基于数理统计与机器学习的玻璃制品成分分析及鉴别

摘　要

玻璃的传入与本土制造最早可以追溯到汉代，铅钡玻璃和高钾玻璃是古时推广范围较广的两种玻璃，长时间的环境影响会使玻璃制品被不同程度地风化，本文基于数理统计和以二分类算法为代表的机器学习，针对古代玻璃文物的成分分析与鉴别问题进行建模求解。

首先，进行**数据预处理**。先将纹饰、类型、颜色全部转化为等级数值变量，将表面风化转化为逻辑数值变量；对化学成分比例进行累加后发现有 15、17 编号采样点的化学成分百分比累加和位于 85%~105%的区间之外，去掉该两组数据；将每个采样点化学成分百分比的成分性不足量均分给含量百分比为空的化学成分，以填补缺失值。

对于问题 1，表面风化和自身物理属性间的关系分析分为相关关系分析和差异关系分析，分析相关关系时采用 **Spearman 相关性分析**，分析差异关系时采用**交叉卡方分析**；分析表面风化与其各类化学成分含量的统计规律时，对数据进行正态检验和方差齐性检验后，采用 **Pearson 相关性**分析检验共线性，采用**单因素 ANOVA 分析**筛除影响意义不显著的化学成分，利用 **Logistic 回归**和 **LDA 线性判别分析**进行二分类规律的探索，两种算法的正确率分别为 **80.6%**和 **82.1%**，仅相差 1.5%，两种算法发现金属氧化物比非金属氧化物对表面风化的影响更显著，其中氧化铁的影响最显著；预测风化点风化前的化学成分含量时，利用已得到的统计规律，以**定比例步长的逐步搜索**形式，搜索突变为无风化时各化学成分的含量比例。

对于问题 2，在进行与问题 1 步骤一致的数据清洗和筛查之后，以采集点化学成分比例和是否风化作为输入，玻璃类型作为输出，并且以训练集:测试集为 8:2 的数据比例训练 4 种机器学习分类算法，包括**朴素贝叶斯（NB）、K 近邻 (KNN)、多层感知器神经网络（MLP）**及**随机森林（RF）**算法，用于玻璃类型分类并统计其分类规律，4 种分类算法的正确率都为 **100%**；通过熵权法分别选取最适合铅钡和高钾玻璃亚分类的化学成分，并分别以提取的化学成分含量归一化值为纵坐标，选取 3 个效果较好的机器学习算法 NB、MLP、RF 的类别判定概率均值作为横坐标进行无监督学习，通过 **K 均值（K-means）聚类**进行亚分类；上下波动亚分类划分标准的 10%，进行合理性和敏感性检验，模型通过检验。

对于问题 3，利用问题 2 训练后的朴素贝叶斯、多层感知器神经网络、随机森林分类模型，分别对附件表单 3 的样本进行玻璃类型的预测，3 个模型算法的**预测结果一致**，可信度高；进行敏感性检验时，3 个模型算法都能通过在采样点的各个化学成分含量随机波动 10%的预测结果不变检验，贝叶斯和多层感知神经网络能通过随机波动 30%的预测结果不变检验。

对于问题 4，将全部数据划分成铅钡玻璃和高钾玻璃共 2 组，检验两组样本满足正态分布和方差齐性之后，对 2 组玻璃文物样品分别采用 Pearson 相关性分析，铅钡玻璃化学成分间负关联度较高，高钾玻璃化学成分间正关联度较高；将两组相关系数集的系数一一配对，进行铅钡玻璃样品和高钾玻璃样品化学成分相关系数的**配对样本 t 检验**，两组之间呈现出 0.01 水平的显著性，即两者化学成分关联关系存在显著差异性，Cohen's d 值为 0.456，说明差异幅度接近中等水平。

关键词：成分分析及鉴别　数理统计　机器学习　二分类算法

一、 问题重述

1.1 问题背景

玻璃是古代丝绸之路贸易往来的宝贵物证。早期玻璃的来源有两种，一是西亚、埃及等国外地区传入，二是我国学习其技术后在本土取材制作。虽然两种玻璃制品的外观相似，但可以从微观化学成分上判断玻璃的来源。

玻璃原料石英砂内含有的二氧化硅（SiO_2）是玻璃的主要化学成成分。除此之外，炼制时还需要加入稳定剂石灰石和助熔剂，助熔剂是判断玻璃类型和来源的依据之一。例如，铅钡玻璃被认为是我国自己发明的玻璃类型，其助熔剂是铅矿石，因此其氧化铅（PbO）、氧化钡（BaO）的含量较高；流行于我国岭南地区以及东南亚和印度的钾玻璃，其助熔剂是草木灰，因此其含钾量高。

除了烧制过程外，古代玻璃的化学成分还会因外部环境的影响而变化。古代玻璃在受到埋藏环境的影响风化的过程中，玻璃内部元素与环境元素进行了大量交换，玻璃成分比例也会大幅变化，导致现在对其类别的判断对产生误差。

1.2 问题的提出

现有一批我国古代玻璃制品的相关数据，考古工作者已根据文物样品的化学成分和其他检测手段将其分为高钾玻璃和铅钡玻璃两种类型。附件表单 1 中给出了这些样品文物的分类信息，附件表单 2 给出了对应的主要成分所占比例。

建立数学模型，根据附件中的数据研究解决下列问题：

（1） 分析玻璃文物表面分化与其玻璃类型、纹饰和颜色之间的关系；再结合玻璃的类型，分析文物样品表面有无风化化学成分含量的统计的规律，并根据表单风化点检测数据，预测其风化前的化学成分含量。

（2） 根据附件数据，分析两种玻璃的分类规律；再对于每个类别选择合适的化学成分进行亚类划分，得出划分方法和结果后，对结果的合理性和敏感性进行分析。

（3） 分析附件表单 3 未知类别玻璃文物的化学成分，鉴别其玻璃类型并对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

2.1 问题一的分析

根据本问的要求，可以分成三个小问，分别建立模型进行求解。对于玻璃文物风化与否和自身属性关系的分析，采用 Spearman 相关系数和交叉卡方分析进行相关性和差异性分析。

对于文物表面风化与否和化学成分含量关系的分析，数据集进行了正态检验、方差齐性检验和 Pearson 相关性分析以考察数据分布和自变量之间的关联性，又利用单因素 ANOVA 分析检验成分含量差异显著性，筛选显著自变量。

数据清洗完毕后，选择 Logistic 回归和 LDA 线性判别分析两种二分类算法，分析两组玻璃文物的表面风化与否和化学成分含量的统计规律，并比较两种算法的优劣。

