frac{(\frac{S_2^2}{m})^2}{m-1}} \tag{9}$$

分子是两组样本均数标准误平方之和的平方。据此近似的得到相应的 $P$ 值。

问题求解

将表单 1 和表单 2 中的数据进行整合，按照玻璃材质分为高钾玻璃和铅钡玻璃两种，某一玻璃材质有无风化的化学成分含量满足相互独立。在剔除异常值后可近似认为某一成分含量服从正态分布，依据其有无风化情况下该成分总体方差是否相同，分别使用上述 I、II 两种方法构造检验量的方法。使用 python 的 levene 和 ttest_ind 函数分别对风化前后两种化学成分是否满足总体方差相等和成分是否有明显差异进行检验，检验 $P$ 值大于 $\alpha = 0.05$ 则接受原假设 $H_0$，即认为该成分在风化前后含量没有差异；$P$ 值小于 0.05 则拒绝原假设，即认为该成分在风化前后含量有明显差异。

## 最终结果及检验

$t$ 检验结果如下：

表 **2**　高钾玻璃各成分风化前后含量 **t** 检验结果

| 变量 | 风化结果 | 样本均值 | $t$ | $t$ 检验 $p$ 值 | 风化前后成分是否有显著差异 |
|---|---|---|---|---|---|
| $SiO_2$ | 风化 | 93.96 | 7.0945 | 2.54e-06 | √ |
| | 无风化 | 67.98 | | | |
| $Na_2O$ | 风化 | 0 | -1.3027 | 0.2111 | |
| | 无风化 | 0.695 | | | |
| $K_2O$ | 风化 | 0.54 | -7.6667 | 7.27e-06 | √ |
| | 无风化 | 9.33 | | | |
| $CaO$ | 风化 | 0.87 | -4.8788 | 0.0004 | √ |
| | 无风化 | 5.3325 | | | |
| $MgO$ | 风化 | 0.20 | -3.0109 | 0.0083 | √ |
| | 无风化 | 1.08 | | | |
| $Al_2O_3$ | 风化 | 1.93 | -4.3933 | 0.0005 | √ |
| | 无风化 | 6.62 | | | |
| $Fe_2O_3$ | 风化 | 0.265 | -3.4580 | 0.0053 | √ |
| | 无风化 | 1.93 | | | |
| $CuO$ | 风化 | 1.56 | -1.2101 | 0.2437 | |
| | 无风化 | 2.4525 | | | |
| $PbO$ | 风化 | 0 | -1.6859 | 0.1112 | |
| | 无风化 | 0.41 | | | |
| $BaO$ | 风化 | 0 | -1.4696 | 0.161 | |
| | 无风化 | 0.60 | | | |
| $P_2O_5$ | 风化 | 0.28 | -1.8790 | 0.0785 | |
| | 无风化 | 1.40 | | | |
| $SrO$ | 风化 | 0 | -2.9822 | 0.0124 | √ |
| | 无风化 | 0.04 | | | |
| $SnO_2$ | 风化 | 0 | -0.6963 | 0.4962 | |
| | 无风化 | 0.20 | | | |
| $SO_2$ | 风化 | 0 | -1.3218 | 0.2047 | |
| | 无风化 | 0.10 | | | |

表 **3**　铅钡玻璃各成分风化前后含量 **t** 检验结果

| 变量 | 风化结果 | 样本均值 | t | t 检验 p 值 | 风化前后成分是否有显著差异 |
|---|---|---|---|---|---|
| $SiO_2$ | 风化 | 24.91 | -8.6669 | 2.62e-11 | √ |
| | 无风化 | 54.66 | | | |
| $Na_2O$ | 风化 | 0.22 | -2.8213 | 0.0091 | √ |
| | 无风化 | 1.68 | | | |
| $K_2O$ | 风化 | 0.13 | -1.2593 | 0.2141 | |
| | 无风化 | 0.22 | | | |
| $CaO$ | 风化 | 2.70 | 2.8781 | 0.0060 | √ |
| | 无风化 | 1.32 | | | |
| $MgO$ | 风化 | 0.65 | -0.0124 | 0.9901 | |
| | 无风化 | 0.64 | | | |
| $Al_2O_3$ | 风化 | 2.97 | -1.7444 | 0.0876 | |
| | 无风化 | 4.46 | | | |
| $Fe_2O_3$ | 风化 | 0.58 | -0.6160 | 0.5408 | |
| | 无风化 | 0.74 | | | |
| $CuO$ | 风化 | 2.28 | 1.0479 | 0.3000 | |
| | 无风化 | 1.43 | | | |
| $PbO$ | 风化 | 43.31 | 6.6896 | 2.41e-08 | √ |
| | 无风化 | 22.08 | | | |
| $BaO$ | 风化 | 11.81 | 1.2799 | 0.2068 | |
| | 无风化 | 9.00 | | | |
| $P_2O_5$ | 风化 | 5.28 | 3.8353 | 0.0004 | √ |
| | 无风化 | 1.05 | | | |
| $SrO$ | 风化 | 0.42 | 2.1864 | 0.0337 | √ |
| | 无风化 | 0.27 | | | |
| $SnO_2$ | 风化 | 0.07 | 0.4335 | 0.6665 | |
| | 无风化 | 0.05 | | | |
| $SO_2$ | 风化 | 1.37 | 1.4292 | 0.1595 | |
| | 无风化 | 0.16 | | | |

从结果表中可以看出，在显著性水平为 0.05 的情况下：

a 高钾玻璃组：二氧化硅（$SiO_2$）、氧化钾（$K_2O$）、氧化钙（$CaO$）、氧化镁（$MgO$）、氧化铝（$Al_2O_3$）、氧化铁（$Fe_2O_3$）、氧化锶（$SrO$）七种成分在高钾玻璃风化前后

含量有显著变化。且风化后二氧化硅含量为显著增加，剩余显著变化的成分含量有不同程度减少。

b 铅钡玻璃组：二氧化硅（$SiO_2$）、氧化钠（$NaO$）、氧化钙（$CaO$）、氧化铅（$PbO$）、五氧化二磷（$P_2O_5$）、氧化锶（$SrO$）六种成分在铅钡玻璃风化前后含量有显著变化。二氧化硅含量显著减少，氧化钡和氧化铅含量显著增加。

### 差值预测法

延续上面的分类方式，例如在对高钾玻璃分类过程中，分为无风化的检测点和风化检测点，对单个成分单独进行分析。由于使用 $t$ 检验已经得到风化前后含量显著变化成分，对于基本不变化的成分，可以认为风化前后含量相同。在风化点记录的某一含量在风化前后有显著变化的成分数据集 $x_1, x_2, ..., x_n$，平均数记为 $\overline{X}$，无风化点记录的该成分数据集 $y_1, y_2, ...y_n$，平均数记为 $\overline{Y}$，定义

$$\Delta = \overline{X} - \overline{Y} \tag{10}$$

为该成分风化前与风化后的含量差值。

这样，对于风化点 $i$ 未风化前的预测值就可以近似表示为

$$y_i = x_i - \Delta \tag{11}$$

以高钾玻璃预测结果为例（铅钡玻璃的预测结果放在附录中）：

表 **4**　用差值法预测风化点取样的高钾玻璃风化前的成分

| 文物采样点 | 07 | 09 | 10 | 12 | 22 | 27 |
|---|---|---|---|---|---|---|
| 二氧化硅 ($SiO_2$) | 66.651 | 69.041 | 70.791 | 68.311 | 66.371 | 66.741 |
| 氧化钠 ($Na_2O$) | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 氧化钾 ($K_2O$) | 8.788 | 9.378 | 9.708 | 9.798 | 9.528 | 8.788 |
| 氧化钙 ($CaO$) | 5.533 | 5.083 | 4.673 | 5.183 | 6.123 | 5.403 |
| 氧化镁 ($MgO$) | 0.883 | 0.883 | 0.883 | 0.883 | 1.523 | 1.423 |
| 氧化铝 ($Al_2O_3$) | 6.670 | 6.010 | 5.500 | 6.150 | 8.190 | 7.200 |
| 氧化铁 ($Fe_2O_3$) | 1.837 | 1.987 | 1.927 | 1.957 | 2.017 | 1.867 |
| 氧化铜 ($CuO$) | 3.240 | 1.550 | 0.840 | 1.650 | 0.550 | 1.540 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 氧化铅 ($PbO$) | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 氧化钡 ($BaO$) | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 五氧化二磷 ($P_2O_5$) | 0.610 | 0.350 | 0.000 | 0.150 | 0.210 | 0.360 |
| 氧化锶 ($SrO$) | 0.042 | 0.042 | 0.042 | 0.042 | 0.042 | 0.042 |
| 氧化锡 ($SnO_2$) | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 二氧化硫 ($SO_2$) | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 含量总和（%） | 94.252 | 94.322 | 94.362 | 94.122 | 94.552 | 93.362 |

