古代玻璃制品的成分分析与鉴别模型

摘要

本文主要建立卡方检验模型、方差分析模型、正态性检验模型、基于基尼指数的 CART 分类决策树模型、K-means 聚类分析模型、皮尔曼相关系数分析模型来解决古代玻璃制品的成分分析与鉴别模型。

首先对附件数据进行预处理，将原本化学成分含量为空的值设为 0；将化学成分总和异常的数据剔除掉；在研究颜色对表面风化的影响时，将颜色的缺失值删除掉；并建立起取样部位与文物属性的联系。

对于问题一，使用卡方检验分析表面风化与玻璃类型、纹饰、颜色的关系，得出表面风化与玻璃类型、纹饰存在较强相关性关系，而与颜色的相关性不强。通过作图分析，发现铅钡玻璃更容易风化，纹饰 A 和纹饰 C 与风化的相关性相反。之后利用方差分析模型分析有无风化条件下玻璃化学成分含量的差异，在每种类别下，都得到了 6 种受风化影响变化显著的变量，用箱型图呈现其前后取值的对比，可以发现这些变量在风化前后呈现出明显的上升下降关系。为了得到这些风化敏感型变量的统计规律，使用正态性检验模型检验其与正态分布的接近程度。最后，利用风化前后风化敏感型变量正态分布的线性映射关系来预测其风化前的含量，同时使用同种工艺下化学成分的均值作为不敏感型变量的预测值。预测结果与同类玻璃文物的化学成分含量接近，说明了预测的准确性。

对于问题二，通过表单二中化学成分信息，建立一棵 CART 分类决策树，来确定未知类别玻璃文物的所属类型，并限制特征的选取来得到使用不同特征的 3 棵决策树，经检验发现用于分类的特征在两种类别玻璃中差异明显。在选取样本所有特征得到的决策树中，氧化铅含量小于 5.46 分类为高钾玻璃，大于等于 5.46 分类为铅钡玻璃。对化学成分进行进一步分析，使用 K-means 聚类模型在每一种类别样本中划分亚类，最终在每一个类别中都划分出了 4 种亚类。在每一个类别中，通过对各个亚类化学成分进行比较，归纳出亚类划分的具体依据。从聚类中心点选取、聚类数目 K 和杰卡德相关系数三方面说明分类结果的合理性，并使用杰卡德相关系数衡量敏感性。

对于问题三，使用问题二得到的玻璃分类模型对未知样本进行分类。用 3 棵决策树分别对这些未知样本进行分类，分类结果一致，表明这 3 棵决策树之间可以相互检验正确性。使用原有的风化和未风化样本分别训练决策树，并用这两棵决策树对未知样本进行分类，分类结果一致，验证了模型的敏感性。

对于问题四，针对不同类别的玻璃文物，绘制斯皮尔曼相关系数热图，分析每两种化学成分之间的关联关系。对于相关关系成立概率超过 99%的化学成分对，通过线条连接，绘制关联关系图。最后，通过关联关系图比较不同类别玻璃化学成分关联关系的差异性，发现铅钡玻璃化学成分之间的相关关系相较于高钾玻璃更加紧密。

关键词：卡方检验，方差分析，CART 分类决策树，K-means 聚类，斯皮尔曼相关系数

一、 问题重述

古代玻璃易受埋藏环境的影响而风化。在风化过程中，内部元素和环境元素发生大量交换，导致其成分比例发生变化，从而影响对其类别的正确判断。现有一批我国古代玻璃制品的相关数据，考古工作者依据这些文物样品的化学成分和其他检测手段将其分成了高钾玻璃和铅钡玻璃两种类型。

附件中，表单 1 给出了文物的分类信息，表单 2 给出了相应的主要化学成分所占比例。这些数据的特点是表示化学成分的含量，所以各成分比例的累加和应该为 100%，但因检测手段等原因可能导致其成分比例的累加和非 100%。本题将成分比例累加和介于 85%~105%的数据视为有效数据。

需要解决的问题如下：

（1） 分析玻璃文物的表面风化与其玻璃类型、纹饰、颜色的关系；结合玻璃类型，分析文物样品表面有无风化化学成分含量的统计规律，并对分类结果的合理性和敏感性进行分析。

（2） 依据附加数据分析高价玻璃和铅钡玻璃的分类规律；对于每个类别选择合适的化学成分对其进行亚类划分，给出具体的划分方法及结果，并分析结果的合理性和敏感性。

（3） 对表单 3 中未知类别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） 对于不同类别的玻璃文物样品，分析其化学成分之间的关联关系，并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

在题目给定的数据中，表单 1 是文物的分类信息，包括文物的编号和其纹饰、类型、颜色、表面是否风化；表单 2 描述了在文物上选取若干采样点，其中有些采样点来自相同文物的不同部分，有的采样点来自风化文物的无风化点、风化点、严重风化点。表单 3 是一些未知类别的玻璃文物，既不知道玻璃的类别，也不知道玻璃的纹饰、颜色等工艺特征，仅仅知道与表单 2 相同的化学成分的含量。

2.1 玻璃文物的风化与化学成分含量、工艺参数的关系

2.1.1 玻璃文物的表面风化与工艺参数的关系

在观察表单 1 的数据时，可以发现有几个数据颜色数据是缺失的，而在分析颜色、纹饰、类型与是否风化的关系时，这几个缺失值并不能完整地反映颜色对是否风化的影响，需要把这几个样本去掉。然后用卡方分析确定颜色、纹饰、类型与是否风化是否存在关系。确定二者存在关系后，可以进行进一步分析，分析在哪种类别下，风化的玻璃最多，以此确定某种因素对风化的具体影响。

2.1.2 文物样品表面有无风化化学成分含量的统计规律

此题要分析的是在同种类别的文物中，风化前后化学成分含量的统计规律，应当包括风化前后发生显著性差异的化学成分有哪些，发生变化的方向和幅度是什么，以及发生风化前后样本自身存在的统计规律——即分布规律。

为了分析风化前后发生显著性差异的化学成分有哪些，可以对每一个化学成分含量都做方差分析，把数据划分为风化与无风化两组，检验两组是否来源于相同的总体。若不是来源于相同的总体，则表明是否风化对这种化学成分的含量产生了显著影响，之后分析影响的方向和幅度。

需要注意，在不同类别中，受风化影响产生显著性变化的变量和其变化的方向有可能不同，需要对每一个类别单独分析，并结合文献分析可能的原因。而分析样本自身的统计规律可以用正态性检验，判断其是否符合正态分布，并计算方差和平均值。

2.1.3 根据风化点检测数据预测风化前的化学成分含量

此问题是预测风化前的各化学成分含量。对于总计 14 个化学成分，分为两组分别考虑：第一组是各类别中受风化影响变化显著的变量，称为风化敏感型变量；第二组是各类别中受风化影响不显著的变量，称为风化不敏感型变量。风化不敏感型变量在风化前后的变化不是很显著，不能直接通过风化后化学成分推导出风化前的化学成分。但是由于同种工艺参数下某种化学成分风化前的含量基本相同，因此，可以用同种工艺参数下该种化学成分风化前的平均含量，来估计该种文物风化前的化学成分含量。

2.2 玻璃制品的分类规律与亚类划分

2.2.1 玻璃制品的分类规律

这一问实际上是要求只使用附件中提供的化学成分含量的特征，来判断该玻璃文物所属的类型。在找出玻璃样品类别的分类规律后，就可以用于问题 3 中对未知类别玻璃文物所属类型的分析。

我们可以用决策树来完成玻璃文物的分类任务。决策树是一种逻辑简单、易于理解的机器学习算法，采用树形结构，使用层层推理来实现最终的分类。决策树由根节点、内部节点、叶节点三种元素组成。根节点表示样本的全集，预测时，从根节点出发，经过内部节点的判断条件选择树上的路径，最终达到叶节点的分类结果。这些规则都是通过传入数据后进行训练得到的，训练的目标是对于所有训练样本，经过决策树分类得到的样本类型能与其标签类型一致。我们通过决策树得到一组树状的分类规则之后，就可以将待分类样本传入决策树，实现分类任务。

一般的决策树算法使用的特征都是离散的数据，而我们的题目中使用到的化学成分含量是连续型变量，因此考虑采用 CART 分类决策树。CART 分类树的思想是把连续的特征离散化，具体来说，如果连续特征 A 有 m 个从小到大排列的值，那么 CART 取相邻两个值之间的平均值作为划分点，一共 m-1 个，之后分别计算以这 m-1 个点作为二元分类点时的基尼系数，选择基尼系数最小的点为该连续特征的二元离散分类点，递归生成决策树，这样就做到了连续特征的离散化。

2.2.2 铅钡玻璃、高钾玻璃的亚类划分

由于没有给出具体的亚类标签，仅给出了每个类别的玻璃采样点的化学成分，无法使用常规的算法得到划分。所以这一问题需要采用无监督聚类学习方法，并从所得划分中总结最显著的化学成分作为该亚类的特征。我们采用最常见的 k-means 聚类算法作为主要划分方法对玻璃类型分类后的表单 2 进行聚类分析，通过选择初始聚类中心，进行中心化迭代，将样本集划分成互不相交的若干个聚类，作为本题要求的划分结果。进一步，通过对于每个玻璃类型亚类划分结果的聚类中心进行均值和图表分析，归纳出各个聚类所具有的化学成分特征，并进一步将其总结为亚类划分的依据。

2.3 未知类别玻璃文物的类型鉴别

在 2.2.1 训练得到决策树的分类模型之后，用它去分类表单 3 中未知类别文

物的所属类别，得到一个分类结果。为了验证分类结果的合理性，可以用 2.2.1 中得到的 3 棵决策树分别分类表单 3 中的文物，然后查看分类结果是否存在差异，如差异较小，说明 3 棵决策树之间可以相互检验正确性，决策树没有过拟合；且三棵决策树选取特征之间存在差异，可以在被测样本某些特征缺省时，灵活选择适合的决策树进行分类，表现出了模型的鲁棒性。

2.4 不同类别玻璃文物中化学成分的关联关系

为了评估不同类别中，玻璃文物样品各化学成分之间的关联关系，可以使用相关系数来量化这个关系。常用的相关系数包括皮尔曼相关系数、斯皮尔曼相关系数，但皮尔曼相关系数仅仅适用于正态分布，而我们的样本中某些化学成分的分布不明确，因此我们可以采用斯皮尔曼相关系数来分析不同类别玻璃文物样品化学成分之间的关联关系。两种化学成分之间的斯皮尔曼相关系数绝对值越接近 1，说明两者的相关性越强。

得到斯皮尔曼相关系数后，可以使用 Spearman 热图直观地展示两两化学成分之间的相关系数，之后选取相关系数足够大的化学成分对，绘制关联关系图，从而可以直观地呈现化学成分之间的关联，也便于在高钾和铅钡之间做化学成分关联的比较。

三、 模型假设

(1) 玻璃文物铸造时的化学成分含量与制造参数（类型、纹饰、颜色）有很大的关系。同种类型、纹饰、颜色的玻璃风化前的各化学成分含量相近。

(2) 风化前后含量均服从正态分布的化学成分，在风化过程中的变化是缓慢而速度稳定的。

(3) 未知样本和已知样本的制造工艺和储藏环境都是一致的，即不考虑除题中所给因素外其他因素的影响。

(4) 由于取样是随机的，除了同一样本上两次不同部位取样的情形，可以认为取样点的化学成分可以代表玻璃样本整体的化学成分。

四、 符号说明

符号	含义
χ^2	卡方值
f_{o_i}	卡方分析中数据 i 的观测频次
f_{e_i}	卡方分析中数据 i 的期望频次
df	卡方分析中数据的自由度
c	风化前的某化学成分含量
$N(\mu,\sigma^2)$	均值为μ, 标准差为σ的正态分布
c'	风化后的某化学成分含量
D	决策树的数据集
$Gini(D)$	数据集D的基尼值
$GiniIndex(D\|A)$	在按照属性 A 划分数据集 D 时，对应的基尼指数
A^*	决策树的最优特征
t^*	决策树最优特征的最优划分点
SSE	样本聚类误差平方和，又称“和方差”
$J(C,C')$	杰卡德相关系数，用于衡量划分集合C和C'的相似度

五、 模型的建立与求解

5.1 玻璃文物的风化与化学成分含量、工艺参数的关系

5.1.1 玻璃文物的表面风化与工艺参数的关系

模型选择

在分析颜色、纹饰、类型与是否风化之间的关系时，首先需要知道哪些变量与是否风化有影响，为此可以使用卡方分析。卡方分析可用于检验两个变量之间是否存在关联关系。在分类问题中，选取特征和分类结果两个变量，通过卡方检验计算特征和分类结果之间的相关性，就可以做特征选择。

卡方分析的基本原理是设定某个离散因素对某个离散结果没有影响的基本假设，并通过假设检验确定在此种假设下题目中数据出现的概率，如果概率很小，表示假设不成立，这两个离散因素之间存在关联；否则，没有足够把握确定假设不成立。确定二者存在关系后，可以进行进一步分析，分析在哪种类别下，风化的玻璃最多，以此确定某种因素对风化的具体影响。

分析流程

以分析玻璃类型与是否风化之间的关联为例，卡方分析的流程如下：

（1） 建立假设检验，在本问题中，原假设是玻璃类型和是否风化之间相互独立。

（2） 计算期望频次，即在二者相互独立的假设下，对于每一种玻璃类型，其期望的是否风化样本的数量。由于假设是二者相互独立，因此有无风化样本数量的比例应当都等于总样本中有无风化样本数量的比例。