对于风化点于风化前化学成分含量的预测，需要以第二问分类算法为基础，采用迭代风化点的自变量值（化学成分含量），逐步搜索的方式，直到至风化点的因变量值（风化与否）变为 0（无风化），由此得到风化前各化学成分含量的预测值。

2.2 问题二的分析

根据附件表单 1 和表单 2 的数据先分析两种玻璃的分类玻璃，再进行亚类划分，并对划分方案进行相关分析。首先，分别将合并后的表单 1 和表单 2 数据按玻璃类型进行划分，得到两个分开的数据集；接着，我们选择并比较分析了 *k* 近邻算法（KNN）、朴素贝叶斯分类器（Naive Bayes Classifiers）和多层感知机（MLP）、随机森林（RF）这四种有监督学习分类算法处理玻璃分类问题的效果。本文中四种分类算法都是划分训练集以建立模型，再对测试集进行预测，交叉验证得到结果。分析预测结果和算法效果，选择多个算法结果求平均概率，作为最终的玻璃分类规律。

对于亚类划分问题，我们使用分类后的数据集使用熵权法，对各化学成分变量的变异性进行分析。选择出一组变异性大的化学成分含量比例归一化后作为 y 轴，将类型概率作为 x 轴，进行 K-means 聚类，算出归一化前实际 y 后，再结合实际化学元素间的关系划分亚类间的阈值。

2.3 问题三的分析

本问需要在问题二玻璃分类模型的基础上，运用该模型对附件表单 3 中的未知类别玻璃文物的化学成分进行分析，得到鉴别结果。需要注意的是，由于我们的模型是建立在原数据处理后的基础之上的，因此在进行分析鉴别之前，我们也需要对表单 3 的数据进行处理，处理之后再使用分类模型进行求解。

2.4 问题四的分析

本问要求根据附件表单 1 和表单 2 的数据，根据高钾玻璃与铅钡玻璃两种类型玻璃的文物样本，分别研究同类玻璃样品下，各化学成分直接的关联关系；在得到化学成分的关联关系后，分析并检验两种类型玻璃文物样品之间化学成分关联关系的差异性。将按玻璃类型将数据集分为两类，分别采用 *Pearson* 相关性分析，计算每组化学成分之间的相关系数，得到化学成分之间的关联关系。对于差异性分析，先将两组相关系数集的相关系数一一配对，使用配对样本 *t* 检测不同玻璃类型间化学成分关联关系的差异性。

三、 模型假设与符号说明

3.1 模型假设

1. 各个附件的数据真实可靠，测量有效的化学成分含量准确无误。
2. 未分类玻璃文物属于高钾玻璃和铅钡玻璃中的一类。
3. 不考虑外界因素如玻璃文物保存环境湿度、距今年份对数据的影响。

3.2 符号说明

符号	说明	单位
d_i	*Spearman* 相关性分析等级差	\
r_i	*Spearman* 相关性系数	\
A	交叉卡方实际值	\
T	交叉卡方理论值	\
X^2	衡量理论与实际的差异程度	\
N	总个数数量	\
x_i	一组数据中第 i 个数据	\

$\bar{x}$	数据平均值	\
S^2	方差	\
W	检验统计量	\
$E(x_i)$	第 i 个变量期望值	\
F	样本标准差	\
SST	总离差平方和	\
SSE	组内平方和	\
SSA	组间平方和	\

四、 数据预处理

4.1 数据整理与删除

在分析数据之前，我们先对现有数据进行了整理。我们使用软件 *Excel* 对附件表单 2 的进行整理，逐行累加并统计了含量比例为 0 的化学成分的个数。

经过整理，我们发现编号 15 和编号 17 的文物的成分比例累加和分别为 79.47%和 71.89%。据题意这两份数据均属于无效数据，应当从数据集中剔除，防止其作为异常数据干扰后续计算。

4.2 缺失值处理

剔除无效数据后，我们还对数据集中的缺失值进行了处理。这里的缺失值分为两类。一类是数据集本身有部分数据缺失，另一类是为方便后续计算将本身数值为 0 的数据记作缺失值。

第一类的缺失值，即附件表单 1 中的颜色列有数据缺失。在分析玻璃文物表面分化与其玻璃类型、纹饰和颜色之间的关系时，由于数据量较少，根据其他数据直接填充可能会对分析结果造成较大误差，因此在该计算过程中直接忽略该行数据。

第二类缺失值，即附件表单 2 中未检测到的成分。原数据集中存在大量值为 0 的数据，为了便于后续计算，我们使用 *Python* 编写算法处理，处理公式如（1），最终得到缺值分配值，以填充原数据集。

$$\text{缺值分配值} = \frac{100 - \text{累加和}}{\text{缺失值个数}} \tag{1}$$

部分处理后数据如下表所示，完整数据集见支撑材料“数据处理补空值去无效合并文物编号类型.xlsx”。

表 1：对附件表单 1、表单 2 数据的处理、合并后结果

文物采样点	SiO2	Na2O	K2O	CaO	MgO	Al2O3	…	文物编号	类型
01	69.33	0.48	9.99	6.32	0.87	3.93	…	01	高钾
02	36.28	0.028	1.05	2.34	1.18	5.73	…	02	铅钡
03 部位 1	87.05	0	5.19	2.01	0	4.06	…	03	高钾
03 部位 2	61.71	0.37	12.37	5.87	1.11	5.5	…	03	高钾

五、 问题一的模型建立与求解

本题首先要求，根据附件表单 1 给定的玻璃文物的基本信息，对玻璃文物表面风化和自身纹饰、类型、颜色间的关系进行分析。在问题求解过程中，建立玻璃文物风

化与否和自身属性关系的统计分析模型，进行相关关系和差异关系的分析。

本题同时要求，根据附件表单 1 和表单 2 给定的基本信息和已分类玻璃文物的化学成分比例，对铅钡玻璃和高钾玻璃分别进行讨论，探究分析玻璃文物样品表面的风化与否跟其各类化学成分含量的统计规律。在问题求解时，建立文物表面风化与否和化学成分含量关系的统计分析模型，探究其统计规律。

本题最后要求，根据表单 2 中风化点的测量数据，基于前面的求解，预测这些风化点在风化之前的化学成分含量。在问题求解时，建立探究风化点于风化前化学成分含量的预测模型，对表单 2 风化点在风化之前的化学含量进行预测。