由上表可以看出，单个样本的化学成分含量总和均在 85% 105% 之间，属于有效数据，预测的结果均合理。

### 模型误差分析

在进行差值法预测玻璃风化前的含量时，为了简便模型，将经过 t 检验得到的风化前后含量无明显变化的成分含量预测值直接取为风化后的值，在小样本空间中会出现一定误差，但是考虑这些成分本身含量也较少，所以造成的误差可忽略不计。

## 5.3 问题二

### 数据可视化

在有效数据中，分别研究高钾玻璃和铅钡玻璃检测样本不同成分含量的占比，做出对应的条形图

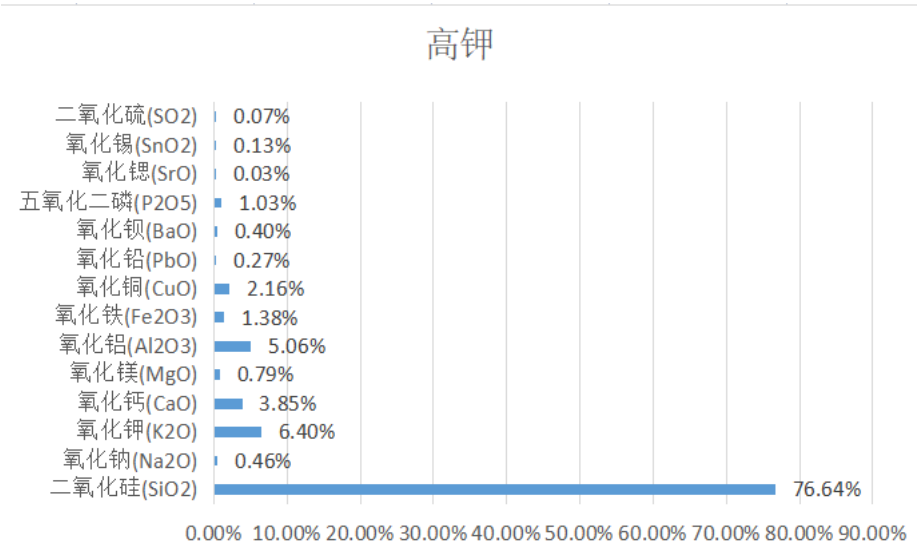



图 **5**　高钾玻璃检测样本不同成分含量的占比

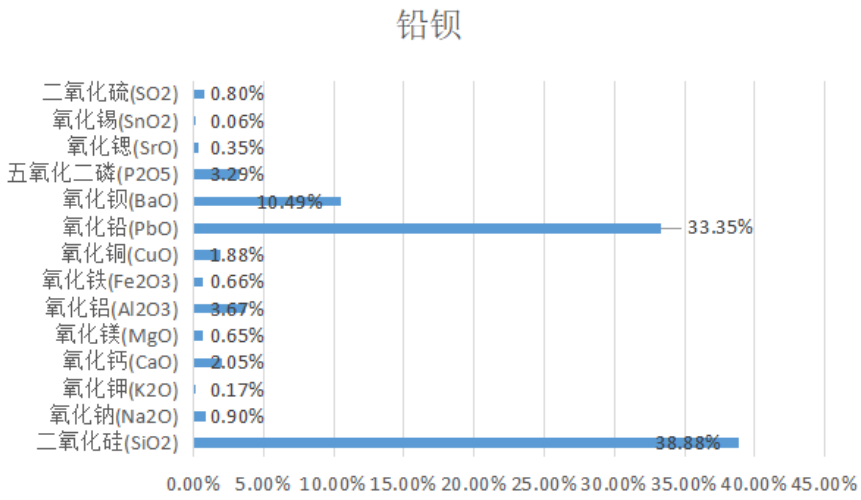



图 **6**　铅钡玻璃检测样本不同成分含量的占比

用 matlab 软件绘制热力图，两种类型玻璃各成分含量的热力图如下所示：

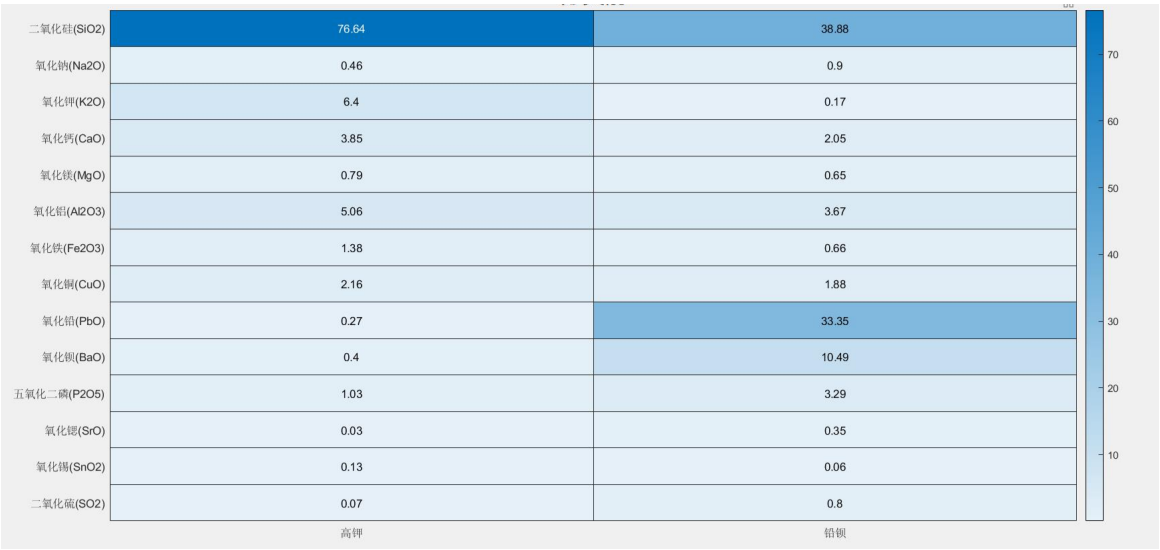



图 **7**　两种类型玻璃不同成分热力图

可以清晰看出，高钾玻璃二氧化硅（$SiO_2$）和氧化钾（$K_2O$）含量占比要远高于铅钡玻璃，而氧化铅（$PbO$）和氧化钡（$BaO$）含量则要远少于铅钡玻璃，这几种主要成分的含量占比是对两种玻璃进行分类的主要指标。

利用 SPSS 软件对两种类型玻璃的不同成分含量做 $t$ 检验，检验结果如下表：

表 **5**　两种玻璃类型“成分含量是否相等”**t** 检验分析结果

| | 玻璃类型 (平均值 ± 标准差) | | $t$ | $p$ |
|---|---|---|---|---|
| | 0.0(n=18) | 1.0(n=49) | | |
| 二氧化硅 ($SiO_2$) | 76.64±14.47 | 38.88±18.65 | 7.764 | 0.000** |
| 氧化钠 ($Na_2O$) | 0.46±1.09 | 0.90±1.81 | -0.968 | 0.337 |
| 氧化钾 ($K_2O$) | 6.40±5.31 | 0.17±0.28 | 4.976 | 0.000** |
| 氧化钙 ($CaO$) | 3.84±3.31 | 2.05±1.63 | 2.205 | 0.039* |
| 氧化镁 ($MgO$) | 0.79±0.71 | 0.65±0.63 | 0.776 | 0.441 |
| 氧化铝 ($Al_2O_3$) | 5.06±3.08 | 3.67±3.01 | 1.665 | 0.101 |
| 氧化铁 ($Fe_2O_3$) | 1.38±1.57 | 0.66±0.95 | 1.832 | 0.081 |
| 氧化铜 ($CuO$) | 2.16±1.49 | 1.88±2.47 | 0.444 | 0.659 |
| 氧化铅 ($PbO$) | 0.27±0.51 | 33.35±14.95 | -15.464 | 0.000** |
| 氧化钡 ($BaO$) | 0.40±0.84 | 10.49±8.33 | -8.364 | 0.000** |
| 五氧化二磷 ($P_2O_5$) | 1.03±1.28 | 3.29±3.91 | -3.567 | 0.001** |
| 氧化锶 ($SrO$) | 0.03±0.04 | 0.35±0.26 | -8.202 | 0.000** |
| 氧化锡 ($SnO_2$) | 0.13±0.56 | 0.06±0.21 | 0.782 | 0.437 |
| 二氧化硫 ($SO_2$) | 0.07±0.16 | 0.80±3.14 | -1.627 | 0.11 |