观测频次：

玻璃类型	风化	未风化	合计	风化比例
高钾	6	12	18	33.3%
铅钡	24	12	36	66.7%
合计	30	24	54	55.6%

假设表面风化与玻璃类型无关时的预测频次：

玻璃类型	风化	未风化	合计	风化比例
高钾	10	8	18	55.6%
铅钡	20	16	36	55.6%
合计	30	24	54	55.6%

表 1 玻璃类型和风化与否的频次

（3） 代入卡方统计公式计算卡方值。公式为：

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_{o_i} - f_{e_i})^2}{f_{e_i}}$$

其中f_{o_i}为观测到的频次，即样本的数量；f_{e_i}是期望得到的频次，即应当出现的样本的数量；n 是观测频次的数量，在本问题中自变量为玻璃类型，有 2 种取值，因变量有无风化也有 2 种取值，因此 $n = 2 * 2 = 4$。

上述公式中，分子代表了实际值与期望值的偏差，而分母则是标准化过程。因此卡方的值越小，表示观测值与期望值(理论值) 越接近，说明两个变量之间越符合卡方分布。而卡方分布的前提是变量之间相互独立，因此卡方值越小，越表示两个变量之间独立。

计算得到$\chi_1{}^2 = \frac{(6-10)^2}{10} + \frac{(12-8)^2}{8} + \frac{(24-20)^2}{20} + \frac{(12-16)^2}{16} = 5.4$

（4） 计算自由度。自由度df_1=(行数-1)(列数-1)，所以四格表的自由度为 1。

（5） 查表，比较卡方值。

自由度为 1，查表得，我们有 97%的把握说明风化与玻璃类型有关。

结果分析

同样可以计算表面风化与玻璃纹饰的卡方值χ_2^2 =6.93，自由度df_2=2，查表得，我们有 95%的把握说明风化与玻璃纹饰有关。

计算表面风化与玻璃颜色的卡方值χ_3^2 =5.79，自由度df_3=7，查表得，我们有不到 50%的把握说明风化与玻璃颜色有关。

同样地，也可以通过画图的方式佐证卡方分析的结果。如下三图是表面风化与玻璃类型、纹饰、颜色的关系统计图。第一张图表明铅钡玻璃相比高钾玻璃更容易发生表面风化。第二张图表示玻璃纹饰为 B 时，全部样本都会发生风化，而在玻璃纹饰为 A 时，无风化的几率略大于风化；在玻璃纹饰为 C 时，风化的几率略高于未风化，这也印证了卡方分析中纹饰与风化相关的结果。第三张图中，卡方分析得到的结果是表面风化和玻璃颜色并没有很大的关系，而图中有的同颜色组中，风化的数量高于未风化，有的风化的数量低于未风化，但差异不显著，有的全是风化，有的全是未风化，但数量较少。这也印证了卡方分析的结果：表面风化与玻璃颜色之间存在一定的关系，但关系不显著。

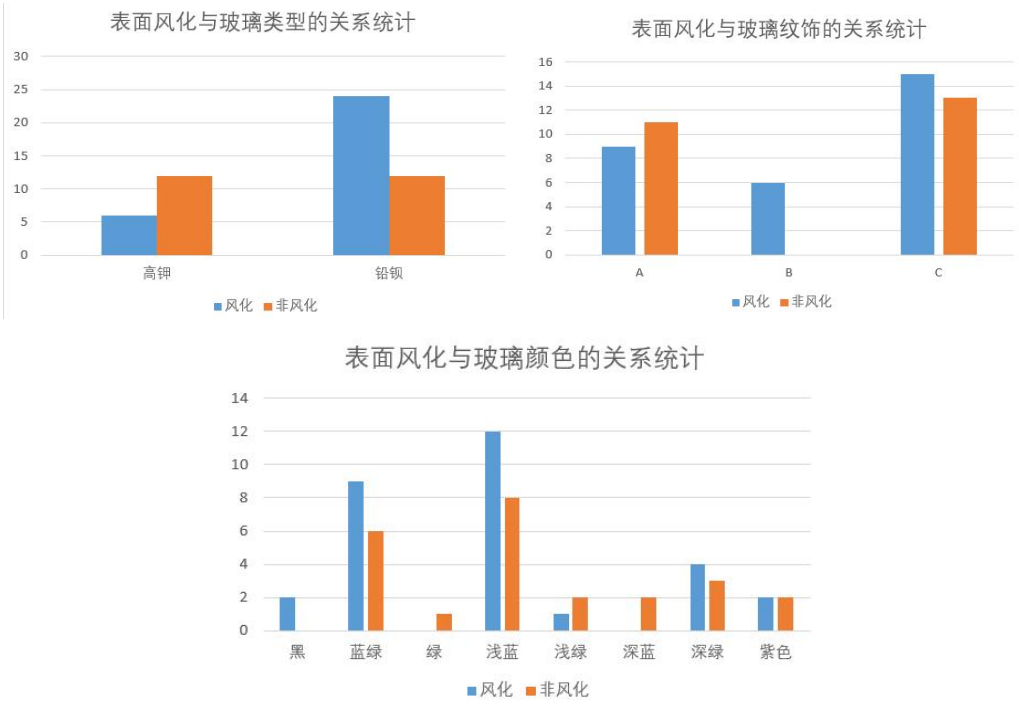


图 1 表面风化与相关因素的关系统计

5.1.2 风化前后化学成分含量的统计规律

模型建立与计算过程

方差分析又称“变异数分析”，是用于两个及两个以上样本均数差别的显著性检验。其原理是通过求出两个多个样本的平均值，然后假定它们来源于相同的总体，并计算两者发生此种平均值差异的概率。若概率小于显著性水平α，那么否定原假设，即两者存在显著差异，明显来自于不同的样本总体。

首先，计算表格中各个文物采样点各种化学成分含量的加和，如果小于 85%或者大于 105%，则标记为异常值，将其从表单 2 数据中剔除。经过检查，总共有 2 条数据被清除。（注：后文使用的所有化学成分数据中均清除了这 2 条数据，后文不再赘述。）

其次分析在同种类型下，有无风化条件下化学成分含量是否存在差异。如果存在显著性差异，则表示此化学成分在风化前后的含量发生了巨大变化，可以用于区分玻璃是否风化。

结果分析

为确定何种化学成分的含量受到风化影响较为显著，取高钾玻璃数据作为样本，我们使用 SPSS 在线分析工具对类别为“有无风化”，指标为各化学成分含量的数据做方差分析，所得到的变量的范围和其方差分析的显著性水平如下图所示。

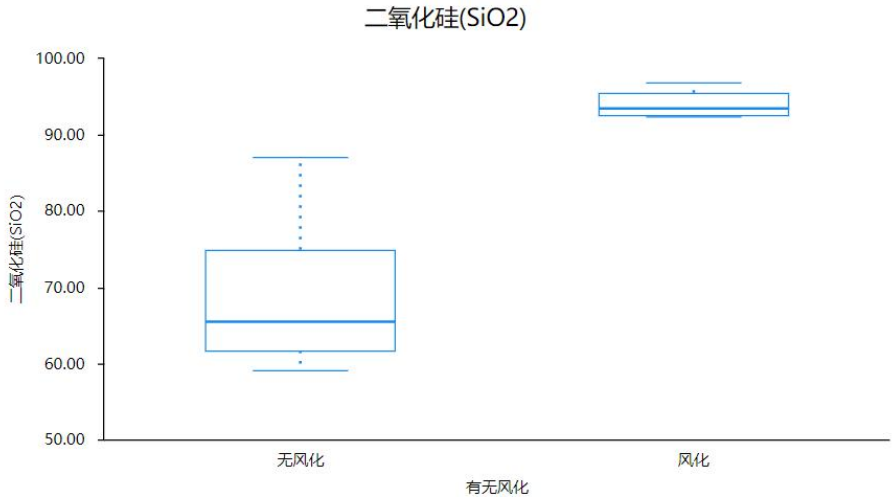
	有无风化(平均值±标准差)		F	p
	无风化(n=12)	风化(n=6)		
二氧化硅(SiO2)	67.98±8.76	93.96±1.73	50.332	0.000**
氧化钠(Na2O)	0.69±1.29	0.00±0.00	1.697	0.211
氧化钾(K2O)	9.33±3.92	0.54±0.45	29.063	0.000**
氧化钙(CaO)	5.33±3.09	0.87±0.49	11.980	0.003**
氧化镁(MgO)	1.08±0.68	0.20±0.31	9.066	0.008**
氧化铝(Al2O3)	6.62±2.49	1.93±0.96	19.301	0.000**
氧化铁(Fe2O3)	1.93±1.67	0.27±0.07	5.813	0.028*
氧化铜(CuO)	2.45±1.66	1.56±0.93	1.464	0.244
氧化铅(PbO)	0.41±0.59	0.00±0.00	2.842	0.111
氧化钡(BaO)	0.60±0.98	0.00±0.00	2.160	0.161
五氧化二磷(P2O5)	1.40±1.43	0.28±0.21	3.531	0.079
氧化锶(SrO)	0.04±0.05	0.00±0.00	4.312	0.054
氧化锡(SnO2)	0.20±0.68	0.00±0.00	0.485	0.496
二氧化硫(SO2)	0.10±0.19	0.00±0.00	1.747	0.205

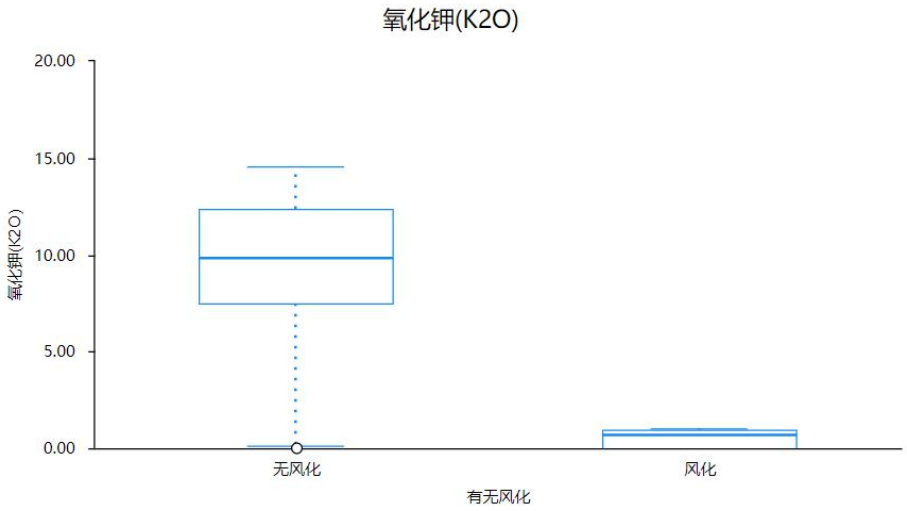
表 2 有无风化与化学成分的方差分析

根据数据，有无风化两组样本对于氧化钠、氧化铜、氧化铅、氧化钡、五氧化二磷、氧化锶、氧化锡、二氧化硫共 8 项化学成分并不会表现出显著性差异，而有无风化样本对于**二氧化硅、氧化钾、氧化钙、氧化镁、氧化铝、氧化铁**共 6 项呈现出显著性差异。

为进一步了解这些化学成分在有无风化样本中的差异表现，选取 p 值较小的若干化学成分含量（图中为二氧化硅、氧化钾、氧化铝），绘制这些指标的箱线

图，用于反映不同样本中数据的区间范围和平均值相对位置。通过观察这三个图形，可以发现两个样本的三个指标数据基本上无交集，平均值的差异足够大，证明有无风化对指标数据的分布有显著性影响。





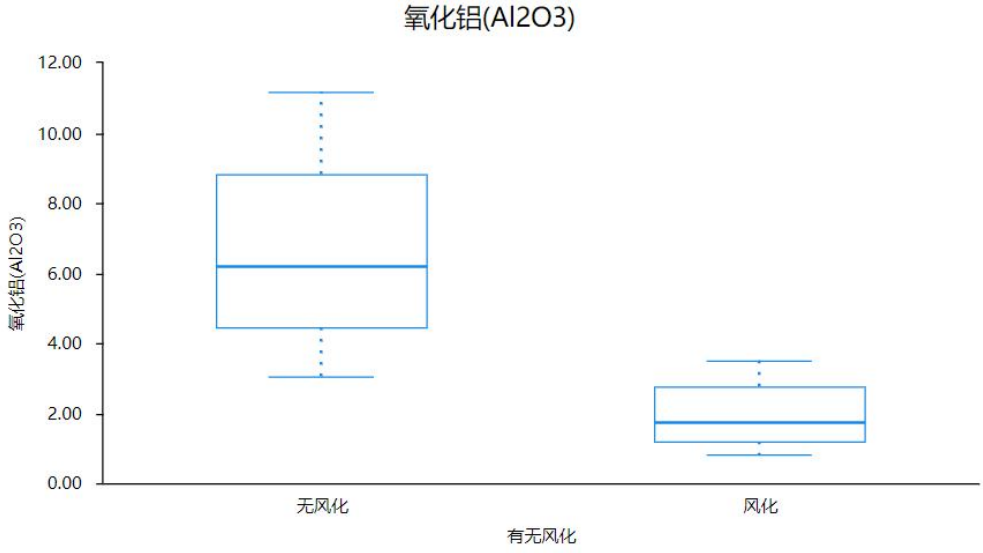


图 2 三种化学成分在风化与未风化条件下的分布箱式图

风化对各化学成分含量产生的影响，可求各风化敏感型变量在风化前后平均值，并比较平均值的变化方向和变化比例，如下表所示：

化学成分	二氧化硅	氧化钾	氧化钙	氧化镁	氧化铝	氧化铁
风化前平均值	67.98	9.33	5.33	1.08	6.62	1.93
风化后平均值	93.96	0.54	0.87	0.20	1.93	0.27
平均值变化方向	升高	降低	降低	降低	降低	降低
平均值变化比例	+38%	-94%	-84%	-81%	-71%	-86%