5.1 玻璃文物风化与否和自身属性关系的统计分析模型

本模型旨在根据附件表单 1 的数据，解决玻璃文物表面风化和自身物理属性间的关系分析。在该模型中，对关系的分析从两方面着手：相关关系和差异关系。其中，相关关系的分析采用 *Spearman* 相关性分析，差异关系的分析采用交叉卡方分析。

5.1.1 模型建立

一般而言，数据之间的关系可以分为 3 种：相关关系、差异关系、其他关系。在这里，我们主要从相关关系和差异关系挖掘分析玻璃文物表面风化和自身物理属性间的关系。

相关关系指因变量与自变量或者自变量与自变量之间存在着非严格的依存关系，常见的相关关系研究方法有 Pearson 相关性分析、Spearman 相关性分析以及一系列回归算法等。由于附件表单 1 提供的数据中，自变量和因变量全部属于定性变量，只有 Spearman 相关性分析可以适用自变量和因变量都是定性变量的场景，例如 Pearson 相关性分析的其他方法，要求变量为定量变量且存在线性关系。因此，对于玻璃文物表面风化和自身物理属性间相关关系的分析，模型采用 Spearman 相关性分析，其过程如下。

先对因变量和各个自变量生成虚拟等级变量，将字符信息转换为数值信息。自变量和各个因变量对应的虚拟等级变量的等级划分如表 2 所示。

表 2：虚拟变量的等级划分表

纹饰	A	0.333333333
	B	0.666666667
	C	1
类型	高钾玻璃	0
	铅钡玻璃	1
颜色	null	0
	浅绿	0.125
	浅蓝	0.25
	绿	0.375
	蓝绿	0.5
	深绿	0.625
	深蓝	0.75
	紫	0.875
	黑	1
表面风化	否	0
	是	1

注：颜色由浅至深（RGB 三个通道的数值由大至小）对应为等级值由小到大。

定义 X、Y 两组数据，其 Spearman（等级）相关系数的计算公式如下：

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{2}$$

其中，d_i 为 X_i 和 Y_i 之间的等级差。

在这里，利用 MATLAB 的 *coeff = corr(X, Y, 'type', 'Spearman')*函数计算 *Spearman* 相关系数。

在对玻璃文物表面风化和自身物理属性进行相关关系分析后，接下来进行差异关系分析。

差异关系指不同样本组的某个指标在样本组间存在的差别关系。常见的差异研究的方法主要基于显著性假设检验，分为参数检验和非参数检验两大类，由于附件表单 1 提供的数据中，自变量和因变量全部属于定性变量，且离散型变量不满足正态分布和方差齐性条件，不能进行参数检验，只能进行非参数检验。非参数检验中的卡方检验，对于自变量和因变量都属于定性变量的数据，具有很好的显著性差异检验分析效果。因此对于玻璃文物表面风化和自身物理属性间差异关系的分析，模型采用交叉卡方分析，其过程如下：

$$x^2 = \sum \frac{(A-T)^2}{T} \tag{3}$$

其中，*A* 为实际值，*T* 为理论值。X^2 值的意义:衡量理论与实际的差异程度。

最后，利用 *SPSSAU* 软件进行交叉卡方分析的计算。

5.1.2 模型求解

经过计算，*Spearman* 相关系数矩阵色阶热力图如图 1 所示。

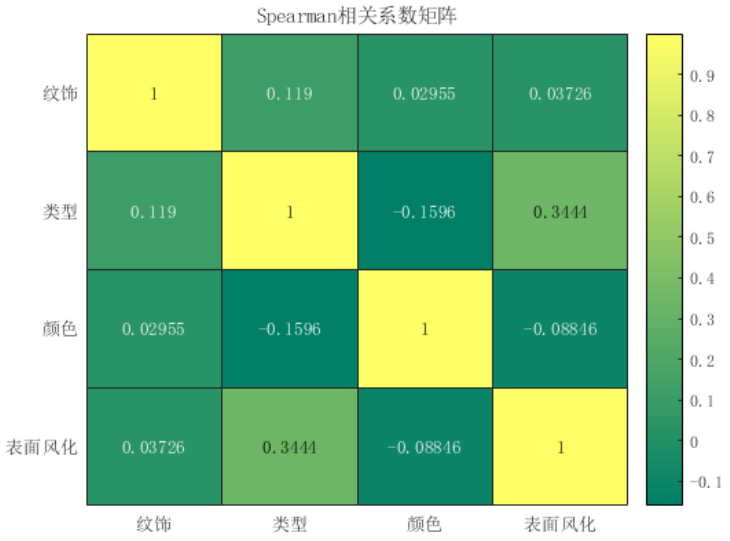

图 1：*Spearman* 相关系数矩阵色阶热力图

根据系数矩阵的第 4 行或者第 4 列，可以看出，玻璃文物表面风化与否和其类型的相关性最大，相关系数绝对值为 0.3444，说明由不同主要化学成分决定的铅钡玻璃和高钾玻璃这两种类型的玻璃抗风化能力不同。与此同时，玻璃文物表面 fenghua 与否和颜色的相关性非常小，为-0.08846。而由系数数值和色阶热力图可以看出，玻璃文物表面风化与否和其纹饰的相关性最小，为 0.03726，由于不知道 3 种纹饰的具体结构，在这里，仅根据相关系数推测，三种纹饰的物理结构对化学成分抗风化的影响一

致性较高。

通过 *SPSSAU*，玻璃文物表面风化和自身物理属性间差异关系的交叉卡方分析结果如表 3 所示。

表 3：交叉(卡方)分析结果

变量	表征	表面风化		总计	χ2	p
		无风化	风化			
纹饰	A	45.83%	32.35%	37.93%	4.957	0.084
	B	0.00%	17.65%	10.34%		
	C	54.17%	50.00%	51.72%		
总计		24	34	58		
类型	铅钡	50.00%	82.35%	68.97%	6.880	0.009**
	高钾	50.00%	17.65%	31.03%		
总计		24	34	58		
颜色	浅绿	8.33%	3.33%	5.56%	6.287	0.507
	浅蓝	33.33%	40.00%	37.04%		
	深绿	12.50%	13.33%	12.96%		
	深蓝	8.33%	0.00%	3.70%		
	紫	8.33%	6.67%	7.41%		
	绿	4.17%	0.00%	1.85%		
	蓝绿	25.00%	30.00%	27.78%		
	黑	0.00%	6.67%	3.70%		
总计		24	30	54		