其中，$*p \leq 0.05 **p \leq 0.01$

从上表中可以很直观地看出，t 检验得到的结论与在热力图中得到的结论较为一致，说明上述分类规律具有较强的可靠性。

### 聚类分析

聚类分析指将物理或抽象对象的集合分组为由类似的对象组成的多个类的分析过程。划分式聚类算法需要预先指定聚类数目或聚类中心，通过反复迭代运算，逐步降低目标函数的误差值，当目标函数值收敛时，得到最终聚类结果 [**?**]。

基于划分（距离）的聚类方法，对给定的有 $n$ 个样本的数据集，根据样本空间差异，将其划分为 $k(k < n)$ 个聚类（分组），这样可以将混乱的数据集进行分类，方便后续研究。

K-means(K-均值) 算法:

K-means 是一种无监督的学习，通过人为选定种子点的个数，其自动选离种子点最近的均值。

K-means 算法步骤：

I 首先选定 $k$ 个种子点位置。

II 通过两点距离计算，将所有样本点归结到离其最近的种子点簇中，这样进行初次划分。

III 然后再通过移动种子点位置到其点群中心的位置，重复 II 过程重新划分点群。

IV 直到最后种子点位置不再改变，则说明找到了一个聚类划分。

对于距离的计算，选择欧拉距离（Euclidean Distance），在 $n$ 维欧氏空间 $R^n$ 中向量 $X(x_1, x_2, ...x_n)^T$ 和 $Y(y_1, y_2, ..., y_n)^T$ 的欧拉距离计算公式为

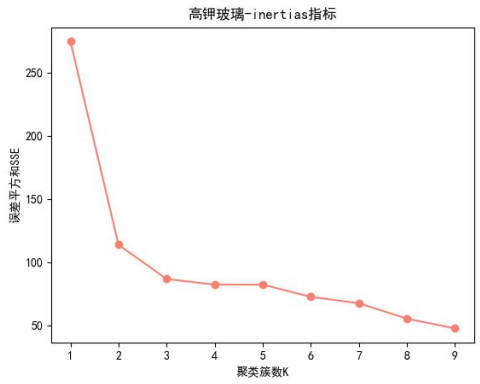
$$distance(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{12}$$

分类型后的玻璃数据，一共有 14 种成分，每种元素当作欧氏空间中的一维，即在 $R^{14}$ 空间中，通过计算欧氏距离的方式聚合样本离散点，将玻璃文物样本分为 $k$ 个簇（类别）。

**聚类划分求解**

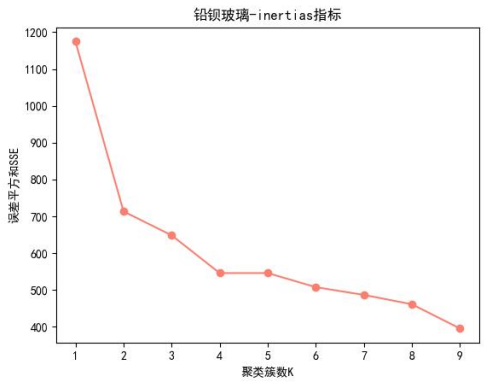
I. 簇数 $k$ 的选择的肘部法则

为了确定在聚类划分结果更加准确，首先需要确定划分的最佳簇数, 使用肘部法则对 K-means 算法的 K 值确定起到指导作用，在选择类别数量上，肘部法则会把不同值的成本函数值画出来。随着值的增大，每个簇包含的样本数会减少，于是样本离其重心会更近平均畸变程度会减小。随着值继续增大，平均畸变程度的改善效果会不断减低。值增大过程中，畸变程度的改善效果下降幅度最大的位置，即折线图中出现的折点最明显的位置对应的横坐标就是该样本划分的最佳簇数，使用 pycharm 软件进行求解得到两种玻璃的肘部法则图：




(a)

图 **8**　高钾玻璃肘部法则图




(a)

图 **9**　铅钡玻璃肘部法则图

从上图中可以看出，高钾玻璃和铅钡玻璃聚类的最佳簇数分别为 2 和 4。

II. 划分亚类：

根据 k-means 算法步骤，使用 pycharm 软件进行求解，得到高钾玻璃主要的两个（完整结果放在支撑材料中）亚类划分（根据划分结果中各亚类主要成分命名）为高硅组（$SiO_2$）和钾铁组（$K_2O$,$Fe_2O_3$), 铅钡玻璃划分分为高铅组（$PbO$）、硫钡组（$SO_2$，$BaO$，$CuO$）、硅铅组（$SiO_2$，$PbO$）和硅钠组（$SiO_2$，$Na_2O$）。

**合理性和敏感性分析 [?]**

I. 轮廓系数法

(1) $a(i)$: 样本 $i$ 到同簇其他样本的平均距离，称为簇内不相似度。

$$a(i) = \frac{1}{n-1}\sum_{j\neq i}^{n} distance(i,j) \tag{13}$$

$a(i)$ 越小，说明样本 $i$ 越应该被聚类到该簇。

(2) 样本 $i$ 到其他某簇 $C_j$ 的所有样本的平均距离 $b_{ij}$，称为样本 $i$ 与最近簇 $C_j$ 的不相似度。另外，$b(i)$ 称为样本 $i$ 的簇间不相似度。

$$b(i) = min\{b_{i1}, b_{i2}, ..., b_{ik}\} \tag{14}$$

$$C_j = arg(\min_{C_k} \frac{1}{n}\sum_{p\in C_k} |p - X_i|^2) \tag{15}$$

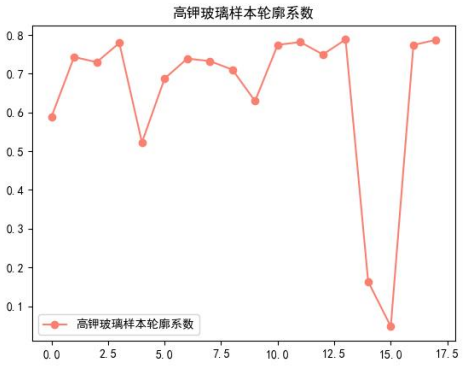
$b(i)$ 越大，说明样本 $i$ 越不属于其他簇。

(3) 根据样本 $i$ 的簇内不相似度 $a(i)$ 和簇间不相似度 $b(i)$，定义样本 $i$ 的轮廓系数：

$$S(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & a(i) < b(i) \\ 0 & a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & a(i) > b(i) \end{cases} \tag{16}$$

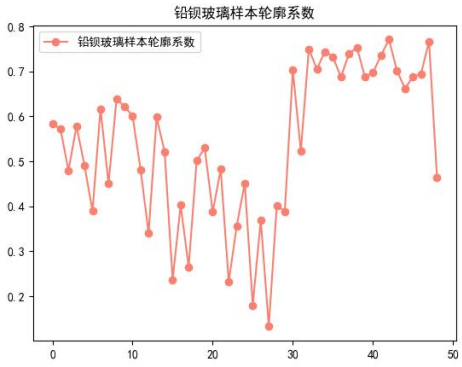
轮廓系数范围在 [-1,1] 之间，该值越大，表示聚类结果越合理。$S_i$ 接近 1，则说明样本 $i$ 聚类合理；$S_i$ 接近-1，则说明样本 $i$ 更应该分类到另外的簇；若 $S_i$ 近似为 0，则说明样本 $i$ 在两个簇的边界上。

通过 pycharm 软件求出两种玻璃聚类后的轮廓系数。




(a)

图 **10** 高钾玻璃每个样本点聚类后的的轮廓系数折线图




(a)

图 **11** 铅钡玻璃每个样本点聚类后的的轮廓系数折线图

高钾玻璃组的总体轮廓系数为 0.6515，铅钡玻璃组的 0.5406，说明聚类效果较好。

II. 通过加入噪音对聚类敏感性进行分析

噪音：在机器学习中在独立随机抽样的时候会出现一些搞错的信息, 聚类分析还面临着来自聚类特征的干扰, 即数据中有可能包含一些与聚类分析结果不相关或影响聚类分析结果的特征, 这些特征被称为噪音特征 (Noisy Feature), 简称噪音 [**?**, **?**, **?**]，这些噪音数据对数据分析会造成不同程度的影响。