表 3 高钾玻璃风化前后风化敏感型变量的变化

与高钾玻璃同理，处理铅钡玻璃数据同样可以得到在有无风化条件下有显著性差异的 6 种化学成分含量，分别是**二氧化硅、氧化钠、氧化钙、氧化铅、五氧化二磷、氧化锶**。对比两种类型的玻璃，可以发现二氧化硅和氧化钙的含量在两种类型中和在风化前后均产生了显著性变化，揭示这两种化学成分含量的变化可能是埋藏文物化学成分含量变化的共同特征。我们定义这些有显著性变化的化学成分为**风化敏感型变量**。铅钡玻璃中风化敏感型变量的变化情况如下表所示：

化学成分	**二氧化硅**	氧化钠	氧化钙	**氧化铅**	五氧化二磷	氧化锶
风化前平均值	54.66	1.68	1.32	22.08	1.05	0.27
风化后平均值	24.91	0.22	2.70	43.31	5.28	0.42
平均值变化方向	减小	减小	增大	增大	增大	增大
平均值变化比例	-54%	-87%	+104%	+96%	+403%	+55%

表 4 铅钡玻璃风化前后风化敏感型变量的变化

正态性检验

在同种类型和同种风化状况（无风化和风化）下的**风化敏感型变量**，其含量不与任何已知的变量相关，取值服从某种概率分布。为检验其分布的类型，我们从两方面入手。首先是直观观察方面，绘制化学成分含量分布的直方图，通过观

察分布的图形形式来粗略估计其分布的类型；其次是利用数据做正态性检验，使用定量的数据来评判其与正态分布的接近程度。

下面为了节省篇幅，将直方图和与正态分布的接近程度指标画在一张图中。下图描述了在**铅钡玻璃类别**下，6 个风化敏感型变量在风化前后的分布。

在此处，我们选取的正态性检验方式为夏皮洛-威尔克检验，一般又称 W 检验。W 检验是一种类似于利用秩进行相关性检验的方法。W 检验的原假设是“样本数据来自的分布与正态分布无显著差异”，因此检验结果 P>0.05 表示没有呈现出显著性，说明样本数据与正态分布无显著差异。由图可见，铅钡玻璃中，**风化组和无风化组均呈现正态分布的化学成分只有二氧化硅和氧化铅**，其余均不符合正态分布。

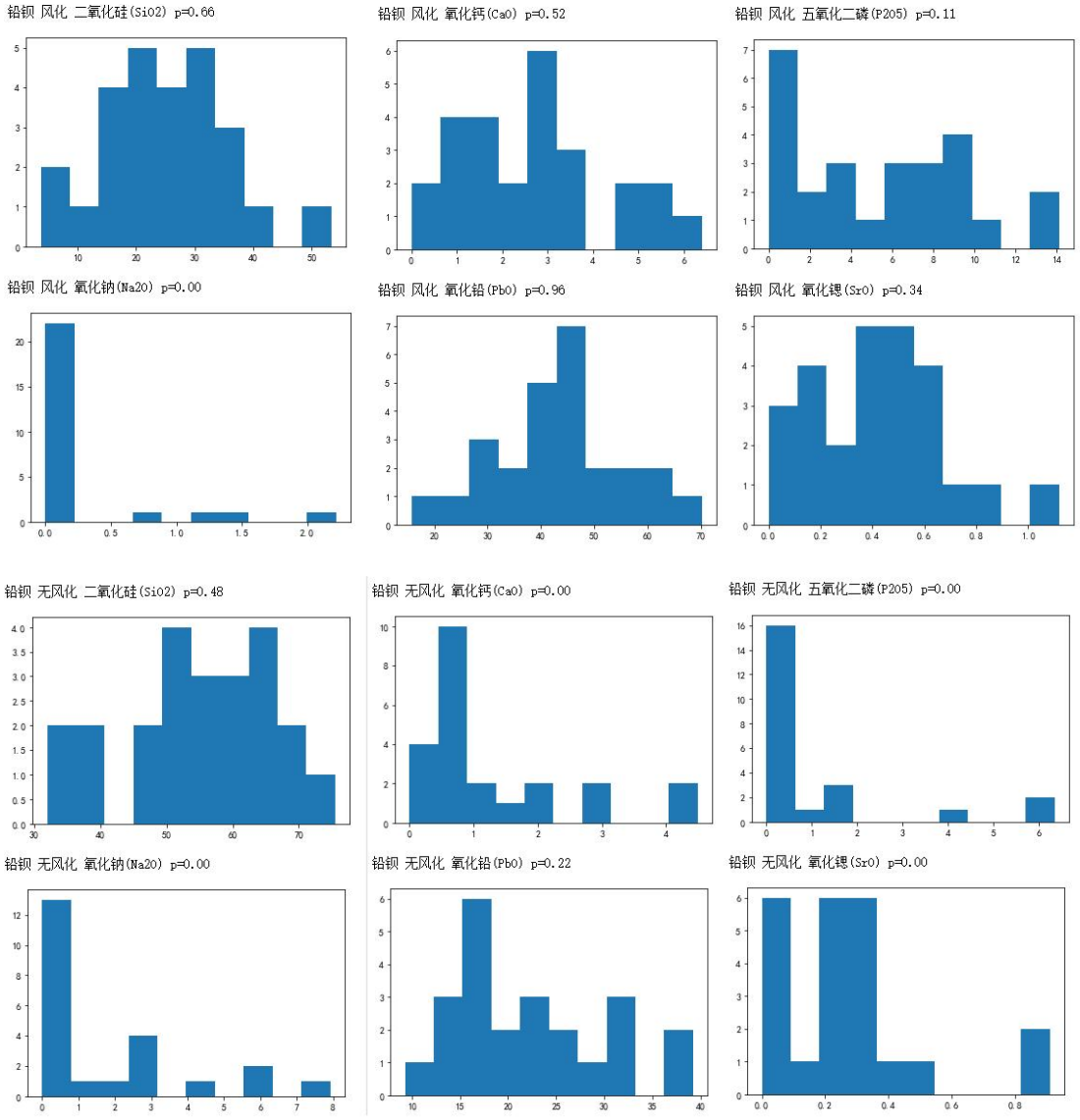


图 3 铅钡玻璃中风化敏感性化学成分变量分布直方图

同样地，在高钾玻璃中，对 6 个风化敏感型变量绘制直方图，并做正态性检验，得到的风化前后均服从正态分布的化学成分有：二氧化硅、氧化钾、氧化钙、氧化铝、氧化铁。剩余的氧化镁不全服从正态分布。

通过阅读文献可知，古代玻璃风化前后有些化学成分会有明显的变化，且不同类型的玻璃的化学成分变化方向可能不同，是因为不同玻璃种类中不同化学成分分别发生了一些物理化学反应，这也印证了我们的分析。

5.1.3 预测风化前的化学成分含量

模型建立

风化不敏感型变量在有无风化两种条件下没有显著差别，因此可以通过求解其在同种类型玻璃下的含量分布，从分布中随机获得一个取值，且这个取值应当与**同种类型、工艺、纹饰**的玻璃的风化前的含量相近。这些化学成分含量在风化前后没有明显的变化，说明它们大部分来自初始时的铸造，而铸造时相应化学成分的含量则与制造参数（工艺、纹饰）有很大关系。因此，可以取对应制造参数且在风化前的所有玻璃制品的化学成分含量的平均值，以此作为这些风化不敏感型变量的**估计值**。

对于**风化敏感型变量**，可以肯定的是风化对其产生了显著的影响，为了了解其风化前的含量数值，需要了解风化的过程对它产生了怎样的**定量影响**。由 1.2 中正态检验可知，一部分风化敏感型变量在风化前后均呈现正态分布。大量的实践表明，在相同的条件下生产的一批产品的质量指标，如本题中风化前的某种化学成分含量，可以看做或近似看做是服从正态分布的。如果经过检验其确实服从正态分布，并且经过风化后的化学成分含量也服从正态分布，**假设**同一类文物的化学成分含量大致以同样的速率和方向变化，那么从风化前的正态分布到风化后的正态分布转移的过程应当是一个确定的映射关系。

设风化前的某物质含量为c，且 $c \sim N(\mu, \sigma^2)$；经过风化后，物质含量**显著改变**，从 c 变为 c'，且 $c' \sim N'(\mu', \sigma'^2)$。由于风化的过程是缓慢、速率稳定的，因此，一个服从N分布的 c 必定能确定地映射到 服从 N' 分布的c'。假设c 到 c' 的映射是线性的，那么为了保证映射后的均值和方差相等，可以令

$$c' = \mu' + \frac{c - \mu}{\sigma} * \sigma'$$

可以证明，这样构造出的c'仍然服从正态分布N'.

通过这种方法同样可以得到c'到c的映射，也就是完成题目要求的预测任务。映射得到c后，检验c是否与相应工艺、颜色的玻璃中风化前 c 的数值接近，如接近，就说明我们的假设是成立的。

对于风化前后不满足正态分布的风化敏感型变量，仍可采用上述方法预测风化前的数值。即求出风化前后样本的均值和标准差，按照线性关系将风化后的数值映射为风化前的数值。因为这种映射考虑到了风化后的值对风化前的值的影响，使得风化后的值越大，对应的风化前的值也相对较大，保留了风化前后化学分含量的相对大小关系，在实际文物考古研究中有一定的意义。

基本模型总结如下：

（1）对于两个类别中的风化不敏感型变量，其取值与初始制造时的工艺，即纹饰和颜色有关，可取相同类别中纹饰和颜色相同的玻璃样本中化学成分的均值作为这些变量的估计值。

（2）对于两个类别玻璃中的风化敏感型变量，由于其中大多数化学成分含量在风化前后均呈现正态分布，因此可通过求解风化前后的平均值μ和标准差σ，通过线性映射的方式预测风化前的化学成分含量。之后可通过检验预测值与同种工艺玻璃风化前化学成分含量的符合程度，以此验证预测的准确性。

结果分析

在求出风化敏感型变量的预测值之后，可以将其与同种工艺参数的一批无风化玻璃样本做比较，并将所有化学成分的含量加和验证其值是否在 100%附近，以检验样本的准确性。

利用上述计算方法求出的风化样本在风化前各个化学成分含量的预测值，其数据在支撑材料的“风化前预测值.xlsx”里面。在总计 32 个样本中，共计有 25 个样本的化学成分含量总和取值为 85%~105%之间，占比为 78%；另有 29 个样本的化学成分含量总和取值为 85%~115%之间，占比为 91%。这些数据表明大部分样品风化前化学成分含量的预测值加和和合理区间之内，表现出我们提出的预测模型具有**自洽性**。

然后，选取多个典型样本，将其风化敏感型变量风化前的预测值与同种制造工艺（类型、颜色、纹饰相同）的未风化样本作对照，结果如下：

组别	二氧化硅	氧化钾	氧化钙
预测组 1	69.71	13.64	4.33
预测组 2	73.58	9.76	3.67
无风化样本均值与标准差	68.6 +- 9.16	7.6 +- 3.81	4.63 +- 2.51
范围	1σ内	2σ内，1σ内	1σ内

表 5 高钾玻璃预测值与相同工艺未风化组差别比较

组别	二氧化硅	氧化钙	氧化铅
预测组 1	58.9	2.77	15.96
预测组 2	46.9	1.70	22.54
标准组	49.8 +- 4.8	1.6 +- 0.52	26.8 +- 3.8
范围	1σ~2σ	1σ~2σ	$1\sigma, 3\sigma$之内

表 6 铅钡玻璃(A，铅钡，黑)预测值与相同工艺未风化组差别比较

组别	二氧化硅	氧化钙	氧化铅
预测组 1	70.9	0	20.9
预测组 2	63.6	0	26.1
预测组 3	66.7	0	24.2
标准组	67.8 +- 7.7	0.76 +- 0.13	16.7 +- 0.54
范围	1σ内	差距较小	$> 3\sigma$

表 7 铅钡玻璃(C，铅钡，深绿)预测值与相同工艺未风化组差别比较

由上可知，两种玻璃风化组中风化敏感型变量的预测值与工艺参数相同的标准组的预测值较为接近，但是有些工艺参数相同的组由于样本量较少（仅有 1~2 个样本），所以其取值很接近，标准差较小；但是预测组风化后的数据样本量较多，成分含量相差较大，预测的结果难免与同工艺组的风化前数据有所不同，但

总体上可以接受。

5.2 玻璃制品的分类规律与亚类划分

5.2.1 玻璃制品的分类规律

这一问实际上是要求只使用附件中提供的化学成分含量的特征，来判断该玻璃文物所属的类型。在找出玻璃样品类别的分类规律后，就可以用于问题 3 中对未知类别玻璃文物所属类型的分析。