* p<0.05　　** p<0.01

从上表可知，利用卡方检验（交叉分析）去研究表面风化与否对于纹饰、 类型、颜色这 3 项自身物理属性的差异关系，可以看出，纹饰和颜色属性变量对应的的 p 值大于 0.05，说明纹饰、颜色对于玻璃文物表面是否风化不存在显著性差异，意味着纹饰和玻璃对于表面是否风化表现出一致性，并没有差异性。而玻璃类型属性变量对应的 p 值不仅小于 0.05，甚至小于 0.01，说明玻璃类型对于玻璃文物表面是否风化存在显著性差异，意味着玻璃类型对于对于表面是否风化表现出很大的差异性。

表面风化和纹饰的交叉图如图 2 所示，表面风化和颜色的交叉图如图 3 所示，表面风化和玻璃类型的交叉图如图 4 所示。

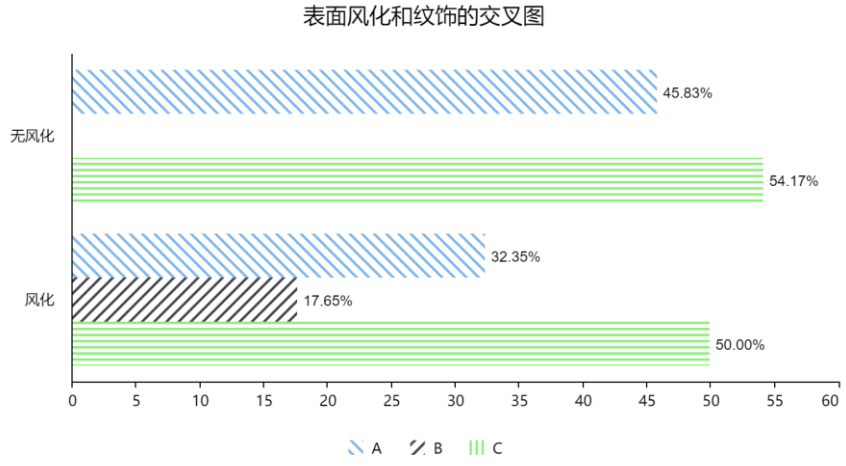

图 2：表面风化和纹饰的交叉图

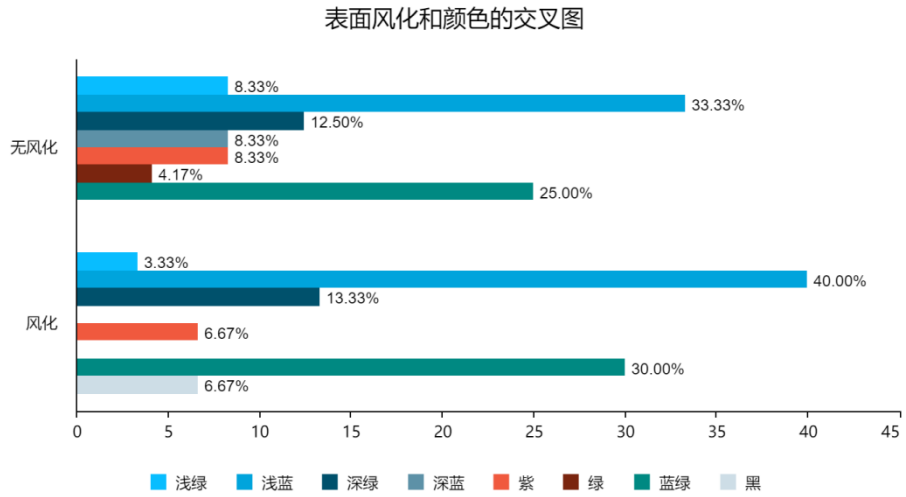

图 3：表面风化和颜色的交叉图

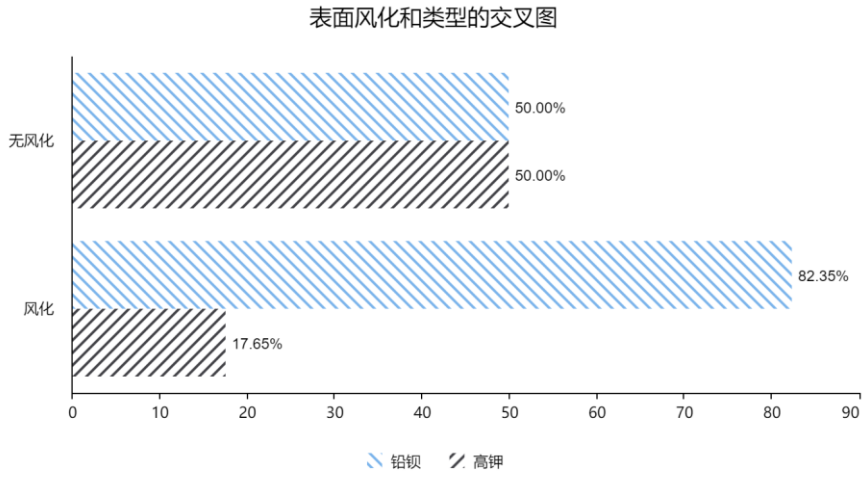

图 4：表面风化和类型的交叉图

玻璃类型对于表面风化呈现出 0.01 水平显著性（chi=6.880, p=0.009<0.01）差异影响。通过百分比对比差异可知，铅钡玻璃被风化的概率比例高达 82.35%，明显高于铅钡玻璃无风化的概率比例；高钾玻璃无风化的概率比例仅为 17.65%，明显低于被风化概率比例。结合化学元素周期规律，元素铅和钡的活跃性小于钾，因此金属铅和钡的氧化性小于钾，从而铅和钡的还原性大于钾，造成金属铅的氧化物和钡的氧化物更不稳定，证明了铅钡玻璃的抗风化能力比高钾玻璃的抗风化能力弱。

5.2 文物表面风化与否和化学成分含量关系的统计分析模型

在本模型中，控制文物表面的风化与否为唯一因变量，在对样本数据分别进行正态检验和方差齐性检验后，利用 *Pearson* 相关性分析进行成分含量的共线性探索，单因素 ANOVA 分析检验成分含量差异显著性，筛选显著影响自变量即显著影响成分含量，用于作接下来二分类模型的自变量，至此完成数据筛选。选择 *Logistic* 回归和 *LDA* 线性判别分析两种常见可用于二分类的算法，分析两组玻璃文物的表面风化与否和化学成分含量的统计规律，并比较两种算法的优劣。