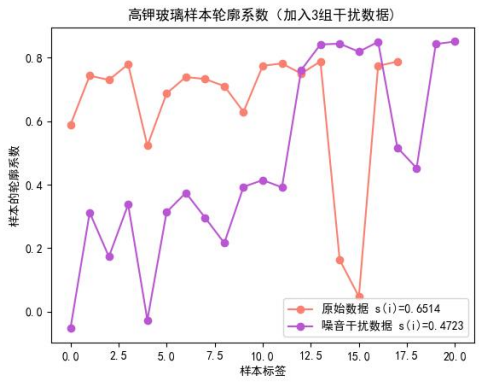
为了分析聚类划分的敏感性，我们通过以下步骤加入人工噪音并分析敏感度。

step1: 随机生成多组与要分析玻璃成分种类（14 种）相同维度的欧氏空间数据，即噪音。

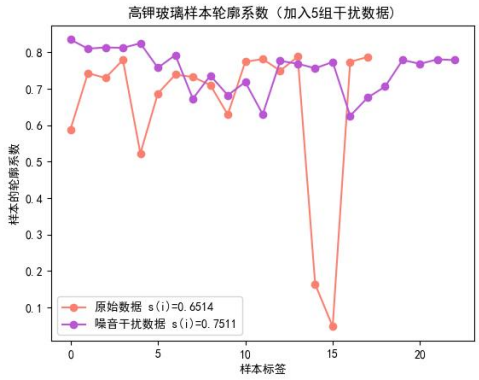
step2：将若干组噪音数据与原始数据一起再次进行聚类分析，并得到相应的聚类结果。

step3: 对加入噪音得到的聚类划分结果进行评价并与未加入噪音得到的数据进行比较，分析噪音对于玻璃类别划分的影响。

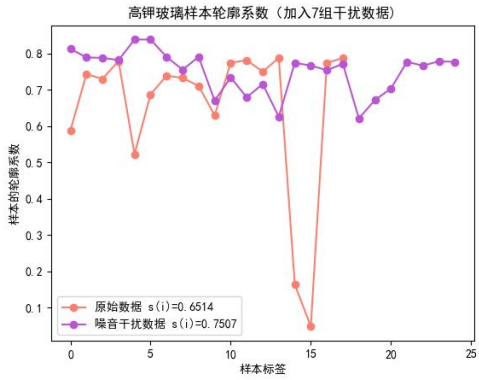
step4：逐步增强噪音（加入更多组噪音数据），观察聚类结果的耐受性，综合分析出分类的敏感性。




(a) 3 组

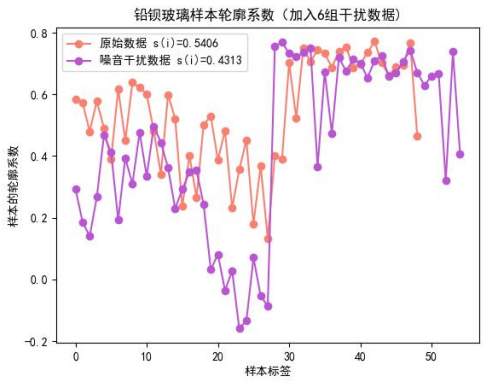



(b) 5 组

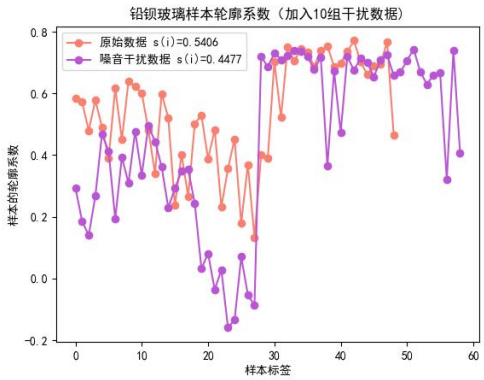



(c) 7 组

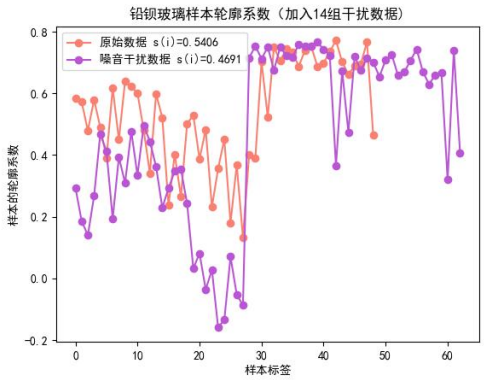
图 **12**　高钾：加入不同组干扰数据之后样本轮廓系数对比图




(a) 6 组




(b) 10 组




(c) 14 组

图 **13**　铅钡：加入不同组干扰数据之后样本轮廓系数对比图

结果说明及误差分析：

通过上述加入噪声前后每个样本点轮廓系数折线图的比较，通过加入不同数量组的干扰数据，铅钡玻璃聚类划分的结果一直保持在较高并且稳定水平，说明铅钡玻璃聚类划分对于噪音的干扰相对不敏感，而高钾玻璃组由于本身有效数据较少聚类结果相对不稳定，因此在加入少量干扰数据后，轮廓系数有较大水平的下降，而在加入较多干扰数

据后，干扰数据反而自成一体（加入的随机噪声有约束条件，因此数据较为接近），成为一个簇，导致在强度较大的噪音干扰下，高钾玻璃组聚类效果更好。

### 5.4 问题三

#### 利用聚类划分的稳定性鉴别新玻璃文物样本的种类

鉴别模型的建立与求解

原理：从问题二的分析中，已经知道该聚类划分方法在加入不超过 5 组干扰数据（噪音）时，对于分类结果基本没有影响，这有赖于其较强的抗干扰性。因此，不妨假设在加入一组未知类别玻璃的检测成分数据后，原划分组没有发生改变。更新数据集，通过查看新聚类划分后，该类别玻璃被划分到哪一个组别，即可认为是该玻璃的类型。

鉴别结果如下表：

表 **6** 文物样本的鉴定结果

| 文物编号 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
|---|---|---|---|---|---|---|---|---|
| 玻璃类型 | 高钾玻璃 | 铅钡玻璃 | 铅钡玻璃 | 铅钡玻璃 | 铅钡玻璃 | 高钾玻璃 | 高钾玻璃 | 铅钡玻璃 |
| 亚类划分 | 高硅组 | 硅铅组 | 硅铅组 | 硅铅组 | 硅钠组 | 高硅组 | 高硅组 | 硅钠组 |

首先，通过待鉴别的数据与已知的数据中对比可以看出，待鉴别的文物样品的化学成分含量的分布特征与已知的数据分布特征有同种分布特征，因此通过聚类算法进行玻璃类型的鉴别是具备高准确度的。另外，在鉴别确定文物的玻璃类型之后，将高钾类型与铅钡类型的数据分开，并将待鉴别的文物加入到相对应的玻璃类型数据中，再次使用聚类算法来确定文物所属的亚类。由于亚类划分的分界线并不明显，因此进行亚类划分时，会存在一定的误差，但准确度在可接受的范围内。

#### 敏感性分析

实现思想:将原未分类玻璃 $\{A1, A2, ...A8\}$ 中某一成分 $j$ 含量记为 $L_{ij}, i = 1, 2, ..., 8, j = 1, 2, ..., 14$，，对 $L_{ij}$ 值进行人为改变，观察重新的聚类结果，找到使该玻璃分类结果发生改变的最小变化含量值上下界为 $LA_{ij}, LB_{ij}(LA_{ij} < L < LB_{ij})$，聚类结果 $i$ 样本玻璃最大能忍受的 $j$ 成分改变百分比

$$P_{ij} = \begin{cases} \frac{min(|LA_{ij}-L_{ij}|,|L_{ij}-LB_{ij}|)}{L_{ij}} & L_{ij} \neq 0 \\ \frac{min(|LA_{ij}-L_{ij}|,|L_{ij}-LB_{ij}|)}{\sum_{k=1}^{8} L_{kj}} & Lij = 0 \end{cases} \tag{17}$$

在所有未分类玻璃样本分类不改变情况下，对 j 成分发生改变的最大平均百分比

$$\overline{P_{\cdot j}} = \frac{\sum_{i=1}^{8} P_{ij}}{8} \tag{18}$$

用于衡量该成分的敏感性。

对二氧化硅、氧化铅、氧化钡等几种主要成分的敏感度进行分析：

例如通过改变二氧化硅含量得到的对未知类别玻璃聚类的结果, 结果如下表所示（红色标注代表与原始数据的类别不同）。其中，表中数据 $SiO_2$-5 表示在原数据基础上，将 $SiO_2$ 含量减少 5，分类组别为 6 种，分别为：

| | | |
|---|---|---|
| C1：高钾钾铁组 | B1：铅钡高铅组 | B3：铅钡硅铅组 |
| C2：高钾高硅组 | B2：铅钡硫钡组 | B4：铅钡硅钠组 |