模型分析

决策树是一种逻辑简单、易于理解的机器学习算法，采用树形结构，使用层层推理来实现最终的分类。决策树由根节点、内部节点、叶节点三种元素组成。根节点表示样本的全集，预测时，从根节点出发，经过内部节点的判断条件选择树上的路径，最终达到叶节点的分类结果。这些规则都是通过传入数据后进行训练得到的，训练的目标是对于所有训练样本，经过决策树分类得到的样本类型能与其标签类型一致。我们通过决策树得到一组树状的分类规则之后，就可以将任意样本传入决策树，实现分类任务。因此，我们可以用决策树来完成玻璃文物的分类任务。

CART 决策树算法使用基尼系数选择特征，基尼系数代表了模型的不纯度，基尼系数越小，纯度越高，特征划分越好。

基尼值的计算公式如下：

$$Gini(D) = \sum_{i=1}^{n} p(x_i) * \left(1 - p(x_i)\right) = 1 - \sum_{i=1}^{n} (p(x_i))^2$$

其中$p(x_i)$是分类x_i出现的概率。直观来说，Gini（D）反映了从数据集 D 中随机抽取两个样本，其类别标记不一致的概率。Gini 值越小，数据集 D 的纯度越高。在我们的问题中，数据集 D 指的是表单 2 中所有文物，其特征为化学成分含量，分类类别有高钾和铅钡两类。

对于样本 D，根据特征 A 是否取某一个可能值，把样本分为两部分D_1和D_2. 在属性 A 的条件下，样本 D 的**基尼系数定义为[2]：**

$$GiniIndex(D|A = a) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

基尼系数表征了划分后各个样本基尼值（样本纯度）的加权平均值，基尼系

数越小，表示划分后样本的在各个分数据集下的纯度总体增大；

因此，我们在候选特征集合S_f中，选择那个使得划分后基尼指数最小的特征A^*作为最优划分属性，即：

$$A^* = arg\min_{A \in S_f} GiniIndex(D, A)$$

一般的决策树算法使用的特征都是离散的数据，**而我们的题目中使用到的化学成分含量是连续型变量。**CART 分类树的思想是把连续的特征离散化，若连续特征 A 有 m 个取值，从小到大为$a_1, a_2, a_3, ..., a_m$,则 CART 选取相邻两样本值的平均数做划分点，一共 m-1 个，第 i 个为$T_i = (a_i + a_{i+1})/2$.

在决策树某个节点选择特征时，遍历所有特征；对于每一个特征，分别计算以这 m-1 个点作为二元分类点时的 GiniIndex，选择基尼系数最小的点为该连续特征的最优划分点，相应记录其最小的基尼指数；最后选择所有特征中最优划分点基尼指数最小的特征及其最优划分值，这样就找到了最优的划分点。重复以上步骤，递归生成分类树即可。

用 A 表示特征，A^*表示最优特征，t 表示划分点，t^*表示最优划分点，T 为划分点的集合，GiniIndex(D, A, t)表示用 A 特征的 t 划分点划分 D 数据集得到的基尼指数，公式表示如下：

$$A^* = \arg\min_{A \in S_f} \left(\min_{t \in T_A} GiniIndex(D, A, t) \right)$$

$$t^* = arg\min_{t \in A^*} GiniIndex(D, A^*, t)$$

A^*和t^*就是我们每层划分的依据。例如，在决策树的某个节点划分时，在 14 个化学成分含量中，求最优的划分和最优的特征，如二氧化硅中划分点为 71 时的基尼指数最小，为二氧化硅中最优的划分点，且该划分点的基尼指数又是所有特征中最优划分点基尼系数的最小值，那么A^*=二氧化硅，$t^* = 71$.

建立 CART 分类树的方法 [1]是从根节点开始，用训练集递归建立 CART 分类树。具体算法细节可以参照附录 2。

结果分析

按照算法流程训练决策树，通过缺省部分特征可以得到三个决策树。其中，在完全训练集的条件下，算法训练出的最优决策树为图 1，叶子结点划分为表 6。可以看到，该决策树中只有一个分支节点 PbO 的含量，而观察数据后，我们也

发现铅钡玻璃和高钾玻璃的氧化铅含量呈现出非常明显的差异：铅钡玻璃的氧化铅远远高于 5.46（范围为 9.3~70.2），高钾玻璃的氧化铅含量远远低于 5.46（范围为 0~1.62）。这也印证了我们决策树划分的合理性。我们所找的文献也支持了高钾玻璃和铅钡玻璃氧化铅含量存在显著性差异的结论。

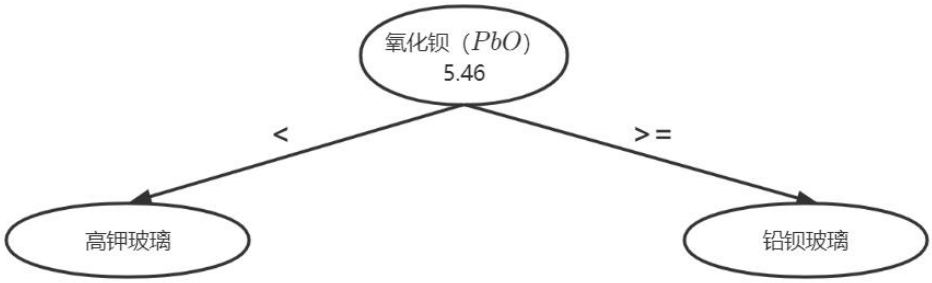


图 4 无限制条件下最优决策树 1

高钾玻璃（18）	01、03 部位 1、03 部位 2、04、05、06 部位 1、06 部位 2、07、09、10、12、13、14、16、18、21、22、27
铅钡玻璃（49）	02、08、08 严重风化点、11、19、20、23 未风化点、24、25 未风化点、26、26 严重风化点、28 未风化点、29 未风化点、30 部位 1、30 部位 2、31、32、33、34、35、36、37、38、39、40、41、42 未风化点 1、42 未风化点 2、43 部位 1、43 部位 2、44 未风化点、45、46、47、48、49、49 未风化点、50、50 未风化点、51 部位 1、51 部位 2、52、53 未风化点、54、54 严重风化点、55、56、57、58

表 8 最优决策树 1 叶子结点数据集划分

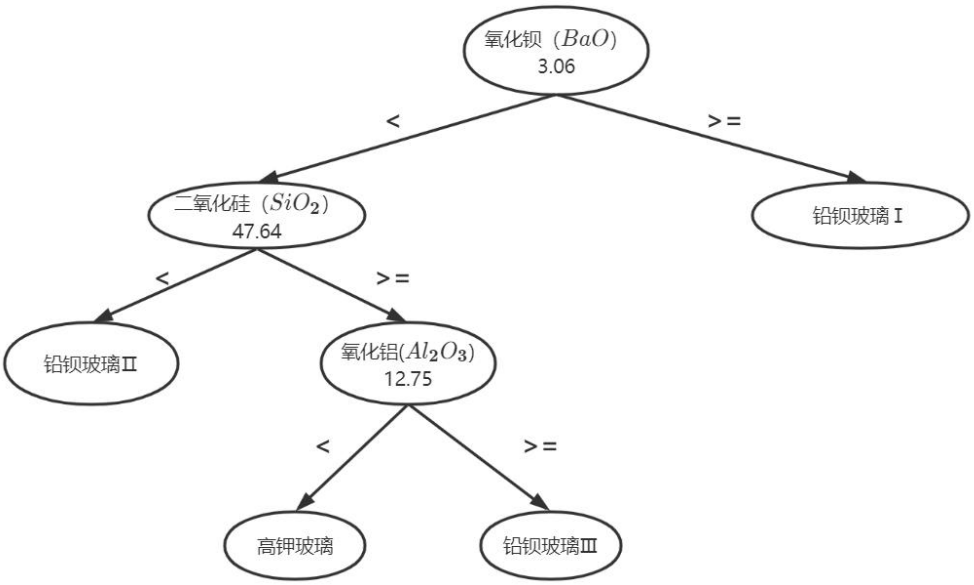


图 5 去掉特征氧化铅之后生成的最优决策树 2

高钾玻璃（18）	01、03 部位 1、03 部位 2、04、05、06 部位 1、06 部位 2、07、09、10、12、13、14、16、18、21、22、27
铅钡玻璃I（45）	08、08 严重风化点、11、19、20、23 未风化点、24、25 未风化点、26、26 严重风化点、28 未风化点、30 部位 1、30 部位 2、31、32、33、34、35、36、37、38、39、40、、41、42 未风化点 1、42 未风化点 2、43 部位 1、43 部位 2、44 未风化点、45、46、47、48、49、49 未风化点、50、50 未风化点、51 部位 1、52、53 未风化点、54、55、56、57、58
铅钡玻璃II（3）	02、51 部位 2、54 严重风化点
铅钡玻璃III（1）	29 未风化点

表 9 最优决策树 2 的叶子结点划分

去掉氧化铅这一显著特征之后，用其它特征对决策树再次训练，得到一种新的最优决策树 2，如图 2 所示。可以看到，判定高钾玻璃的依据为(BaO < 3.06, SiO2 >= 47.64, 且 Al2O3 < 12.75)，而判定铅钡玻璃的依据则有(BaO >= 3.06), (BaO < 3.06, 且 SiO2 < 47.64), 和 (BaO < 3.06, SiO2 >= 47.64, 且 Al2O3 >= 12.75)三种。

决策树 2 虽然结构比最优决策树 1 略微复杂，但是在主要指标氧化铅受到干扰，或因仪器问题测量时，此决策树可作为一种重要的备选方案。经过验证，最优决策树 2 对表单 2 已分类样本的分类结果与标签值一致。

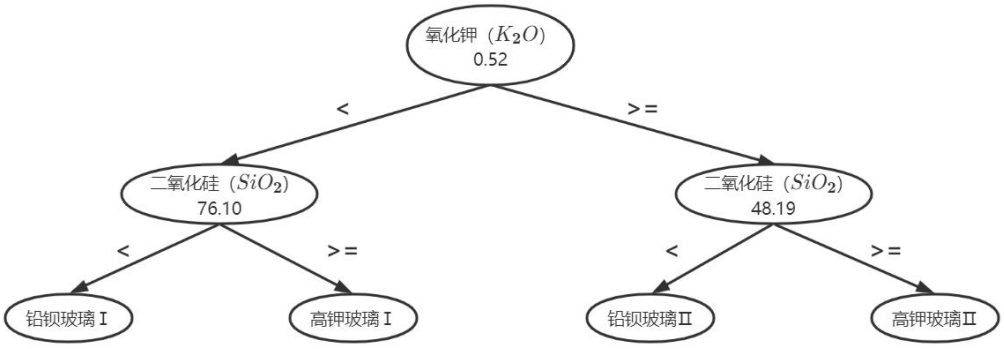


图 6 去除特征氧化铅和氧化钡之后得到的最优决策树 3

将数据集中去掉氧化铅和氧化钡特征之后，得到的最优决策树如图 3 所示，其分类结果与标签值一致，在此不再赘述。能够在不同数据集上成功训练出三棵决策树也证明我们的 CART 决策树生成算法较为合理。

模型泛化

另外，原则上，为了防止我们的决策树过拟合，需要把风化和无风化作为单独的数据集进行训练。但实践发现，我们的数据集数据量较小，是否区分风化不会对决策树的复杂程度产生太大的干扰，所以可以将风化数据和无风化数据混合在一个样本中来训练决策树。但处理特别的任务时，如区分已风化的玻璃所属类型时，最好针对具体的样本类型单独训练决策树，以提高模型的决策能力。

5.2.2 玻璃制品的亚类划分

模型建立过程与结果

本问题需要给已知各种化学成分的高钾玻璃和铅钡玻璃分别进行亚类划分。由于没有给出亚类划分的依据，我们不能使用常规的数值方法对其进行分类，应该选用无监督聚类算法，将化学成分相似的样本自动归入一个类别。这里我们使用无监督聚类算法 k-means。

k-means 算法是最常用的无监督聚类算法，其主要思路是将样本点之间距离较近的分入一个聚簇（亚类）中，同时使得不同聚簇（亚类）的样本点之间的距离尽可能地远。这样就可以仅通过化学成分差异得到较为准确的亚类划分。

这里需要给 k-means 算法的初始 k 值和初始类簇中心点。由于 k 值的选取和初始类簇中心的点的选取会大大影响算法的收敛速度和聚类结果，我们分别采用

肘击法和 k-means++算法，以得到最优的 k 值和最优的 k 个初始聚簇中心。完整算法流程见附录 2，具体代码见附录 3。

聚类分析算法把高钾和铅钡玻璃两种样本中，每一种类别都划分为 A、B、C、D 四个亚类。值得注意的是，我们在聚类分析时，使用的是若干采样点的采样数据，而同一个文物可能因为采样部位的不同而有多个采样点数据，所以在聚类分析完成后，需要将采样点编号对应到其所属的文物编号。高钾玻璃和铅钡玻璃的亚类划分结果如下图所示：