5.2.1 模型建立

首先，在样本数据中控制采样点风化与否为唯一因变量，各化学成分含量百分比和玻璃类型为各自变量。值得注意的是，在对表单 2 的数据进行因变量标签标注时，

需要先将表单 1 中各文物编号对应的表面风化变量值移植到表单 2 中文物采样点相同前缀编号对应的表面风化变量，进行赋值；对应赋值结束后，可以观察到部分文物采样点变量值中提示了采样点是否风化，若此处的提示与上一步赋值的表面风化变量值不一致，则以提示为准，修改该样本的表面风化标签变量值。例如，表单 2 中文物采样点为“49 未风化点”的样本点位于表单 1 中文物编号为“49”的玻璃文物，后者表面风化为“风化”，则将前者表面风化先按对应准则标注为“风化”；之后，根据表单 2 中文物采样点为“49 未风化点”的“未风化点”提示，将上一步赋值的“风化”修改为“无风化”，其它情景以此类推。

由于玻璃属于混合物，表单 2 的化学成分种类较多，在对玻璃进行 Logistic 回归二分类和 LDA 线性判别分析二分类之前，可以先对数据进行筛选，选取显著性影响较大的自变量及其数值。由于化学成分含量属于定量连续数据，数据的筛选步骤，可以由单因素 ANOVA 分析完成，在这之前可以用 Pearson 相关分析先检验一下共线性。但由于进行两种参数检验方法之前，需要检验数据是否呈正态分布以及方差齐性，在这里，我们对样本数据分别进行正态检验和方差齐性检验。

A.正态检验与方差齐性检验

首先进行正态检验，该检验通过计算得到一个相关系数 W，相关系数越接近 1，表明数据和正态分布拟合得越好，检验步骤如表 4。

表 4：正态分布检验步骤

正态检验步骤
Step 1：将两组数据集中的数据分别按数值大小升序排列，使$x_1 \leqslant x_2 \leqslant ... \leqslant x_n$；
Step 2：求样本平均值；
Step 3：根据式（4），计算检验统计量W；
Step 4：给定显著性水平α；
Step 5：查表得到判断临界值W_α；
Step 6：若$W > W_\alpha$，按计算的显著性水平 α 舍弃正态性假设；若$W > W_\alpha$，接受正态性假设；
Output：样本数据是否服从正态分布。

注：使用单侧检验。

接下来进行方差齐性检验，分析统计模型中的全部或一部分参数是否适合估计总体。计算公式如下，检验步骤如表 5。

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{4}$$

其中，S^2为方差，即样本标准差的平方。

$$F = \frac{S_1^2}{S_2^2} \tag{5}$$

其中，S_1^2 为第一部分独立样本的方差，S_2^2 为第二部分独立样本的方差。

表 5：方差齐性检验步骤

方差齐性检验步骤
Step 1：记两独立总体，对两独立总体分别进行随机抽样；
Step 2：根据式（4）计算第一部分、第二部分独立样本的方差；
Step 3：根据式（5）计算检验统计量 F；
Step 4：给定显著性水平α；

Step 5：查表得到判断临界值F_α；

Step 6：若$W<W_\alpha$，拒绝两总体有显著差异的假设；若$W>W_\alpha$，接受有显著差异的假设；

Output：是否有显著差异。

注：使用单侧检验。

B.Pearson 相关性分析与单因素 ANOVA 分析

Pearson 相关系数又称 *Pearson* 积矩相关系数，用于度量两个变量之间的相关性，其值介于-1 与 1 之间。*Pearson* 相关系数的计算步骤与公式如下：

Step 1：根据式（6），求出变量 1 的期望与方差；

$$E(x_1)=\frac{\sum_{i=1}^{n}x_1(i)}{n},\quad \sigma_{x_1}=\sqrt{\frac{\sum_{i=1}^{n}(x_1(i)-E(x_1))^2}{n}} \tag{6}$$

Step 2：根据式（7），求出变量 2 的期望与方差；

$$E(x_2)=\frac{\sum_{i=1}^{n}x_2(i)}{n},\quad \sigma_{x_2}=\sqrt{\frac{\sum_{i=1}^{n}(x_2(i)-E(x_2))^2}{n}} \tag{7}$$

Step 3：根据式（8），求出变量 1 与变量 2 的协方差；

$$Cov(x_1,x_2)=\frac{\sum_{i=1}^{n}(x_1(i)-E(x_1))(x_2(i)-E(x_2))}{n} \tag{8}$$

Step 4：根据式（9），求出皮尔逊相关系数。

$$\rho_{x_1,x_2}=\frac{Cov(x_1,x_2)}{\sigma_{x_1}\sigma_{x_2}} \tag{9}$$

由于数据样本量较小，而自变量化学成分则有 14 个，我们借助 *SPSS 26* 软件进行单因素 ANOVA 分析，考察所有化学成分含量百分比与因变量之间的关系，即判断各自变量对有无风化是否存在显著差异，筛选存在显著性差异的自变量，用于后面的 *Logistic* 回归和 LDA 线性判别分析二分类，保证结果更加可靠。

单因素 ANOVA 分析的计算步骤与公式如下所示：

Step 1：提出原假设和备择假设；

H_0: 均值是否相等

H_1: 均值不完全相等

Step 2：计算离差平方和；

Step2.1：总离差平方和（*SST*），自由度为 *n-1*；

$$SST=\sum\sum(x_{ij}-\bar{x})^2 \tag{10}$$

Step2.2：组内平方和（*SSE*），自由度为 *n-m*；

$$SSE=\sum_{j}\sum_{i}(x_{ij}-\bar{x}_j)^2,\ \text{平均平方}\ MSE=\frac{SSE}{n-m} \tag{11}$$

Step2.3：组间平方和（*SSA*），自由度为 *n-m*；

$$SSA=\sum_{m}n_j\cdot(\bar{x}_j-\bar{x})^2,\ \text{平均平方}\ MSA=\frac{SSA}{m-1} \tag{12}$$

Step2.4：*SST=SSE+SSA*；

Step 3：计算统计检验量 *F*；

$$F = \frac{MSA}{MSE} \tag{13}$$

Step 4：给定显著性水平 α；

Step 5：查表得到判断临界值 F_α；

Step 6：若 $F > F_\alpha$，按计算的显著性水平 α 拒绝原假设，存在显著差异；若 $F < F_\alpha$，接受原假设，不存在显著差异。（注：使用单侧检验）

C.二分类算法：Logistic 回归二分类/LDA 线性判别分析

对定性因变量（表面风化）和定量自变量（各化学成分含量百分比）进行 *Logistic* 回归二分类和 LDA 线性判别分析二分类，以探究文物样品表面有无风化化学成分含量的统计规律。