表 **7**　$SiO_2$ 含量改变对目标样本聚类结果的影响

| 聚类结果 文物编号 / $SiO_2$ 含量变化 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
|---|---|---|---|---|---|---|---|---|
| $SiO_2$-10 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-9 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-8 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-7 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-6 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-5 | C1 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-4 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-3 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-2 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$-1 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |

| 样本原数据 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
|---|---|---|---|---|---|---|---|---|
| $SiO_2$+1 | C2 | B3 | B3 | B3 | B4 | C2 | C2 | B4 |
| $SiO_2$+2 | C2 | B3 | B3 | B3 | B4 | B4 | C2 | B4 |
| $SiO_2$+3 | C2 | B3 | B3 | B3 | B4 | B4 | C2 | B4 |
| $SiO_2$+4 | C2 | B3 | B3 | B3 | C1 | B4 | C2 | B4 |
| $SiO_2$+5 | C2 | B3 | B3 | B3 | C1 | B4 | B4 | B4 |
| $SiO_2$+6 | C2 | B3 | B3 | B3 | C1 | B4 | B4 | B4 |
| $SiO_2$+7 | C2 | B3 | B3 | B4 | C1 | B4 | B4 | B4 |
| $SiO_2$+8 | C2 | B3 | B3 | B4 | C1 | B4 | B4 | B4 |
| $SiO_2$+9 | C2 | B3 | B3 | B4 | C1 | B4 | B4 | B4 |
| $SiO_2$+10 | C2 | B3 | B3 | B4 | C1 | B4 | B4 | B4 |

可以看到在 $SiO_2$ 含量在分别减少 5 和增加 2 就开始会改变 A1，A6 样本划分结果。对其余成分采取类似做法，得到各主要成分对应的 $\overline{P_{\cdot j}}$ 值：

表 **8**　几种成分的 $\overline{P_{\cdot j}}$ 值

| 主要成分 | $SiO_2$ | $BaO$ | $PbO$ | $Fe_2O_3$ |
|---|---|---|---|---|
| $\overline{P_{\cdot j}}$ | 20.88% | 339.47% | 53.313% | 1718.04% |

对上述结果进行分析，由于二氧化硅、氧化铅等在玻璃制品中含量较多，在改变其含量时，对分类结果影响较大，而氧化铁之类的成分含量较少，按照倍数增加其含量，依旧不会对产生大的影响，认为其对结果敏感性较低。

### 5.5　问题四

#### 相关系数矩阵

通过相关系数矩阵计算不同类别玻璃化学成分之间的相关性。

(1) 对原始数据进行标准化处理。将各指标 $a_{ij}$ 转化成标准化指标 $\widetilde{a_{ij}}$，有

$$\widetilde{a_{ij}} = \frac{a_{ij} - \mu_j}{s_j}, i = 1, 2, ..., m, j = 1, 2, ..., n, \tag{19}$$

式中：

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} a_{ij}; s_j = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(a_{ij}-\mu_j)^2}, j = 1,2,...,n, \tag{20}$$

即 $\mu_j, s_j$ 为第 $j$ 个指标的样本均值和样本标准差，其中 $m$ 为样本数，$n$ 为指标数（$n$=14）

对应地，称

$$\widetilde{x_j} = \frac{x_j - \mu_j}{s_j}, j = 1,2,...,n \tag{21}$$

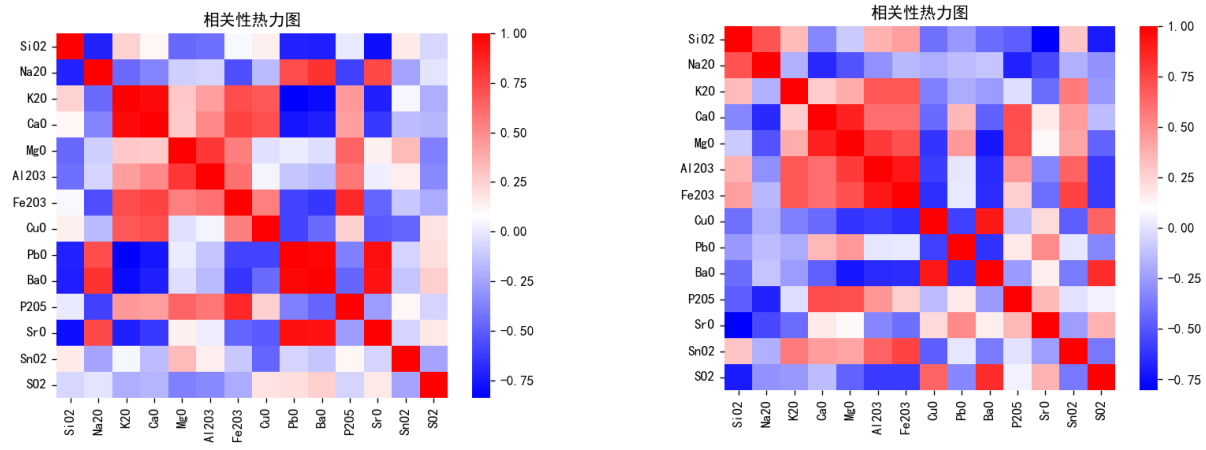
为标准化指标变量

(2) 计算相关系数矩阵 $R$ 相关系数矩阵 $R = (r_{ij})_{n\times n}$，有

$$r_{ij} = \frac{\sum_{k=1}^{m} \widetilde{a_{ki}} \cdot \widetilde{a_{ki}}}{m-1}, i,j = 1,2,...,n \tag{22}$$

式中：$r_{ii} = 1; r_{ij} = r_{ji}, r_{ij}$ 第 $i$ 个指标与第 $j$ 个指标的相关系数，$r_{ij} \in [-1,1]$，$r_{ij}$ 越接近 1，表示两个指标的相关性越高；若取值为正值，表示两个指标之间是正相关关系；反之，为负相关。

在满足显著水平 0.05 的情况下，分别得到两种类别玻璃成分之间的相关系数，组成相关系数矩阵，用热力图绘出结果如下：




(a) 1　　(b) 2

图 **14**　高钾玻璃和铅钡玻璃成分含量相关系数矩阵热力图

结果分析：

从相关系数矩阵热力图中可以可以看出，高钾玻璃的氧化铅（$PbO$）与氧化钡（$BaO$）都有很强的相关性，相关系数分别为 0.844，且其主要成分二氧化硅（$SiO_2$）与其它组成分间为负相关关系或基本无关，铅钡玻璃氧化钾和氧化钡之间为明显的负相关，氧化铜和氧化钡之间为正相关，相关系数为 0.699。可以直观看出总体而言铅钡玻璃组成成分之间的相关性要强于高钾玻璃，差异主要在于氧化铅和氧化钡之间的相关关系。

# 六、模型的评价与推广

## 6.1 模型优点

本文建立的古玻璃制品成分分析鉴别的综合评价模型，采用多种方法和评价标准对测得的古玻璃制品成分进行多层次的分析，对所要求的问题都能给出合理数据支撑和解答，在各种评价指标下有不错的表现，并可以根据已有数据挖掘出有用信息帮助我们建立相应的古玻璃分类和分析方式。

1. 所建立的模型认可度高，简单易懂，且模型能够求解得出较为理想、合理的结论。
2. 模型的计算采用专业软件求解，例如 excel、matlab、spss、pycharm 等软件，使模型得出的结论更具可信度。
3. 建立的模型能够与实际紧密联系，使模型具备实用性、通用性以及推广性。

## 6.2 模型缺点

1. 在根据风化点检测数据预测风化前成分含量的预测时，没有考虑各成分间的风化过程化学变化的相互关系，会导致预测结果有一定误差。
2. K-means 聚类方法在高维空间易受异常值的影响，导致分类的结果存在一定误差。

## 6.3 模型改进

在分析不同类型玻璃及是否风化时，可以建立不同成分间准确数值关系，结合相应的风化过程化学反应，定量讨论成分间含量的变化关系，对玻璃类型鉴别给出直接的计算公式，以未知类别的古玻璃各成分含量为自变量，将因变量作为鉴别结果，使鉴别过程更简便易行。

## 6.4 模型改进

通过对古玻璃制品数据的研究，本文给出了检验、预测、评价、灵敏度分析等数据处理的方法，在如今数据极为重要的时代，对于各领域的数据分析都有着一定的参考借鉴作用，可以将这些分析方法类似运用于市场经营、数据调研等诸多领域。

# 七、参考文献


[1] 邹志文, 朱金伟. 数据挖掘算法研究与综述 [J]. 计算机工程与设计,2005(09):2304-2307.DOI:10.16208/j.issn1000-7024.2005.09.014.