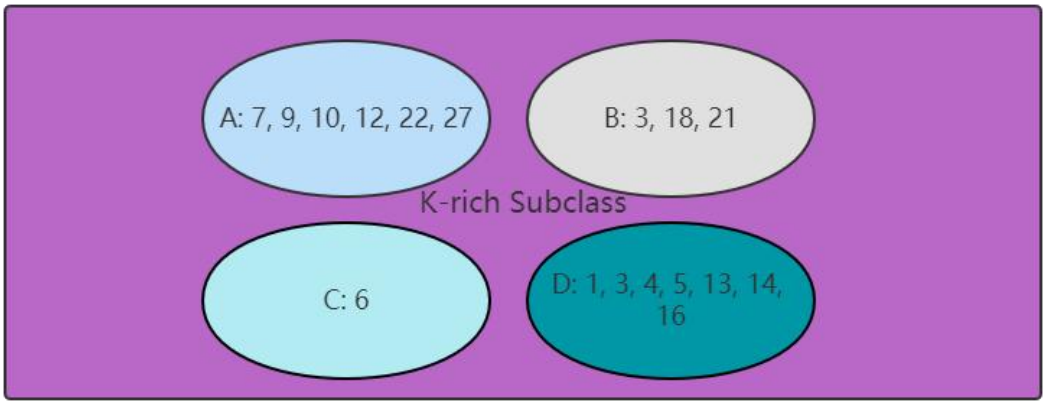


图 7 高钾玻璃的亚类划分结果

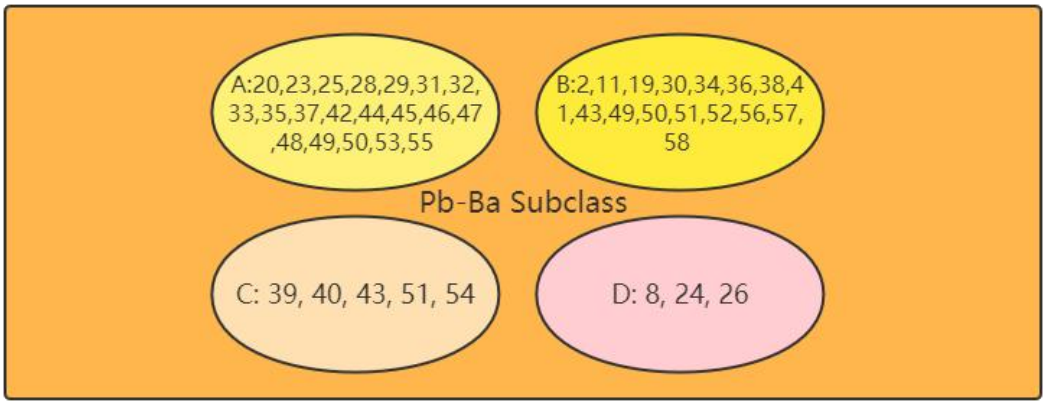


图 8 铅钡玻璃的亚类划分结果

合理性分析

K-means 聚类分析中 K 值选取和 K 个初始簇中心选取会极大的影响模型聚类效果。下面就这两点分别进行分析：

（1） 对于聚类中心的选取方法，我们初始时随机在数据集中选择一个样本点作

为聚类的中心。接着逐步选择离已确定的聚类中心点较远的样本点，重复这个过程，直到 k 个聚类中心都被确定。初始时的聚类中心越分散，最终得到的若干个亚类之间的差异就越大，这样就增强了模型的分隔度。

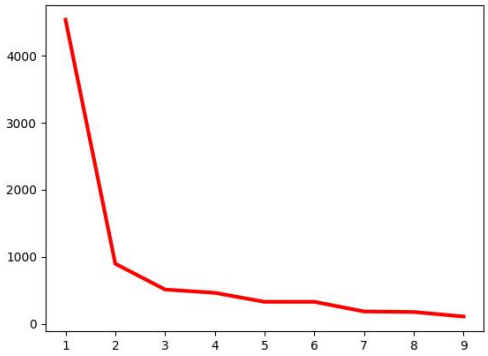
（2） K 值选取——手肘法：

定义聚类的误差平方和 SEE，其公式为：

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2$$

其中，C_i是第 i 个类，p是C_i类中的所有样本点，m_i是C_i类的中心。

以选取的 k 值为横坐标，纵坐标是 SEE 值，对高钾玻璃聚类时其变化曲线为：



铅钡玻璃的变化曲线：

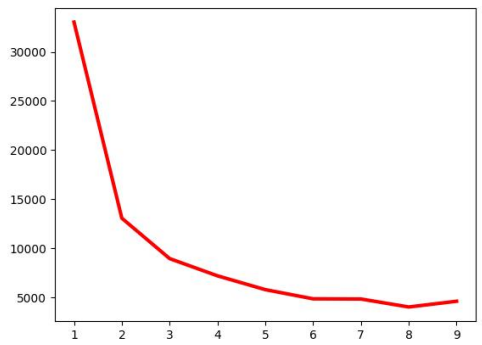


图 9 k-SSE 值变化曲线

随着聚类数 k 的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和 SSE 自然会逐渐变小，当 k 小于最优聚类数时，

由于 k 的增大会大幅增加每个簇的聚合程度，故 SSE 的下降幅度会很大；而当 k 到达最优聚类数后，再增加 k 所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着 k 值的继续增大而趋于平缓，也就是说 SSE 和 k 的关系图是一个手肘的形状，而这个肘部对应的 k 值就是数据的最优聚类数。

对于高钾玻璃，我们发现 K 值选取 3，4 均可，为了更精准地划分，我们取 4；而对于铅钡玻璃，我们发现 K 值取 4 较好。

（3）划分结果——min SEE

由于 K-means 聚类的随机性，每次亚分类结果可能不相同。为此，我们进行 1000 次实验，对每次实验结果计算 SEE 值，例如高钾玻璃亚类划分：

SEE	类别 1	类别 2	类别 3	类别 4	出现次数
379.6667	[5, 6]	[1, 14, 15]	[0, 2, 3, 4, 11, 12, 13]	[7, 8, 9, 10, 16, 17]	56
386.2112	[5, 6]	[14, 15]	[1, 7, 8, 9, 10, 16, 17]	[0, 2, 3, 4, 11, 12, 13]	65

表 10 高钾玻璃亚类划分结果最好的两例

注：类别内标号为训练样本编号，并非文物编号。

比较发现，最小的两个 SEE 值对应的类别划分差别不大，且其出现次数占总次数的 12.1%，因此据此得到最优亚分类结果（见图 9、10）。后续我们分别进行了高钾玻璃和铅钡玻璃各 1000 次亚分类实验，计算每一个亚分类集结果与最优亚分类结果的杰卡德相似系数，并求平均值，发现 $averageJ_K$=0.89，$averageJ_{PbBa}$=0.71，接近于 1，说明亚分类相似度高，证明了划分结果的合理性。

敏感性分析

为定量分析模型敏感性，我们采用杰卡德相似系数衡量相似度，公式如下。

$$J(C,C') = \sum_{i=1}^{K} \max_{j=1\ldots K} \frac{C_i \cap C_j'}{C_i \cup C_j'}$$

其中，C,C’表示我们的划分结果，例如 C={[1,3,5],[2,6,7,9]，……}（每一个中括号为划分的亚类，数字为文字编号），C_i表示第 i 个亚类。显然有 $0 \leq J(C,C') \leq 1$。$J(C,C')$即可衡量划分集合(C,C')的相似程度。

由于 K-means 聚类的随机性，我们用多次实验的结果来衡量变量变化。现对 1000 次亚分类实验结果，相邻两个结果计算杰卡德相似系数，然后求均值。

（1） 如果该系数趋向于 0，则说明这 1000 次实验结果差异很大，由于我们使用的同样的样本，所以该差异仅是由于 K-means 聚类的随机性导致的，说明我们的模型敏感性太高。

（2） 如果该系数趋向于 1，则说明这 1000 次实验结果十分相似，说明 K-means 聚类的随机性对模型影响很小，说明我们的模型敏感性太低。

通过计算，得到高钾玻璃的杰卡德系数均值为$averageJ_K(C,C')$=0.66，铅钡玻璃的杰卡德系数均值为$averageJ_{PbBa}(C,C')$=0.64，说明模型敏感性较好。

亚类划分的化学成分依据

我们根据每一类别的亚类划分结果，分别制作了不同亚类各化学成分含量比较图，高钾玻璃的亚类分类结果如图所示。

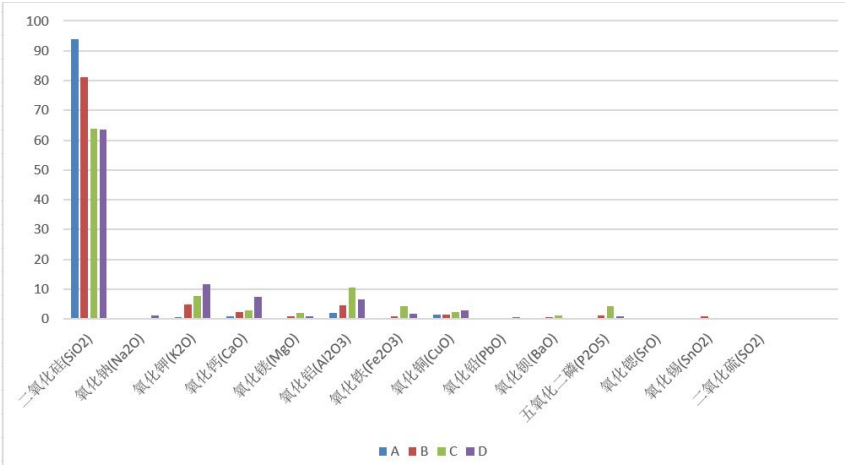


图 10 高钾玻璃亚类各化学成分含量比较

我们可以**选择合适的化学成分来解释亚类的划分**。在高钾玻璃中，A 亚类二氧化硅含量均值高达 90%以上，经过与表单 2 的验证，可以发现 A 亚类的所有文物都是风化的文物，这说明我们的聚类分析算法成功在混杂有风化和无风化的样本数据中区分出了只含有风化样本的亚类，这恰恰体现出我们所建立模型的敏感性。B 亚类二氧化硅含量均值在 80%左右，C、D 亚类二氧化硅含量均在 60%左右，单纯通过二氧化硅含量无法区分两个亚类；但是 C 亚类氧化铝含量较高，在 10%左右；D 亚类氧化钾和氧化钙含量较高。可以依照这一原则对 C、D 亚类

做进一步的划分。

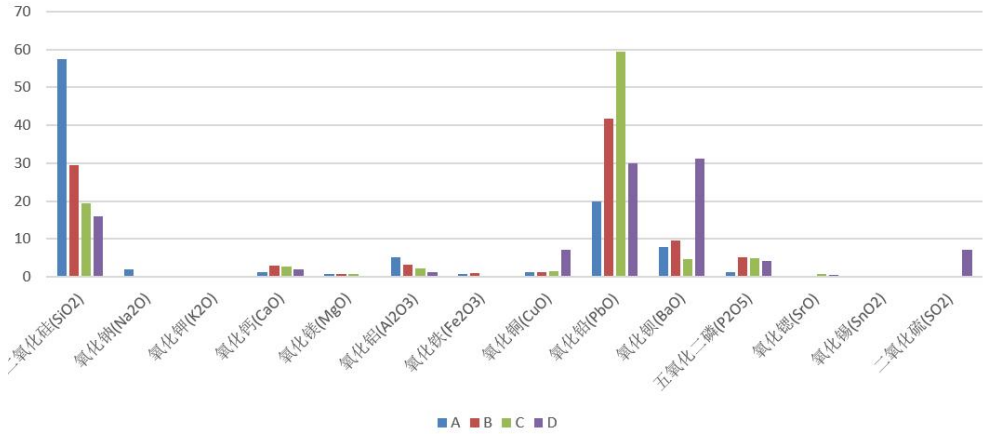


图 11 铅钡玻璃亚类各化学成分含量比较图

在铅钡玻璃中，如图 7 所示，A 亚类二氧化硅含量均值相对较高，为 50%以上；C 亚类相对氧化铅含量较高，为 60%左右；D 亚类相对氧化钡含量较高，为 30%左右，高出其他类别近 20%；其余为 B 亚类。

5.3 未知类别玻璃文物的类型鉴别

5.3.1 结果分析

在 2.2.1 训练得到决策树的分类模型之后，用它去分类表单 3 中未知类别文物的所属类别，得到一个分类结果。由于我们在问题 2.1 中选取不同根节点特征生成了 3 棵最优决策树，因此需要同时使用 3 棵决策树对表单 3 中的未知玻璃样品类别进行分类。三棵决策树的决策过程和决策结果如下：