Logistic 回归假设数据服从二项分布，通过极大化似然函数方法，运用梯度下降求解参数，最终达到数据二分类目的。*Logistic* 回归二分类的步骤和公式如下：

Step 1：记表面风化与否（伯努利分布）：

$$\begin{cases} P(y=1\,|\vec{x}) = F\left(\vec{x},\vec{\beta}\right) \\ P(y=0\,|\vec{x}) = 1 - F\left(\vec{x},\vec{\beta}\right) \end{cases} \tag{14}$$

其中，$F\left(\vec{x},\vec{\beta}\right)$ 定义域为[0,1]，值域为[0,1]；

Step 2：$F\left(\vec{x},\vec{\beta}\right)$ 取 *Sigmoid* 函数；

$$F\left(\vec{x},\vec{\beta}\right) = S\left(\vec{x}_i',\vec{\beta}\right) = \frac{exp\left(\vec{x}_i',\vec{\beta}\right)}{1+exp\left(\vec{x}_i',\vec{\beta}\right)} \tag{15}$$

Step 3：用极大似然方法（MLE）进行估计；

Step 4：用 $\hat{y}$ 表征“*y=1*”发生的概率；

Step 5：等量代换得到最终公式；

$$\hat{y}_i = P(y_i=1\,|\vec{x}) = S\left(\vec{x}_i',\vec{\beta}\right) = \frac{exp\left(\vec{x}_i',\vec{\beta}\right)}{1+exp\left(\vec{x}_i',\vec{\beta}\right)} \tag{16}$$

Step 6：若 $\hat{y} \geq 0.5$，则认为其预测的 *y=1*；反之认为其预测的 *y=0*。

LDA 线性判别分析是一种经典的分类算法，核心思想是以一种基于降维的方式将全体样本映射到一维坐标轴上，在此基础上设定一个阈值，将样本进行区分。LDA 线性判别分析二分类的计算流程如表 6 所示。

表 6：LDA 线性判别步骤

LDA 线性判别步骤
Step 1：记两类样本（风化与无风化）的中心在直线上的投影
Step 2：计算投影后两类样本的类内类间散度值
Step 3：计算两类样本的中心在直线上的投影的距离的平方
Step 4：构造目标函数

Step 5：拉格朗日乘子法最优求解得到最优 w

Output：输出最优 w

5.2.2 模型求解

在用 SPSS 26 对全部样本数据进行了正态检验和方差齐性检验之后，满足参数检验的条件，继续完成 Pearson 相关性分析和单因素 ANOVA 分析，以去掉共线性较大、相对意义较小的自变量。

我们先借助 Python 编写的算法对数据分别进行 Pearson 相关性分析，预览化学成分含量百分比之间的共线性关系，如图 5 所示。

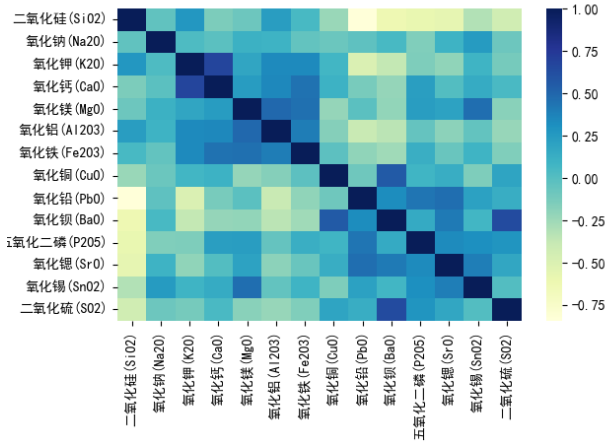

图 5：皮尔逊相关分析结果

从上图中我们可以看出自变量之间还是存在一定的线性相关性，有几对变量之间的关联性显著，这会对我们之后的分类结果产生影响，因此我们接下来又进行了单因素 ANOVA 分析以减少自变量的个数。

单因素 ANOVA 分析的结果如下表 7

表 7：单因素 ANOVA 分析的结果

			平方和	自由度	均方	F	显著性
二氧化硅(SiO2)	组间	（组合）	7632.796	1	7632.796	15.804	0.000
氧化钠(Na2O)	组间	（组合）	14.616	1	14.616	6.647	0.012
氧化钾(K2O)	组间	（组合）	131.388	1	131.388	10.563	0.002
氧化钙(CaO)	组间	（组合）	2.964	1	2.964	0.568	0.454
氧化镁(MgO)	组间	（组合）	0.762	1	0.762	2.337	0.131
氧化铝(Al2O3)	组间	（组合）	98.141	1	98.141	12.202	0.001
氧化铁(Fe2O3)	组间	（组合）	6.922	1	6.922	6.124	0.016
氧化铜(CuO)	组间	（组合）	1.995	1	1.995	0.401	0.529
氧化铅(PbO)	组间	（组合）	7002.812	1	7002.812	25.325	0.000
氧化钡(BaO)	组间	（组合）	199.262	1	199.262	2.939	0.091
五氧化二磷(P2O5)	组间	（组合）	165.133	1	165.133	16.795	0.000
氧化锶(SrO)	组间	（组合）	0.16	1	0.16	1.312	0.256
氧化锡(SnO2)	组间	（组合）	0.228	1	0.228	0.691	0.409
二氧化硫(SO2)	组间	（组合）	23.462	1	23.462	3.502	0.066
数值化玻璃类型	组间	（组合）	0.403	1	0.403	2.055	0.156

取显著性水平 α 为 0.1，当自变量对应的显著性 p 值大于 0.1 时，表明其影响不显

著，把对应的自变量筛除。因此，根据单因素 ANOVA 分析的结果，将氧化钙、氧化镁、氧化铜、氧化锶、氧化锡、数值化玻璃类型筛除，不再用于之后的二分类算法，以减小共线性和差异性对分类模型的影响。经过数据的清洗筛除后，提取保留二氧化硅、氧化钠、氧化钾、氧化铝、氧化铁、氧化铅、氧化钡、五氧化二磷、二氧化硫化学成分含量百分比继续作为自变量。

完成这些工作后，接下来，以数值化风化与否作为因变量，以二氧化硅、氧化钠、氧化钾、氧化铝、氧化铁、氧化铅、氧化钡、五氧化二磷、二氧化硫化学成分含量百分比作为自变量，利用 SPSS 26，分别采用 Logistic 回归和 LDA 线性判别分析探索表面有无风化的二分类规律。