[2] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究 [J]. 软件学报,2008(01):48-61.

[3] 蔡毅, 邢岩, 胡丹. 敏感性分析综述 [J]. 北京师范大学学报 (自然科学版),2008(01):9-16.

[4] JIANGD,TANG C,ZHANG A.Clusteranalysisforgeneex pression-data:asurvey[J].IEEETransactionsonKnowledge& DataEngineering,2004.

[5] SMYTHC,COOMANSD,EVERINGHAM Y.Clusteringnoisy datainareduceddimension-spacevia multivariateregression trees[J]. PatternRecognition,2006.

[6] 杨虎, 付宇, 范丹. 噪音特征对聚类内部有效性的影响 [J]. 计算机科学,2018,45(07):22-30+52.

# 八、附录

附录 1：关联分析 python 实现源代码

附录 2：独立样本 t 检验 python 源代码

附录 3：铅钡玻璃风化前的化学成分含量预测结果

附录 4：k-means 类 python 源代码

附录 5：K-means 聚类算法 python 源代码

附录 6：聚类分析判断玻璃类型 python 源代码

## 8.1 关联分析 python 实现源码

```
#关联分析
import csv

#1. 从表单 1 的第二行第二列开始读取有效数据
def loadDataSet(filename):
    dataSet = []
    row = 0
    with open(filename,'r',encoding='utf-8-sig') as file:
        csvReader=csv.reader(file)
        for line in csvReader:
            if row > 0:
                dataSet.append(line[1:])
            row += 1
    return dataSet

#2. 构建候选 1 项集 C1
def createC1(dataSet):
    C1 = []
    for transaction in dataSet:
        for item in transaction:
            if not [item] in C1:
                C1.append([item])

    C1.sort()
    return list(map(frozenset, C1))

#3. 将候选集 Ck 转换为频繁项集 Lk
#D: 原始数据集
#Cn: 候选集项 Ck
#minSupport:支持度的最小值
def scanD(D, Ck, minSupport):
    #候选集计数
    ssCnt = {}
```

```
    for tid in D:
        for can in Ck:
            if can.issubset(tid):
                if can not in ssCnt.keys(): ssCnt[can] = 1
                else: ssCnt[can] += 1

    numItems = float(len(D))
    Lk= []  # 候选集项Cn生成的频繁项集Lk
    supportData = {} #候选集项Cn的支持度字典
    #计算候选项集的支持度, supportData key:候选项,  value:支持度
    for key in ssCnt:
        support = ssCnt[key] / numItems
        if support > minSupport:
            Lk.append(key)
        supportData[key] = [support,ssCnt[key]]
    return Lk, supportData

#4.连接操作，将频繁Lk-1项集通过拼接转换为候选k项集
def aprioriGen(Lk_1, k):
    Ck = []
    lenLk = len(Lk_1)
    for i in range(lenLk):
        L1 = Lk_1[i]
        for j in range(i + 1, lenLk):
            L2 = Lk_1[j]
            if len(L1 & L2) == k - 2:
                L1_2 = L1 | L2
                if L1_2 not in Ck:
                    Ck.append(L1 | L2)
    return Ck

#5.
def apriori(dataSet, minSupport = 0):
    C1 = createC1(dataSet)
    L1, supportData = scanD(dataSet, C1, minSupport)
    L = [L1]
    k = 2
    while (len(L[k-2]) > 0):
        Lk_1 = L[k-2]
        Ck = aprioriGen(Lk_1, k)
        #print("ck:",Ck)
        Lk, supK = scanD(dataSet, Ck, minSupport)
        supportData.update(supK)
        #print("lk:", Lk)
        L.append(Lk)
        k += 1
    return L, supportData

```

```
#6. 计算规则的可信度，并过滤出满足最小可信度要求的规则
def calcConf(freqSet, H, supportData, brl, minConf=0.7):
    ''' 对候选规则集进行评估 '''
    prunedH = []
    for conseq in H:
        conf = supportData[freqSet][0] / supportData[freqSet - conseq][0]
        if conf >= minConf:
            if conseq == frozenset({'风化'}) or conseq == frozenset({'无风化'}):
                print (freqSet - conseq, '-->', conseq, 'conf:', conf , 'support:',
                    supportData[freqSet])
            brl.append((freqSet - conseq, conseq, conf))
            prunedH.append(conseq)
    return prunedH

#7. 根据当前候选规则集H生成下一层候选规则集
def rulesFromConseq(freqSet, H, supportData, brl, minConf=0.7):
    m = len(H[0])
    while (len(freqSet) > m): # 判断长度 > m，这时即可求H的可信度
        H = calcConf(freqSet, H, supportData, brl, minConf) #
            返回值prunedH保存规则列表的右部，这部分频繁项将进入下一轮搜索
        if (len(H) > 1): # 判断求完可信度后是否还有可信度大于阈值的项用来生成下一层H
            H = aprioriGen(H, m + 1)
            #print ("H = aprioriGen(H, m + 1): ", H)
            m += 1
        else: # 不能继续生成下一层候选关联规则，提前退出循环
            break

#8. 频繁项集列表L
# 包含那些频繁项集支持数据的字典supportData
# 最小可信度阈值minConf
def generateRules(L, supportData, minConf=0.7):
    bigRuleList = []
    # 频繁项集是按照层次搜索得到的，
        每一层都是把具有相同元素个数的频繁项集组织成列表，再将各个列表组成一个大列表，所以需要遍历Len(L)次，
        即逐层搜索
    for i in range(1, len(L)):
        for freqSet in L[i]:
            H1 = [frozenset([item]) for item in freqSet] #
                对每个频繁项集构建只包含单个元素集合的列表H1
            rulesFromConseq(freqSet, H1, supportData, bigRuleList, minConf) #
                根据当前候选规则集H生成下一层候选规则集
    return bigRuleList

if __name__=='__main__':
    dataset = loadDataSet('D:\.nvscode\py\guosai\表单1.csv')
    L, supportData = apriori(dataset, minSupport=0.1)
    for key,values in supportData.items():
        if '风化' in key or '无风化' in key:
```

```
            print(key, values)
    # print(supportData)
    # 基于频繁项集生成满足置信度阈值的关联规则
    rules = generateRules(L, supportData, minConf=0.7)
```

## 8.2 独立样本 t 检验 python 源代码

```
#独立样本t检验
import csv
from scipy.stats import ttest_ind, levene
import pandas as pd

def loadData(filename):
    dataSet = []
    with open(filename,'r',encoding='utf-8-sig') as file:
        csvReader=csv.reader(file)
        for line in csvReader:
            print(line[0])
            x = line[1:7]      #高钾玻璃组，6组风化，12组无风化
            y = line[7:]
            x = list(map(float, x))
            y = list(map(float, y))
            print(levene(x,y))     #方差齐次性检查
            if levene(x,y).pvalue>0.05:
                print(ttest_ind(x, y)) # 独立样本T检验,默认方差齐性
            else:
                print(ttest_ind(x,y,equal_var=False)) #如果方差不齐性，则equal_var=False
            print("\n")

if __name__ == '__main__':
    loadData('D:\.nvscode\py\guosai\高钾玻璃t检验.csv')
```