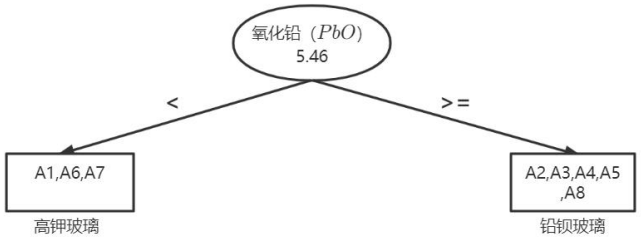


图 12 决策树一的鉴别结果

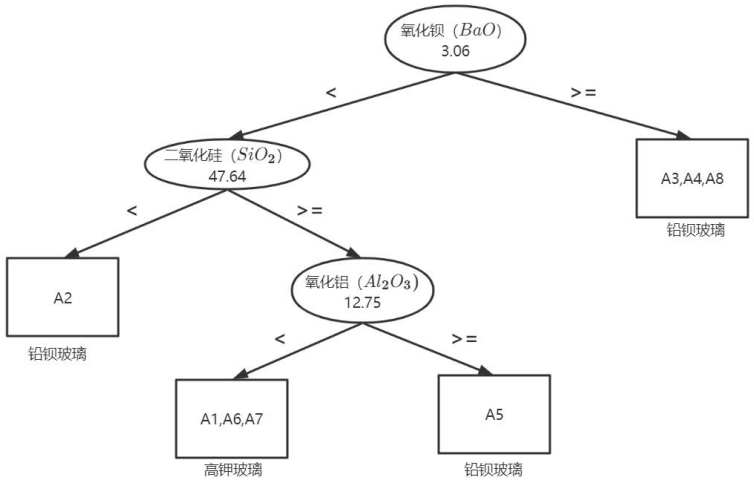


图 13 决策树二的鉴别结果

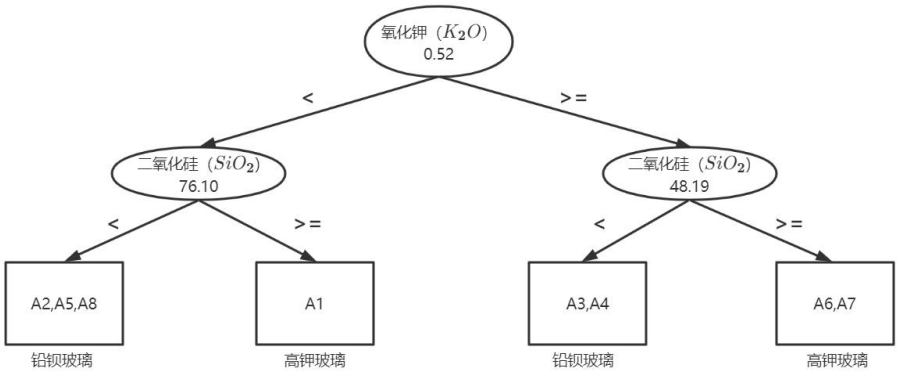


图 14 决策树三的鉴别结果

决策树类型	A1	A2	A3	A4	A5	A6	A7	A8
决策树一	高钾	铅钡	铅钡	铅钡	铅钡	高钾	高钾	铅钡
决策树二	高钾	铅钡	铅钡	铅钡	铅钡	高钾	高钾	铅钡
决策树三	高钾	铅钡	铅钡	铅钡	铅钡	高钾	高钾	铅钡

表 11 三个决策树的鉴别结果对比

在利用决策树一进行类型鉴别时，我们依据氧化铅的含量进行划分，含量小于 5.46 为高钾玻璃，大于等于 5.46 为铅钡玻璃。在对鉴别后玻璃中氧化铅含量分析，我们发现：鉴别为高钾玻璃的氧化铅含量均值为 0，显著低于划分标准 5.46；而鉴别为铅钡玻璃的氧化铅含量均值为 26.33，明显高于划分标准 5.46。说明以氧化铅含量作为划分标准，可以明确划分玻璃类型。下图 7 中左边的红色样本为分类出的高钾玻璃，右侧蓝色样本是铅钡玻璃，柱的高度代表氧化铅的含量，可

以看出分界值 PbO=5.46 可以很明显地将高钾样本和铅钡样本划分开。

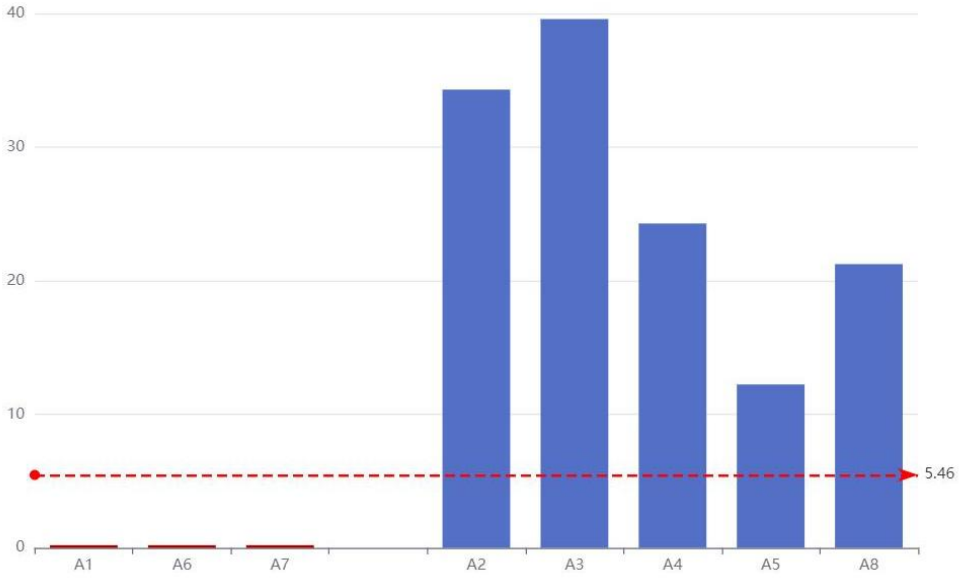


图 15 未知样本中高钾玻璃和铅钡玻璃氧化铅含量对比

为了验证分类结果的合理性，本题用 2.2.1 中得到的 3 棵决策树分别分类表单 3 中的文物，查看分类结果是否存在差异，如表 8 所示，可见三种决策树的分类结果是一致的。

5.3.2 模型泛化

另外，使用 3 棵决策树同时分类的优点如下：

（1） 分类结果如果无差异或者差异较小，说明 3 棵决策树之间可以相互检验正确性，决策树没有过拟合。因为如果有一个或多个决策树出现了过拟合，由于三棵决策树选取的特征不同，因此他们大概率会因为对不同特征的过度分类而与正常分类结果产生较大的偏差，且偏差的方向大相径庭，最终会导致分类结果之间出现很大的偏差。

（2） 三棵决策树选取特征之间存在差异，在被测样本某些特征缺省时，可以灵活选择适合的决策树进行分类，而不会因为特征的缺少而无法分类，表现出了模型的鲁棒性。

5.3.3 敏感性分析

在题目中提到，古代玻璃极易受埋藏环境的影响而风化。在风化过程中，内部元素与环境元素进行大量交换，导致其成分比例发生变化，从而影响对其类别的正确判断。为分析模型敏感性，我们在训练决策树模型时，对风化和未风化样

本单独训练决策树模型，与未考虑是否风化得出的三个决策树做对比：

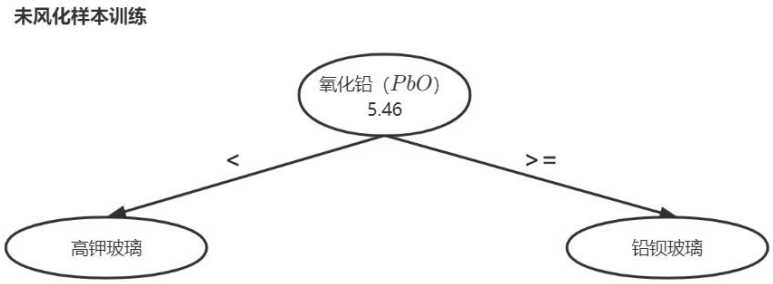


图 16 未风化样本训练得到的决策树

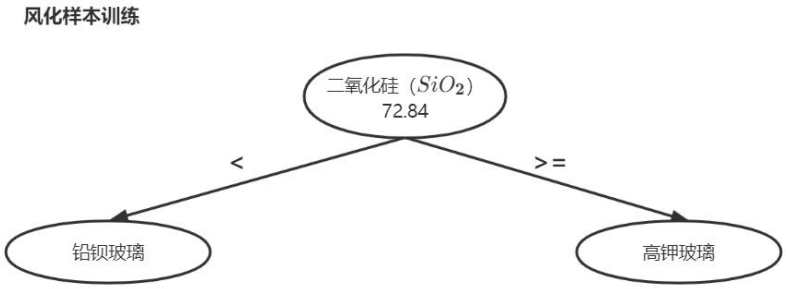


图 17 风化样本训练得到的决策树

据此对未知类型玻璃进行鉴别如下：

未风化				风化			
A1	A3	A4	A8	A2	A5	A6	A7
高钾	铅钡	铅钡	铅钡	铅钡	铅钡	高钾	高钾

表 12 未知类型玻璃鉴别结果

我们发现，用风化和未风化样本单独训练决策树模型，鉴别结果与未考虑风化影响鉴别出的结果一致。

5.4 不同类别玻璃文物中化学成分的关联关系

5.4.1 关联关系

为了评估不同类别中玻璃文物样品各化学成分之间的关联关系，可以使用相关系数来量化这个关系。常用的相关系数包括皮尔曼相关系数、斯皮尔曼相关系数，但皮尔曼相关系数仅仅适用于正态分布。由 5.1.2 可知，某些化学成分的含量并不服从正态分布，因此我们可以采用斯皮尔曼相关系数来分析不同类别玻璃

文物样品化学成分之间的关联关系。两种化学成分之间的斯皮尔曼相关系数绝对值越接近 1，说明两者的相关性越强。

得到斯皮尔曼相关系数后，使用 Spearman 热图可以直观地展示两两化学成分之间的相关系数，如下图所示：

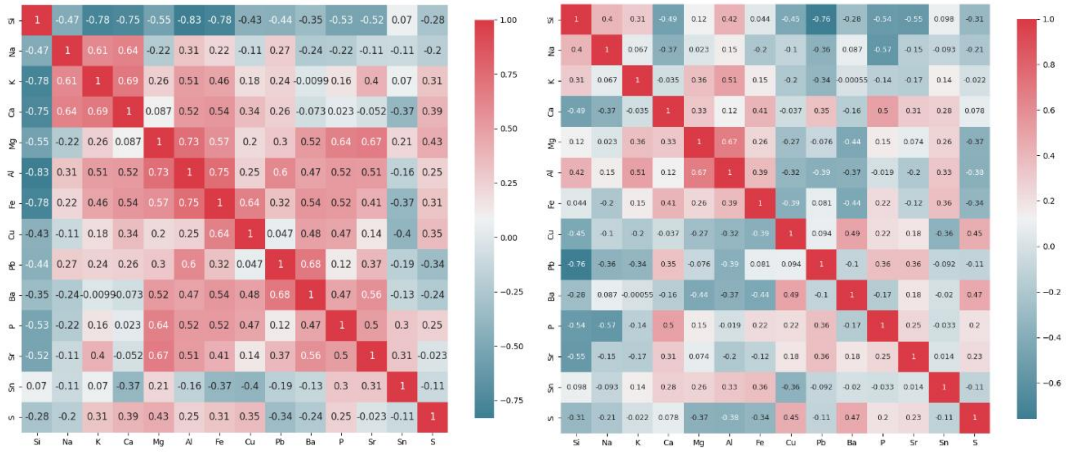


图 18 高钾和铅钡玻璃化学成分 Spearman 热图

Spearman 热图中的每一个单元格表示的都是行所对应的化学成分和列所对应的化学成分的相关系数。红色表示两者成正相关关系，蓝色表示两者成负相关关系，颜色越深，表示相关关系越强。对角线上表示化学成分与自身的相关关系，所以都为 1。

5.4.2 关联关系的差异性

对于高钾玻璃，有 18 个样本，为使两两化学成分存在相关关系的可能性达到 99%，在 n=18 且显著性水平为 0.01 时，我们取斯皮尔曼相关系数大于 0.55 的两种化学成分进行连线，以直观呈现一些较强的相关关系对。

对于铅钡玻璃，有 49 个样本，为使两两化学成分存在相关关系的可能性达到 99%，在 n=49 且显著性水平为 0.01 时，我们取斯皮尔曼相关系数大于 0.33 的化学成分对，然后进行连线。

将 14 种化学成分画在同一个圆上，如果有足够强的相关关系，就将它们连线，从而得到关联关系图，如下所示：

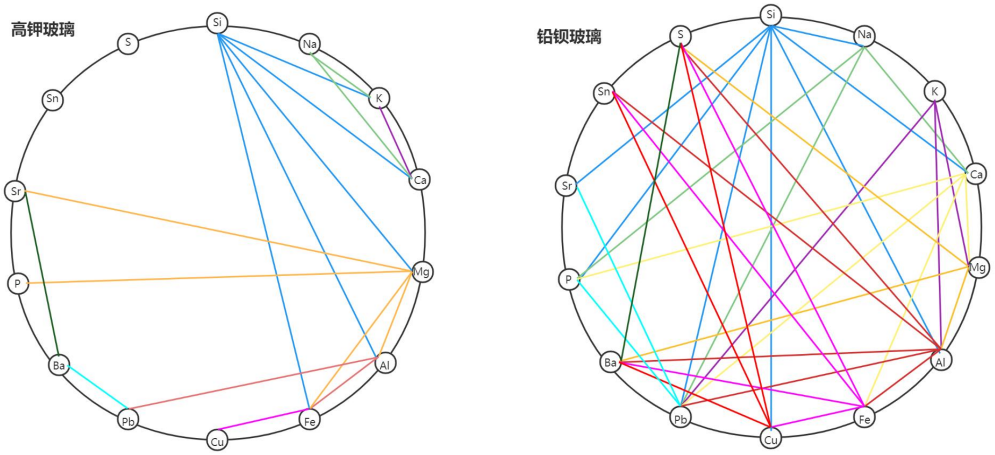


图 19 高钾和铅钡玻璃相关关系图

观察上图，可以很明显的发现铅钡玻璃中化学成分间有更强的相关关系。

六、 模型的评价与改进

6.1 模型优点

（1） 问题一在结合玻璃类型，分析文物样品表面有无风化化学成分含量的统计规律时，首先利用了方差分析，筛选出分化前后变化显著的化学成分，对他们进行分布拟合，而对于变化不显著的化学成分，结合颜色纹饰等信息，进行均值分析，使模型更加可靠。