Logistic 回归二分类结果如表 8 所示，自变量系数如表 9 所示。

表 8

分类表 a					
	实测		预测		
			风化与否		正确百分比
			0	1	
	风化与否	0	29	6	82.9%
		1	7	25	78.1%
	总体百分比				80.6%
分界值为 0.500					

表 9

变量	SiO2	Na2O	K2O	Al2O3	Fe2O3	PbO	BaO	P2O5	SO2	常量
系数	-0.333	-0.943	-0.76	-0.541	-1.679	-0.266	-0.516	-0.068	-0.127	33.512

LDA 线性判别分析二分类结果如表 10 所示，自变量系数如表 11 所示。

表 10

分类结果 a					
		风化数字化	预测组成员信息		总计
			0	1	
原始	计数	0	29	6	35
		1	6	26	32
	%	0	82.9	17.1	100.0
		1	18.8	81.3	100.0
正确率： 82.1%					

表 11

		SiO2	Na2O	K2O	Al2O3	Fe2O3	PbO	BaO	P2O5	SO2	常量
系数	0	22.361	23.987	37.706	24.928	37.983	22.248	29.114	29.767	13.42	-1114.267
	1	22.127	23.165	37.218	24.476	36.861	22.07	28.767	29.793	13.384	-1090.883

Logistic 回归和 LDA 线性判别分析这两种二分类算法的效果都较好，Logistic 回归的总体分类正确率为 80.6%，LDA 线性判别分析的总体分类正确率为 82.1%，仅相差 1.5%，正确率非常接近；与此同时，本模型选用的两个算法较好地探究了风化与否跟化学成分的二分类规律，输出了各个化学成分的系数。

Logistic 回归和 LDA 线性判别分析两个算法的系数表可以看出，金属氧化物对表面风化的影响明显大于非金属氧化物。其中，氧化铁与表面风化的关系最密切，根据化学方面的知识，氧化铁非常容易在空气中生锈，说明氧化铁确实对表面风化有很大的影响。因此，本文选用的两个二分类算法，对风化与否跟化学成分含量的统计规律有较好的解释性。

5.3 探究风化点于风化前化学成分含量的预测模型

在本模型中，利用上文 1.2 中 *Logistic* 回归算法得到的函数表达式作为基础，通过对全体采样点中风化采样点的自变量值（化学成分含量）迭代增减指定步长，从而以逐步搜索的形式探索至风化点的因变量值（风化与否）突变为 0（无风化），此时求解得到的各化学成分含量（各自变量值）即为风化点在风化之前的各化学成分含量预测值。需要说明的是，求解得到的预测值可以是多个离散组，也可以是区间划分，由于区间划分可能涉及到多区间交叉，求解效果受步长方向的影响较大，在这里，我们选择以多个离散组的形式呈现预测结果。

5.3.1 模型建立

Step 1：将 1.2 单因素 ANOVA 分析筛选出的化学成分含量作为函数表达式的自变量，风化与否（转化为数值型虚拟变量，无风化为 0，风化为 1）作为函数表达式的因变量。取各自变量的搜索步长为*-0.1*权重系数*；

Step 2：由 1.2 可知，Logistic 回归和 LDA 线性判别分析的分类准确率非常相近，由于 Logistic 回归的权重系数较小，更适合满足 Step 1 的定比例步长的逐步搜索。Logistic 回归的自变量权重系数及截距为：

'二氧化硅'	'氧化钠'	'氧化钾'	'氧化铝'	'氧化铁'	'氧化铅'	'氧化钡'	'五氧化二磷'	'二氧化硫'	'常量'
-0.333	-0.943	-0.760	-0.541	-1.679	-0.266	-0.516	-0.068	-0.127	33.512

Step 3：通过对自变量值迭代增减指定步长，递进收敛自变量值至因变量值从 1 突变为 0；

Step 4：由于 Logistic 回归对于$\hat{y} \geq 0.5$，则认为其预测的 *y=1*，反之认为其预测的 *y=0*，因此对于$\hat{y} < 0.5$时，因变量都赋值为 0，此时离散组求解结果数目非常多。为便于大大提高预测精确度和展示，同时提高机器计算的误差容错率，将概率判定区间缩小至$[-0.05,0.05]$，即预测风化点风化之前的**完全无风化**（$\hat{y} \approx 0$）的化学成分含量。

Step 5：假设每个采样点的化学总量为 100，每个采样点的各初始化学成分含量等于化学成分含量百分比。

Step 6：将每个风化采样点化学成分量加上 Step 1 设定的步长，再将其准换为化学成分含量百分比输入本模型的函数表达式。

Step 7：将输出值$\hat{y}$的绝对值与上一次的输出绝对值比较，若本次输出绝对值更小，则认为本次迭代比上一次更接近**完全无风化**的化学成分含量百分比，从而更新上次的记录数据。

Step 8：重复 Step 5、Step 6、Step 7 进行迭代，迭代 600 次，输出风化点风化之前**完全无风化**的化学成分含量百分比和$\hat{y}$。

5.3.2 模型求解

利用 Python 编写算法求解，部分风化点在风化之前完全无风化的部分化学成分含量百分比预测值如下表 12 所示，完整数据见支撑材料“完全风化前化学成分.xlsx”。根据化学元素规律，铅钡元素主导的氧化铅、氧化钡，其相对质量远高于钾元素主导的氧化钾，因此，铅钡玻璃样品完全无风化时的氧化铅、氧化钡含量百分比，远大于高钾玻璃样品完全无风化时的氧化钾含量百分比，说明预测值符合基本常理。