## 8.3 铅钡玻璃风化前的化学成分含量预测结果

表 **9** 铅钡玻璃风化前的预测结果

| 文物采样点 | 二氧化硅 | 氧化钠 | 氧化钾 | 氧化钙 | 氧化镁 | 氧化铝 | 氧化铁 |
|---|---|---|---|---|---|---|---|
| 02 | 66.03 | 1.47 | 1.05 | 0.97 | 1.18 | 5.73 | 1.86 |
| 08 | 49.89 | 1.47 | 0.00 | 0.11 | 0.00 | 1.34 | 0.00 |
| 08 严重风化点 | 34.36 | 1.47 | 0.00 | 1.82 | 0.00 | 1.11 | 0.00 |
| 11 | 63.34 | 1.47 | 0.21 | 2.14 | 0.71 | 2.69 | 0.00 |
| 19 | 59.39 | 1.47 | 0.00 | 1.56 | 0.59 | 3.57 | 1.33 |
| 26 | 49.54 | 1.47 | 0.00 | 0.07 | 0.00 | 0.70 | 0.00 |
| 26 严重风化点 | 33.47 | 1.47 | 0.40 | 1.64 | 0.00 | 1.18 | 0.00 |
| 34 | 65.53 | 1.47 | 0.25 | -0.59 | 0.00 | 1.62 | 0.47 |
| 36 | 69.32 | 3.69 | 0.14 | -1.00 | 0.00 | 1.60 | 0.32 |
| 38 | 62.68 | 2.85 | 0.00 | -0.69 | 0.00 | 2.57 | 0.29 |
| 39 | 56.00 | 1.47 | 0.00 | -0.26 | 0.00 | 0.50 | 0.00 |
| 40 | 46.46 | 1.47 | 0.00 | 0.50 | 0.00 | 0.45 | 0.19 |
| 41 | 48.21 | 1.47 | 0.44 | 3.59 | 2.73 | 3.33 | 1.79 |
| 43 部位 1 | 42.16 | 1.47 | 0.00 | 3.87 | 0.89 | 2.25 | 0.76 |
| 43 部位 2 | 51.45 | 1.47 | 0.00 | 5.03 | 0.95 | 3.41 | 1.39 |
| 48 | 83.08 | 2.27 | 0.32 | 1.45 | 1.54 | 13.65 | 1.03 |
| 49 | 58.54 | 1.47 | 0.00 | 3.21 | 1.47 | 5.38 | 2.74 |
| 50 | 47.73 | 1.47 | 0.00 | 1.82 | 0.47 | 1.87 | 0.33 |
| 51 部位 1 | 54.36 | 1.47 | 0.00 | 2.21 | 1.19 | 5.25 | 1.19 |
| 51 部位 2 | 51.10 | 1.47 | 0.00 | 3.76 | 1.45 | 2.51 | 0.42 |
| 52 | 55.49 | 2.69 | 0.00 | 0.90 | 0.55 | 1.16 | 0.23 |
| 54 | 52.03 | 1.47 | 0.32 | 1.82 | 1.28 | 4.15 | 0.00 |
| 54 严重风化点 | 46.86 | 1.47 | 0.00 | -1.37 | 1.11 | 3.65 | 0.00 |
| 56 | 58.90 | 1.47 | 0.00 | -0.16 | 0.00 | 1.85 | 0.00 |
| 57 | 55.17 | 1.47 | 0.00 | -0.06 | 0.00 | 2.18 | 0.00 |
| 58 | 60.14 | 1.47 | 0.34 | 2.12 | 0.79 | 3.52 | 0.86 |

| 氧化铜 | 氧化铅 | 氧化钡 | 五氧化二磷 | 氧化锶 | 氧化锡 | 二氧化硫 | 含量总和（%） |
|---|---|---|---|---|---|---|---|
| 0.26 | 26.20 | 0.00 | -0.66 | 0.04 | 0.00 | 0.00 | 104.12 |
| 10.41 | 7.45 | 31.23 | -0.64 | 0.22 | 0.00 | 2.58 | 104.05 |
| 3.14 | 11.22 | 30.62 | 3.33 | 0.38 | 0.00 | 15.03 | 102.47 |
| 4.93 | 4.16 | 14.61 | 5.15 | 0.22 | 0.00 | 0.00 | 99.62 |
| 3.51 | 21.59 | 5.35 | 4.60 | 0.04 | 0.00 | 0.00 | 102.99 |
| 10.57 | 8.30 | 32.25 | -1.10 | 0.30 | 0.00 | 1.96 | 104.05 |
| 3.60 | 8.69 | 35.45 | 1.81 | 0.47 | 0.00 | 15.95 | 104.12 |
| 1.51 | 25.32 | 10.00 | -3.89 | 0.07 | 0.00 | 0.00 | 101.75 |
| 0.68 | 20.38 | 10.83 | -4.16 | 0.07 | 0.00 | 0.00 | 101.86 |
| 0.73 | 28.08 | 9.79 | -3.75 | 0.26 | 0.00 | 0.00 | 102.80 |
| 0.88 | 39.80 | 7.22 | -3.07 | 0.46 | 0.00 | 0.00 | 102.99 |
| 0.00 | 48.98 | 6.69 | -2.46 | 0.53 | 0.00 | 0.00 | 102.80 |
| 0.19 | 22.89 | 9.76 | 3.23 | 0.32 | 0.00 | 0.00 | 97.94 |
| 5.35 | 38.62 | 7.29 | -4.23 | 0.49 | 0.00 | 0.00 | 98.91 |
| 1.51 | 23.52 | 3.26 | 8.60 | 0.32 | 0.00 | 0.00 | 100.90 |
| 0.00 | -5.52 | 7.31 | -3.13 | 0.10 | 1.31 | 0.00 | 103.40 |
| 0.70 | 12.95 | 6.10 | 6.87 | 0.31 | 0.00 | 0.00 | 99.73 |
| 1.13 | 22.77 | 14.20 | 2.11 | 0.51 | 0.00 | 0.00 | 94.40 |
| 1.37 | 19.01 | 8.94 | 3.87 | 0.24 | 0.47 | 0.00 | 99.56 |
| 0.75 | 30.11 | 0.00 | 4.52 | -0.15 | 0.00 | 0.00 | 95.93 |
| 0.70 | 26.19 | 8.64 | 1.48 | 0.29 | 0.00 | 0.00 | 98.31 |
| 0.83 | 34.23 | 7.04 | 0.01 | 0.73 | 0.00 | 0.00 | 103.90 |
| 1.34 | 37.23 | 0.00 | 9.90 | 0.97 | 0.00 | 0.00 | 101.15 |
| 0.79 | 20.02 | 15.45 | -1.69 | -0.15 | 0.00 | 0.00 | 96.47 |
| 1.16 | 23.87 | 17.30 | -4.23 | -0.15 | 0.00 | 0.00 | 96.70 |
| 3.13 | 18.12 | 7.66 | 4.76 | 0.09 | 0.00 | 0.00 | 102.99 |

## 8.4 k-means 类 python 源代码

```
import numpy as np

class KMeans:
    def __init__(self, n_clusters=1, max_iter=300, random_state=0):
        self.k = n_clusters
        self.N = max_iter
```

```
        np.random.seed(random_state)

    def assign_to_centers(self, centers, X):
        assignments = []
        for i in range(len(X)):
            distances = [np.linalg.norm(X[i]-centers[j], 2) for j in range(self.k)]
            assignments.append(np.argmin(distances))
        return assignments

    def adjust_centers(self, assignments, X):
        new_centers = []
        for j in range(self.k):
            cluster_j = [X[i] for i in range(len(X)) if assignments[i] == j]
            if cluster_j != []:
                new_centers.append(np.nanmean(cluster_j, axis=0))
            else:
                new_centers.append(0)
        return new_centers

    def fit_transform(self, X):
        idx = np.random.randint(0, len(X), self.k)
        centers = [X[i] for i in idx]
        assignments = self.assign_to_centers(centers, X)
        for t in range(self.N):
            assignments = self.assign_to_centers(centers, X)
            centers = self.adjust_centers(assignments, X)
        return np.array(centers, dtype=object), np.array(assignments, object)
```

## 8.5 K-means 聚类算法 python 源代码

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

plt.rcParams['font.sans-serif'] = ['simHei']
plt.rcParams['axes.unicode_minus'] = False
name = ['高钾', '铅钡', '高钾与铅钡', '表单1']
# SiO2
Y = []
Y1 = pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[3], usecols="C:P") #
    原始数据
Y.append(Y1)
for i in range(1, 11):
    Y2 = Y1.copy()
    Y2["二氧化硅(SiO2)"][:] += i
```

```
    Y.append(Y2)
for i in range(1, 11):
    Y2 = Y1.copy()
    Y2["二氧化硅(SiO2)"][:] -= i
    Y.append(Y2)

# BaO
Z = []
Z1 = pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[3], usecols="C:P") # 原始数据
Z.append(Z1)
for i in range(1, 11):
    Z2 = Z1.copy()
    Z2["氧化钡(BaO)"][:] += i
    Z.append(Z2)
for i in range(1, 11):
    Z2 = Z1.copy()
    Z2["氧化钡(BaO)"][:] -= i
    Z.append(Z2)

# PbO
V = []
V1 = pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[3], usecols="C:P") # 原始数据
V.append(V1)
for i in range(1, 11):
    V2 = V1.copy()
    V2["氧化铅(PbO)"][:] += i
    V.append(V2)
for i in range(1, 11):
    V2 = V1.copy()
    V2["氧化铅(PbO)"][:] -= i
    V.append(V2)