（2） 问题二第一问通过建立 CART 分类决策树模型，进行分类规律研究，考虑到检测手段等实际情况，在对分类规律影响显著的化学成分缺少的情况下，模型训练出的决策树是否能合理的给出分类规律，我们建立了多颗决策树，并发现分类规律与相应的化学成分含量较为对应，使模型更加客观且真实。

（3） 问题三的对未分类玻璃类型鉴别问题采用了问题二的决策树，但是考虑到未分类玻璃数据中是否风化这一指标没有用到上一模型的决策树中，可能导致信息遗漏。因此，我们在问题二决策树模型基础上对是否风化样本分开训练，据此分开鉴别未知类别玻璃，使得模型可信度更高。

（4） 问题四在分析不同类别的玻璃文物样品化学成分之间的关联关系时，采用 Sspearman 热图呈现；比较不同类别之间的化学成分关联关系的差异性时，我们借鉴社交网络绘图，针对不同类别，分别绘制不同成分关联关系圆中连线图，

使模型更加直观，模型结果表现优秀。

6.2 模型缺点

（1） 在进行问题一玻璃文物的表面风化与颜色的关系进行分析时，对颜色缺失数据进行去除，忽略了这部分数据带来的影响，不够全面。

（2） 在建模过程中，例如 K-means 迭代结束条件优化，K 值选取等可以考虑更多条件。

（3） 在比较不同类别之间的化学成分关联关系的差异性，暂没有找到相关文献支持。

6.3 模型改进

（1） 问题二中的是通过决策树探究高钾玻璃、铅钡玻璃的分类规律，并使用 K-means 聚类进行亚分类，可以考虑更多的模型来解决。

（2） 问题四分析不同类别的玻璃文物样品化学成分之间的关联关系以及比较不同类别之间的化学成分关联关系的差异性时，可以查阅更多的参考文献。

七、 参考文献

[1] 决策树算法--CART 分类树算法，https://zhuanlan.zhihu.com/p/139523931

[2] 周志华，机器学习，清华大学出版社，2016

[3] Teresa Palomar, Chemical composition and alteration processes of glasses from the Cathedral of León (Spain), Boletín de la Sociedad Española de Cerámica y Vidrio, Volume 57, Issue 3, 2018, Pages 101-111

八、 附录

附录 1：支撑材料文件列表

附录 2：补充表格、图片和公式推导

附录 3：Python 代码

附录 1：支撑材料文件列表

文件名列表
C 位置标注版.xlsx
风化前预测值.xlsx
高钾玻璃化学成分.xlsx
铅钡玻璃化学成分.xlsx
决策树训练集.csv
高钾玻璃聚类分析数据集.csv
铅钡玻璃聚类分析数据集.csv
1000 次亚类划分与评价指标结果.xlsx
问题一相关代码
化学成分统计规律以及风化前化学成分含量预测.ipynb
问题二相关代码
决策树总代码.py
聚类分析总代码.py
问题三相关代码
决策树总代码.py
问题四相关代码
斯皮尔曼相关系数分析总代码.py

附录 2：补充表格、图片和公式推导

决策树训练算法：

（1）输入：

训练集 D，每个样本数据包括 14 种化学成分的含量，以及玻璃的类别

切分的最少样本个数阈值

（2）输出：

分类树 T，能根据 14 种化学成分的含量分类玻璃的类别

（3）算法流程：

1. 对于当前节点的数据集为 D，如果**数据集 D 中含有的样本全部属于一个类别**，则对数据集的再划分已没有意义，则返回决策子树，当前节点停止递归。所以，不必将样本的所有特征都用完，只要分割到所有叶节点

的样本都属于一个类别，就可以结束划分；

2. 计算样本集 D 的基尼系数，如果基尼系数小于阈值，则返回决策树子树，当前节点停止递归；
3. 计算当前节点现有各个特征的各个划分点的基尼系数；
4. 在计算出来的各个特征，各个划分点的基尼系数中，选基尼系数最小的特征 A 以及其对应的划分点 a 作为最优特征和最优切分点。之后根据最优特征和最优切分点，将本节点的数据集划分为两部分D_1和D_2，同时生成本节点的两个子节点，左节点的数据集为D_1,右节点的数据集为D_2.
5. 对左右的子节点递归调用 1-4 步，生成 CART 分类树。

   对生成的 CART 分类树做预测时，如果测试集中的样本落到了某个叶子结点，而该节点中有多个训练样本，则该测试样本的类别为这个叶子节点里概率最大的类别，**即数量最多的类别**。

k-means 算法：

（1）输入：

数据集 data，包括 n 个样本，每个样本数据包括 14 种化学成分的含量

希望生成的聚簇数 k

（2）输出：

聚簇集 cluster，包含 k 个聚簇，每个聚簇中的元素是玻璃文物的标签值。

聚簇中心集 cluster_center，为聚簇中心，与上述聚簇一一对应

（3） 算法流程：

1. 从 data 中随机选择 k 个样本点作为初始聚簇中心 $y_1, \dots, y_k$
2. 令 cluster 为空
3. 对于每个样本x_i，计算其与各个聚簇中心的距离，并将其划分之距离最近的聚簇中心y_j标定的聚簇中，$cluster_{y_j} = cluster_{y_j} \cup \{x_i\}$
4. 对于每个聚簇$cluster_{y_i}$，计算新的聚簇中心 $y_i' = \frac{1}{|C_i|}\sum_{x \in C_i} x$
5. 若$y_i \neq y_i'$，则更新聚簇中心为$y_i = y_i'$。若全部未更新，则停止迭代，输出聚簇集 cluster 和聚簇中心集$cluster\ center = \{y_1, \dots, y_k\}$，若有更新，则返回第 2 步。

k-means++算法：

（1）输入：

数据集 data，包括 n 个样本，每个样本数据包括 14 种化学成分的含量

希望生成的聚簇数 k

（2）输出：

聚簇中心集 cluster_center

（4） 算法流程：

1. 在 data 中随机选择一个样本点作为第一个初始化的聚类中心
2. 对于每个样本x_i，计算其与各个已初始化的聚簇中心的距离
3. 选择一个新的数据点作为新的聚类中心，选择的原则是：距离较大的点，被选取作为聚类中心的概率较大
4. 重复上述过程，直到 k 个聚类中心都被确定

附录 3：Python 代码

问题 1 相关代码：

```
import numpy as np
import openpyxl
from scipy.stats import kstest, shapiro
#cdf 中可以指定要检验的分布，norm 表示我们需要检验的是正态分布
#常见的分布包括 norm,logistic,expon,gumbel 等
configs = {
    "铅钡": {
        "xlsx": "铅钡玻璃化学成分.xlsx",
        "row": 50,
        "fenghua_col": "T",
        "list": ["B", "C", "E", "J", "L", "M"],
        "names": [],
        "data": {
            "风化": [],
            "无风化": [],
        },
    },
    "高钾": {
        "xlsx": "高钾玻璃化学成分.xlsx",
        "row": 19,
        "fenghua_col": "R",
```

```
        "list": ["B", "D", "E", "F", "G", "H"],
        "names": [],
        "data": {
            "风化": [],
            "无风化": [],
        },
    }
}
# 同时包括风化与无风化，铅钡和高钾，总计四类

for category in configs:
    new_configs = configs[category]
    vb = openpyxl.load_workbook(new_configs["xlsx"])
    sheet = vb["Sheet1"]
    n_row = new_configs["row"]
    for column_name in new_configs["list"]:
        # 分别记录风化和无风化两种数据
        fenghua = []
        unfenghua = []
        new_configs["names"].append(sheet[column_name + "1"].value)
        fenghua_col = new_configs["fenghua_col"]
        for row in range(2, n_row+1):
            if sheet[fenghua_col+str(row)].value == "风化":
                fenghua.append(sheet[column_name + str(row)].value)
            else:
                unfenghua.append(sheet[column_name + str(row)].value)

        new_configs["data"]["风化"].append(fenghua)
        new_configs["data"]["无风化"].append(unfenghua)

targetMap = {}

for boli_type in configs:
    new_configs = configs[boli_type]
    for fenghua in ["风化", "无风化"]:
        for index in range(len(new_configs["list"])):
            component = new_configs["names"][index]
            volume = new_configs["data"][fenghua][index]

            arr = np.array(volume)
```

```
            pvalue = shapiro(arr).pvalue
#             if pvalue < 0.05:
#                 print(boli_type, fenghua, component)
#                 print("非正态分布", pvalue)
#                 print(kstest(arr, cdf = "norm"))
#             if pvalue > 0.05:
            print(boli_type, fenghua, component, "p=%.2f" % pvalue)
            plt.hist(arr)
            plt.show()

expectMap = {
    "铅钡": {},
    "高钾": {}
}

for boli_type in configs:
    new_configs = configs[boli_type]
    for index in range(len(new_configs["list"])):
        component = new_configs["names"][index]
        pvalue = [0, 0]
        mu = [0, 0]
        sigma = [0, 0]
        for i, fenghua in enumerate(["风化", "无风化"]):
            volume = new_configs["data"][fenghua][index]

#             print(boli_type, fenghua, component)
#             print(volume)
            arr = np.array(volume)
            pvalue[i] = shapiro(arr).pvalue
            mu[i] = arr.mean()
            sigma[i] = arr.std()
#             if pvalue[i] > 0.05:
#                 print(boli_type, fenghua, component, "mu = %f, sigma = %f" % (mu,
sigma))
#                 print("正态分布", pvalue[i])
#                 print()
        print(boli_type, component)
        expectMap[boli_type][component] = {
            "type": "normal_dist",
            "风化": {"mu": mu[0], "sigma": sigma[0]},
```

```
            "无风化": {"mu": mu[1], "sigma": sigma[1]},
        }

print(expectMap)

# (type, wenshi, color)
avgMap = {}

# 读取文件
vb = openpyxl.load_workbook("C 位置标注版.xlsx")
sheet = vb["表单 2"]

# 遍历每一行
for row in range(2, 68):
    # 去掉无风化的样本
    if sheet["T"+str(row)].value != "无风化": continue

    # 获取类型、纹饰、颜色
    type_ = sheet["R"+str(row)].value
    wenshi = sheet["Q"+str(row)].value
    color = sheet["S"+str(row)].value
    key = (type_, wenshi, color)

    # 在 avgMap 中创建一个表项用于记录 所属的工艺参数
    if key not in avgMap:
        avgMap[key] = {}

        # 在每个表项里建立化学成分的新表项
        for i in range(ord("B"), ord("O")+1):
            col = chr(i)
            component = sheet[col+"1"].value
            avgMap[key][component] = {
                "sum": 0,
                "n": 0,
                "avg": 0,
                "lst": [],
            }

    # 遍历所有化学成分
    for i in range(ord("B"), ord("O")+1):
```

```
        col = chr(i)
        component = sheet[col+"1"].value
        value = sheet[col+str(row)].value

        avgMap[key][component]["lst"].append(value)
        avgMap[key][component]["sum"] += value
        avgMap[key][component]["n"] += 1

for key in avgMap:
    for component in avgMap[key]:
        avgMap[key][component]["avg"] = avgMap[key][component]["sum"]
            / avgMap[key][component]["n"]
        del avgMap[key][component]["sum"]

component_names = []
for i in range(ord("B"), ord("O")+1):
    col = chr(i)
    component = sheet[col+"1"].value
    component_names.append(component)

print(avgMap)
print(component_names)

wb = openpyxl.Workbook()
wb.remove(wb["Sheet"])
w_sheet = wb.create_sheet(title="风化前预测值")

# 读表头
for i in range(ord("A"), ord("T")+1):
    col = chr(i)
    table_head = sheet[col+"1"].value
    w_sheet[col+"1"].value = table_head

w_sheet["U1"].value = "总值"

write_row = 2 # 记录当前写的位置
for read_row in range(2, 69):
    # 跳过没有风化的
    if sheet["T"+str(read_row)].value != "风化":
        continue
```

```
type_ = sheet["R"+str(read_row)].value
wenshi = sheet["Q"+str(read_row)].value
color = sheet["S"+str(read_row)].value
key = (type_, wenshi, color)

# 枚举表头:各个化学成分
for i in range(ord("A"), ord("T")+1):
    col = chr(i)
    table_head = sheet[col+"1"].value

    if table_head in component_names:
        # 是化学成分
        if table_head in expectMap[type_]:
            # 风化敏感型变量
            mu = expectMap[type_][table_head]["风化"]["mu"]
            sigma = expectMap[type_][table_head]["风化"]["sigma"]
            mu_ = expectMap[type_][table_head]["无风化"]["mu"]
            sigma_ = expectMap[type_][table_head]["无风化"]["sigma"]

            c = sheet[col+str(read_row)].value
            c_ = mu_ + (c - mu) / sigma * sigma_
            expect_val = c_
        else:
            # 风化不敏感型变量，直接取同类的平均

            if key in avgMap:
                # 存在完全相同的未风化类别
                expect_val = avgMap[key][table_head]["avg"]
            else:
                print("apply multi average", key)

                sum_ = 0
                n = 0
                for key_new in avgMap:
                    # 需要类型和颜色与当前样本一致的一组无风化样本
                    if key_new[0] == type_ and key_new[2] == color:
                        avg = avgMap[key_new][table_head]["avg"]
                        sum_ += avg * avgMap[key_new][table_head]["n"]
                        n += avgMap[key_new][table_head]["n"]
```

```
                if n == 0:
                    sum_ = 0
                    n = 0
                    for key_new in avgMap:
                        # 需要类型和纹饰与当前样本一致的一组无风化样本
                        if key_new[0] == type_ and key_new[1] == wenshi:
                            avg = avgMap[key_new][table_head]["avg"]
                            sum_ += avg * avgMap[key_new][table_head]["n"]
                            n += avgMap[key_new][table_head]["n"]

                    expect_val = sum_ / n

            # 如果预测值小于0，就取0
            if expect_val < 0: expect_val = 0

            # 写入预测值
            w_sheet[col+str(write_row)].value = expect_val

        else:
            # 非化学成分项：保持与原表相同即可
            w_sheet[col+str(write_row)].value = sheet[col+str(read_row)].value

    # 将各种化学成分加和
    sum_ = 0
    for i in range(ord("B"), ord("O")+1):
        col = chr(i)
        sum_ += w_sheet[col+str(write_row)].value
    w_sheet["U"+str(write_row)].value = sum_

    write_row += 1

wb.save("风化前预测值.xlsx")
print("已写入文件")
```