# Fe2O3
B = []
B1 = pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[3], usecols="C:P") # 原始数据
B.append(B1)
for i in range(1, 11):
    B2 = B1.copy()
    B2["氧化铁(Fe2O3)"][:] += i
    B.append(B2)
for i in range(1, 11):
    B2 = B1.copy()
    B2["氧化铁(Fe2O3)"][:] -= i
    B.append(B2)

```

```
61
62  def sure_location(centers, assignments, k):
63      location = -1
64      xx = []
65      for i in range(len(centers)):
66          xx.append([])
67      for i in range(len(assignments)):
68          xx[assignments[i]].append(i + 1)
69      for i in range(len(xx)):
70          if k in xx[i]:
71              location = i
72          print(xx[i])
73      return location
74
75
76  # 判断在两大类中的哪一类
77  def judge_double(X):
78      model = kmeans.KMeans(n_clusters=4, max_iter=15)
79      centers, assignments = model.fit_transform(X)
80      location = -1
81      xx = []
82      for i in range(len(centers)):
83          xx.append([])
84      for i in range(len(assignments)):
85          xx[assignments[i]].append(i + 1)
86      for i in range(len(xx)):
87          if 68 in xx[i] and 1 in xx[i]:
88              print("高钾类别")
89              return i
90          elif 68 in xx[i]:
91              location = i
92          print(xx[i])
93      print("铅钡类别")
94      return location
95
96
97  # 判断为高钾中的哪一亚类
98  def judge_GK(X):
99      model = kmeans.KMeans(n_clusters=2, max_iter=15)
100     centers, assignments = model.fit_transform(X)
101     return sure_location(centers, assignments, 19)
102
103
104 # 判断为铅钡中的哪一亚类
105 def judge_QB(X):
106     model = kmeans.KMeans(n_clusters=4, max_iter=15)
107     centers, assignments = model.fit_transform(X)
108     return sure_location(centers, assignments, 50)
```

```
109
110
111 # 循环判断
112 #
    判断的顺序为（原始数据,+1,+2,+3,+4,+5,+6,+7,+8,+9,+10,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10）
113 def circle_judge(W):
114     for j in range(len(W)):
115         for i in range(len(W[j])):
116             X = np.concatenate((x3, np.array([W[j].iloc[i]], dtype=object)))
117             temp = judge_double(X)
118             print(temp)
119             X = np.concatenate((x2, np.array([W[j].iloc[i]], dtype=object)))
120             print("类别: ", judge_QB(X) + 1)
121             X = np.concatenate((x1, np.array([W[j].iloc[i]], dtype=object)))
122             print("类别: ", judge_GK(X) + 1)
123             print()
124         plt.legend() # 暂停作用
125         plt.show()
126
127
128 x1 = np.array(pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[0],
    usecols="E:R"), dtype=object)
129 x2 = np.array(pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[1],
    usecols="E:R"), dtype=object)
130 x3 = np.array(pd.read_excel('K-means聚类预测类别.xlsx', sheet_name=name[2],
    usecols="E:R"), dtype=object)
131
132 # SiO2
133 circle_judge(Y)
134 print("------------分割线------------")
135 # BaO
136 circle_judge(Z)
137 print("------------分割线------------")
138 # PbO
139 circle_judge(V)
140 print("------------分割线------------")
141 # Fe2O3
142 circle_judge(B)
143 print("------------分割线------------")
```

## 8.6 聚类分析判断玻璃类型 python 源代码

```
1 import numpy as np
2 import pandas as pd
3 import random
4 import math
```

```
import matplotlib.pyplot as plt
import kmeans

plt.rcParams['font.sans-serif'] = ['simHei']
plt.rcParams['axes.unicode_minus'] = False
name = ['高钾', '铅钡']
Name = ['原始数据', '噪音干扰数据']
color = ['salmon', 'mediumorchid']


def calculate_distance(point, center):
    dist = 0
    for l in range(len(point)):
        dist += (point[l]-center[l])**2
    return math.sqrt(dist)


def select_k_value(x, w):
    SSE = []
    for i in range(1, 10):
        model = kmeans.KMeans(n_clusters=i, max_iter=15)
        centers, assignments = model.fit_transform(x)
        inertias = 0
        for j in range(len(assignments)):
            inertias += calculate_distance(x[j], centers[assignments[j]])
        SSE.append(inertias)
    fig = plt.subplot()
    fig.plot(list(range(1, 10)), SSE, 'o-', color='salmon', label='inertias-k的关系')
    fig.set_title(name[w]+'玻璃-inertias指标')
    fig.set_xlabel('聚类簇数K')
    fig.set_ylabel('误差平方和SSE')
    fig.figure.savefig(name[w]+'玻璃-inertias指标.jpg')
    plt.show()


# 轮廓系数计算
def sc(x, assignments, k, ww):
    xx = []
    for j in range(k):
        xx.append([])
    for j in range(len(assignments)):
        xx[assignments[j]].append(x[j])
    ai = []
    bi = []
    si = []
    for j in range(len(xx)):
        for l in range(len(xx[j])):
            # 计算ai
```

```
            a = 0
            for w in range(len(xx[j])):
                if l == w:
                    continue
                else:
                    a += calculate_distance(xx[j][l], xx[j][w])
            ai.append(a / (len(xx[j]) - 1))
            # 计算blw
            bl = 0
            blw = []
            for w in range(len(xx)):
                if j == w:
                    continue
                else:
                    for r in range(len(xx[w])):
                        bl += calculate_distance(xx[j][l], xx[w][r])
                    blw.append(bl/len(xx[w]))
            bi.append(min(blw))
    for j in range(len(ai)):
        si.append((bi[j]-ai[j])/max(ai[j], bi[j]))
    print(sum(si)/len(si))
    return si


def Print(x, k, w):
    model = kmeans.KMeans(n_clusters=k, max_iter=15)
    centers, assignments = model.fit_transform(x)
    xx = []
    for j in range(k):
        xx.append([])
    for j in range(len(assignments)):
        xx[assignments[j]].append(j+1)
    for j in range(k):
        print(xx[j])
    si = sc(x, assignments, k, w)
    print()
    plt.plot(list(range(len(si))), si, 'o-', color=color[0],
        label=name[w]+'玻璃样本轮廓系数')
    plt.xlabel('样本标签')
    plt.ylabel('样本的轮廓系数')
    plt.title(name[w]+'玻璃样本轮廓系数')
    plt.legend()
    plt.savefig(name[w]+'玻璃样本轮廓系数.jpg')
    plt.show()


def contrast(x, j):
    model = kmeans.KMeans(n_clusters=K[j], max_iter=15)
```

```
    centers, assignments = model.fit_transform(x)
    return sc(x, assignments, K[j], 0)


def Rand(x, j, num):
    Y = pd.read_excel('随机数.xlsx', sheet_name='Sheet1', usecols="A:N")
    for i in range(num):
        r = random.randint(1, 99)
        x = np.concatenate((x, np.array(Y.iloc[r:r+1], dtype=object)), axis=0)
    model = kmeans.KMeans(n_clusters=K[j], max_iter=15)
    centers, assignments = model.fit_transform(x)
    return sc(x, assignments, K[j], 1)


k1, k2 = 2, 4
K = [k1, k2]
for i in range(2):
    print(name[i] + ":(与标签号对应)")
    X = np.array(pd.read_excel('K-means聚类的分类结果.xlsx', sheet_name=name[i],
        usecols="E:R"), dtype=object)
    select_k_value(X, i)
    Print(X, K[i], i)

n = 5 # 加入n组与2n组干扰数据
for i in range(2):
    X = np.array(pd.read_excel('K-means聚类的分类结果.xlsx', sheet_name=name[i],
        usecols="E:R"), dtype=object)
    model = kmeans.KMeans(n_clusters=K[0], max_iter=15)
    centers, assignments = model.fit_transform(X)
    si = contrast(X, i)
    plt.plot(list(range(len(si))), si, 'o-', color=color[0], label=Name[0]+'
        s(i)='+str(int(sum(si)/len(si)*10000)/10000))
    si = Rand(X, i, n*(i+1))
    plt.plot(list(range(len(si))), si, 'o-', color=color[1], label=Name[1]+'
        s(i)='+str(int(sum(si)/len(si)*10000)/10000))
    plt.title(name[i]+'玻璃样本轮廓系数（加入'+str(n*(i+1))+'组干扰数据)')
    plt.xlabel('样本标签')
    plt.ylabel('样本的轮廓系数')
    plt.legend()
    plt.savefig(name[i] + '玻璃样本轮廓系数对比('+str(n*(i+1))+').jpg')
    plt.show()
    print()
```