问题二相关代码：

决策树代码

```
import csv
```

```
import copy

def loadDataset(filename):
    with open(filename, 'r') as f:
        lines = csv.reader(f)
        data_set = list(lines)

    # 整理数据
    category = data_set[0]
    del (data_set[0])

    for i in range(len(data_set)):
        for j in range(len(data_set[0])):
            data_set[i][j] = float(data_set[i][j])

    return data_set, category

def split_data(data): # 将数据与标签分离

    data_set = copy.deepcopy(data)

    data_mat = []
    label_mat = []
    for i in range(len(data_set)):
        label_mat.append(data_set[i][-1])
        del(data_set[i][-1])
        data_mat.append(data_set[i])

    return data_mat, label_mat


def get_best_split_value(data, result):
    # 找到当前特征下的最佳划分值
    data_set = copy.deepcopy(data)
    result_set = copy.deepcopy(result)
    lst = list(zip(data_set, result_set))
    lst.sort(key=lambda x: x[0])
    length = len(data_set)
    for i in range(length):
        data_set[i] = lst[i][0]
```

```
        result_set[i] = lst[i][1]

    if length == 1:
        return float("Inf"), float("Inf")

    split_value = []

    for i in range(length-1):
        split_value.append((data_set[i+1] + data_set[i]) / 2)

    gini = []

    for i in range(length - 1):
        gini.append(get_gini(i, result_set))

    min_gini = 0

    for i in range(len(gini)):
        if gini[i] < gini[min_gini]:
            min_gini = i

    return split_value[min_gini], gini[min_gini]

def get_gini(split_value_position, label_sorted):
    # 求出当前划分下的gini_index
    k_count = p_count = 0
    for i in range(split_value_position+1):
        if label_sorted[i] == 0:
            k_count += 1
        else:
            p_count += 1
    gini_1 = 1 - ((k_count / (k_count + p_count)) ** 2 + (p_count / (k_count + p_count))
** 2)

    i = split_value_position + 1
    k_count = p_count = 0
    while i < len(label_sorted):
        if label_sorted[i] == 0:
            k_count += 1
        else:
```

```
            p_count += 1
        i += 1
    gini_2 = 1 - ((k_count / (k_count + p_count)) ** 2 + (p_count / (k_count + p_count))
** 2)

    gini_gain = (split_value_position+1)/len(label_sorted) * gini_1 + (1 -
(split_value_position+1)/len(label_sorted)) * gini_2

    return gini_gain

def get_best_feature(data, category):
    # 找到gini_index最小值的特征
    length = len(category)-1

    data_set, result = split_data(data)

    feature_gini = []

    split_feature_value = []

    feature_values = []

    for i in range(length):
        feature_gini.append(0)
        split_feature_value.append(0)

        feature_values.append([])
        for j in range(len(data_set)):
            feature_values[i].append(data_set[j][i])

    for i in range(length):
         split_feature_value[i], feature_gini[i] =
get_best_split_value(feature_values[i], result)

    feature_num = 0
    # 找到最小gini_gain对应的feature编号
    for i in range(length):
        if feature_gini[i] < feature_gini[feature_num]:
            feature_num = i
```

```
    return feature_num, split_feature_value[feature_num]

class Node(object):
    def __init__(self, category, item):
        self.name = category
        self.elem = item
        self.lchild = None
        self.rchild = None

def leaf_value(data): # 返回叶子节点值
    sum = 0
    for i in range(len(data)):
        sum += data[i][-1]

    return sum/len(data)

def creat_tree(data, labels, feature_labels=[]):
    # 递归生成决策树
    # 结束条件
    if len(labels) == 1:
        return Node('result', leaf_value(data))
    if abs(leaf_value(data) - 1) < 1e-5:
        return Node('result', leaf_value(data))
    if abs(leaf_value(data)) < 1e-5:
        return Node('result', leaf_value(data))


    # 最优特征的标签
    best_feature_num, best_feature_value = get_best_feature(data, labels)

    feature_labels.append(labels[best_feature_num])

    node = Node(labels[best_feature_num], best_feature_value)

    ldata = []
    rdata = []
    i = 0
    for d in data:
        if d[best_feature_num] <= best_feature_value:
            ldata.append(d)
```

```
        else:
            rdata.append(d)
        del (d[best_feature_num])
        i+=1

    labels2 = copy.deepcopy(labels)
    del(labels2[best_feature_num])

    tree = node
    tree.lchild = creat_tree(ldata, labels2, feature_labels)
    tree.rchild = creat_tree(rdata, labels2, feature_labels)

    return tree

def breadth_travel(tree):
    # 广度遍历
    queue = [tree]
    while queue:
        cur_node = queue.pop(0)
        print(cur_node.name, end=" ")
        print(cur_node.elem, end=" ")
        if cur_node.lchild is not None:
            print('my lchild is', cur_node.lchild.name, cur_node.lchild.elem, end=' ')
            queue.append(cur_node.lchild)
        if cur_node.rchild is not None:
            print('my rchild is', cur_node.rchild.name, cur_node.rchild.elem, end=' ')
            queue.append(cur_node.rchild)
        print()


if __name__ == "__main__":
    # 装入数据
    train_set, category = loadDataset('决策树训练集.csv')
    # 生成树
    my_tree = creat_tree(train_set, category)
    # 打印结果
    breadth_travel(my_tree)
```

聚类分析

```
import copy
```

```
import numpy as np
import math
import csv
from random import random
import openpyxl
import matplotlib.pyplot as plt

def loadDataset(filename): # 读入数据
    with open(filename, 'r') as f:
        lines = csv.reader(f)
        data_set = list(lines)

    # 整理数据
    feature = data_set[0]
    del (data_set[0])

    for i in range(len(data_set)):
        for j in range(len(data_set[0])):
            data_set[i][j] = float(data_set[i][j])

    return data_set, feature

def get_dist(p1, p2): # 计算点间的欧氏距离
    dist_ = 0
    for i in range(len(p1)):
        dist_ += (p1[i] - p2[i]) ** 2
    return math.sqrt(dist_)

def nearest(cur_point, cluster_centers): # 找到距离当前节点最近的聚类中心点
    min_dist = np.inf
    m = np.shape(cluster_centers)[0]  # 当前已经初始化的聚类中心的个数
    for i in range(m):
        d = get_dist(cur_point, cluster_centers[i])  # 计算 cur_point 与每个聚类中心之间
的距离
        if min_dist > d:  # 选择最短距离
            min_dist = d
    return min_dist

def get_centroids(data, k): # K-means++算法，产生初始的聚类中心
    data_set = copy.deepcopy(data)
```

```
    m = len(data_set)  # 行数 即数据点的个数
    n = len(data_set[0])  #列数 即特征的个数

    cluster_centers = []  # 初始化聚类中心
    for i in range(k):
        cluster_centers.append([])

    index = np.random.randint(0, m)  # 1、随机选择一个样本点为第一个聚类中心
    cluster_centers[0] = copy.deepcopy(data_set[index])

    d = [0.0 for _ in range(m)]  # 2、初始化一个距离的序列
    for i in range(1, k): # 外层是更新 cluster_centers 的值
        sum_all = 0
        for j in range(m):
            d[j] = nearest(data_set[j], cluster_centers[0:i])  # 3、对每一个样本找到最近的聚类中心点 并返回最近距离值
            sum_all += d[j]  # 4、将所有的最短距离相加
        sum_all *= random()  # 5、取得 sum_all 之间的随机值
        for j, di in enumerate(d):  # 6、获得距离最远的样本点作为聚类中心点
            sum_all -= di
            if sum_all > 0:
                continue
            cluster_centers[i] = copy.deepcopy(data_set[j])
            break
    return cluster_centers

def generate_clusters(k, data, cluster_centers, n): # k-means 算法 更新迭代聚类中心
    flag = 1
    while flag:
        flag = 0
        cluster = []
        for i in range(k):
            cluster.append([])
        for i in range(len(data)):
            d = float('inf')
            for j in range(k):
                d_j = get_dist(data[i], cluster_centers[j])
                if d > d_j:
                    d = d_j
                    center_num = j
```

```
            cluster[center_num].append(i)
        for i in range(k):
            for j in range(len(cluster[i])):
                if j == 0:
                    new_center = copy.deepcopy(data[cluster[i][0]])
                    continue
                for q in range(n):
                    new_center[q] += data[cluster[i][j]][q]
            for p in range(len(new_center)):
                new_center[p] /= len(cluster[i])
            if new_center != cluster_centers[i]:
                cluster_centers[i] = new_center
                flag = 1

    return cluster_centers, cluster

def get_sse(k, cluster_centers, clusters, data): # 计算聚类划分评价值和方差SSE
    sum_see = 0
    for i in range(k):
        for x in clusters[i]:
            sum_see += get_dist(data[x], cluster_centers[i]) ** 2
    return sum_see

def get_Silhouette_Coefficient(k, cluster, data): # 计算聚类划分评价值轮廓系数
    if k == 1:
        return 0
    else:
        sum_s_co = np.zeros(len(data))
        for i in range(len(data)):
            a = 0

            for cluster_instance in cluster:
                if i in cluster_instance:
                    cluster_instance_i = cluster_instance
                    break
            for j in cluster_instance_i:
                 a += get_dist(data[i], data[j])
            a /= len(cluster_instance_i)
            b_min = float('inf')
            for cluster_instance in cluster:
```

```
                b = 0
                if cluster_instance != cluster_instance_i:
                    for j in cluster_instance:
                        b += get_dist(data[i], data[j])
                    b /= len(cluster_instance)

                    if b < b_min:
                        b_min = b
            sum_s_co[i] = (b_min - a) / max(b_min, a)
        return sum_s_co.mean()

def get_similarity(base_cluster, cur_cluster): # 计算模型稳定行和敏感性指标 划分相似度
    mean_similarity = 0
    for c in cur_cluster:
        jaccard = 0
        for b in base_cluster:
            c = set(c)
            b = set(b)
            tmp = len(c&b)/len(c|b)
            if tmp > jaccard:
                jaccard = tmp
        mean_similarity += jaccard
    return mean_similarity/4

if __name__ == '__main__':

    # 仅以计算相邻两次生成的聚类相似度为例，展示函数的调用关系 其他 main 函数不再赘述

    data_set, feature_list = loadDataset('k-means-PbBa.csv')
    wb = openpyxl.Workbook()
    wb.remove(wb["Sheet"])
    w_sheet = wb.create_sheet(title="相似度")
    base_cluster = [[5, 6],
                    [1, 14, 15],
                    [0, 2, 3, 4, 11, 12, 13],
                    [7, 8, 9, 10, 16, 17]]
    for k in range(1000):
        cluster_centers = get_centroids(data_set, 4)
        final_cluster_center, final_cluster = generate_clusters(4, data_set,
cluster_centers, len(data_set[0]))
```

```
        w_sheet.cell(row=k+1, column=1).value = get_similarity(base_cluster,
final_cluster)
        base_cluster = copy.deepcopy(final_cluster)
    wb.save('PbBa-1000次实验-与相邻基准的相似度.xlsx')
```

问题四相关代码：

斯皮尔曼相关系数计算与可视化

```
# coding=utf-8
import pandas as pd
import matplotlib.pyplot as plt  # 可视化
import seaborn as sns  # 可视化
import scipy

df = pd.read_csv('T4-PbBa.csv')  # 读取数据

_, ax = plt.subplots(figsize=(12, 10))

corr = df.corr(method='spearman')
cmap = sns.diverging_palette(220, 10, as_cmap=True)
_ = sns.heatmap(
    corr,
    cmap=cmap,
    square=True,
    cbar_kws={'shrink': .9},
    ax=ax,
    annot=True,
    annot_kws={'fontsize': 9})
plt.savefig('PbBa-spearman.png')
plt.show()
```