表 12：预测结果

采样点	二氧化硅	氧化钾	氧化镁	氧化铝	氧化铁	氧化铜	氧化铅	氧化钡	五氧化二磷
08	20.22	0.05	0.03	1.52	0.08	8.59	27.69	34.84	3.08
11	33.81	0.26	0.61	2.99	2.14	4.21	24.74	16.06	8.29
12	94.27	1.05	0.06	1.49	0.33	1.60	0.06	0.06	0.15
19	30.15	0.44	0.46	4.26	3.60	2.75	41.34	6.26	7.30
26	19.85	0.04	0.03	0.78	0.07	9.00	28.68	35.32	2.76
34	35.93	0.32	0.51	1.83	0.98	1.25	44.96	11.16	0.29
36	39.56	0.15	0.76	1.63	0.36	0.66	41.33	11.02	0.07
38	32.97	0.42	0.31	2.79	0.47	0.64	48.10	10.52	0.43
39	27.19	0.43	0.12	0.73	1.72	0.52	56.59	10.08	0.77
40	17.60	0.56	0.13	0.71	2.32	0.13	64.94	10.01	1.11
41	18.57	0.52	2.41	3.63	2.94	0.17	43.24	10.53	6.76
43 部位 1	12.77	1.71	0.64	2.88	3.01	3.84	57.32	9.10	0.82
43 部位 2	22.37	1.24	0.73	4.21	4.31	1.16	43.51	3.94	10.47
49	28.83	1.22	1.38	5.61	3.51	0.66	33.79	6.33	10.55
50	18.06	3.02	0.40	2.08	0.61	0.96	42.79	15.58	5.59
51 部位 1	24.68	1.84	1.05	5.70	1.95	1.21	39.32	9.62	7.32
51 部位 2	22.55	2.68	0.94	3.58	2.78	0.48	49.19	1.90	6.26
52	26.24	3.14	0.39	1.47	0.90	0.50	45.01	10.69	4.37
54	23.48	0.79	0.68	6.65	1.14	0.44	50.98	10.72	2.59
56	29.14	1.15	1.07	1.89	1.25	0.76	40.97	15.72	2.47
57	25.40	0.98	0.91	2.22	1.06	1.12	44.78	17.60	0.92

六、 问题二模型的建立与求解

本问要求根据先已有数据，建立高钾玻璃和铅钡玻璃的分类模型，其关键在于找出两种玻璃化学成分上的特征。我们通过四种有监督学习算法，划分训练集与测试集，分别求解分类结果，再横向进行比较，得出其中较优的算法。为了准确地划分亚类，我们先运用熵权法算出权重，在已有多个化学成分特征中筛选出最具有代表性的一个或几个成分，再通过 K-means 聚类划分亚类，得出具体划分方案。

6.1 玻璃分类模型的建立

6.1.1 基于 KNN 算法的分类模型

KNN 算法是一种常用的分类算法，其原理是当预测一个新的值时，根据它最近的 k 个点的类别来判断待分类点属于哪个类别。具体算法流程如下：

Step1：计算待分类点与已知类别的点之间的距离。这里距离使用的是欧氏距离，即

$$d=\sqrt{(x_2-x_1)\ +(y_2-y_1)^2} \tag{17}$$

Step2：按照距离递增次序排序；

Step3：选取与待分类点距离最小的 k 个点；

Step4：确定 k 的值后，计算前 k 个点所在类别的出现次数；

Step5：以前 k 个点出现次数最高的类别作为待分类点的预测分类。

由于当 k 值较小时，训练误差会相对较小。但如果 k 过小，如 k 取 1 时，模型会会产生过拟合现象，容易受到噪声值影响。我们经过几次尝试，最终将 k 值定为 2。

6.1.2 基于朴素贝叶斯分类器的分类模型

贝叶斯分类算法是统计学中的一种分类算法，利用贝叶斯公式建立概率模型，其基本原理是通过样本的先验概率，利用贝叶斯公式计算出其后验概率，即该样本属于某一类别的概率，最终选择具有最大后验概率的类别作为该样本的类别[1]。具体算法流程如下：

Step1：设待分类项为 $x=\{a_1,a_2,...,a_m\}$，a_i 为 x 的一个特征属性，$i=1,2,...,m$；类别集合 $C=\{y_1,y_2\}$；

Step2：根据贝叶斯公式 $P(y_i|x)=\dfrac{P(x|y_i)P(y_i)}{P(x)}$，计算 $P(y_1|x)$, $P(y_2|x)$，

Step3：如果 $P(y_k|x)=\max\{P(y_1|x),P(y_2|x)\}$，则 $x\in y_k$, $k=1,2$。

6.1.3 基于多层感知机的分类模型

感知机能够把训练集的正例和反例划分为两个部分，并且能够对未来输入的数据进行分类。感知机可以有一个或多个输入，但只有一个输出，输入与输出之间还包括激活函数和偏置项。基本过程是感知机接受输入，将它们乘以权重，然后将结果传递到激活函数以产生输出。偏置项的作用是避免输入为零对权重产生影响。假设输入 x 是一个 n 维的向量，即 $x=(x_1,x_2,...,x_n)$，可以给出感知机 *f(x)* 的基本模型：

$$f(x)=sign(w\times x+b) \tag{18}$$

其中，向量 $x=(x_1,x_2,...,x_n)$ 中每个分量代表向量 x 的每个分量 x_i 的权重，b 称为偏差，$w\times x$ 表示向量 w 和 x 的内积，*sign* 是一个符号函数，其表达式如下：

$$sign(x)=\begin{cases}+1, & x\geq 0\\ -1, & x<0\end{cases} \tag{19}$$

上述函数 *f(x)* 称为感知机。如果下图的神经网络结构中，每个神经元都是由感知机构成，则这个网络称为 MLP 多层感知机。根据此原理，我们可以通过感知机对玻璃的类型进行判别。

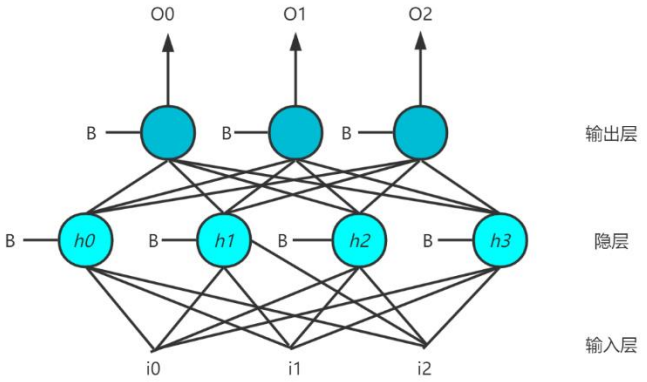

图 6：神经网络结构

6.1.4 基于随机森林的分类模型

随机森林可以解决分类和回归问题，并且使用的是随机样本和随机特征，即随机行和列，减少了基模型的相关性[2]。随机森林由多个独立的决策树组成的，当有一个

新的待分类样本进入时，就让森林中的每一棵决策树都分别进行判断该样本所属类别，被选择最多的类别就作为待分类样本的类别输出。每棵决策树的生成过程如下：

Step1：如果训练集大小为 *N*，对于每棵树而言，随机且有放回地从训练集中的抽取 *N* 个训练样本，作为该树的训练集；

Step2：从 *M* 个特征中，选取 *m*（*m*<<*M*）个特征子集，每次选择 *m* 个特征中最优的进行分裂。

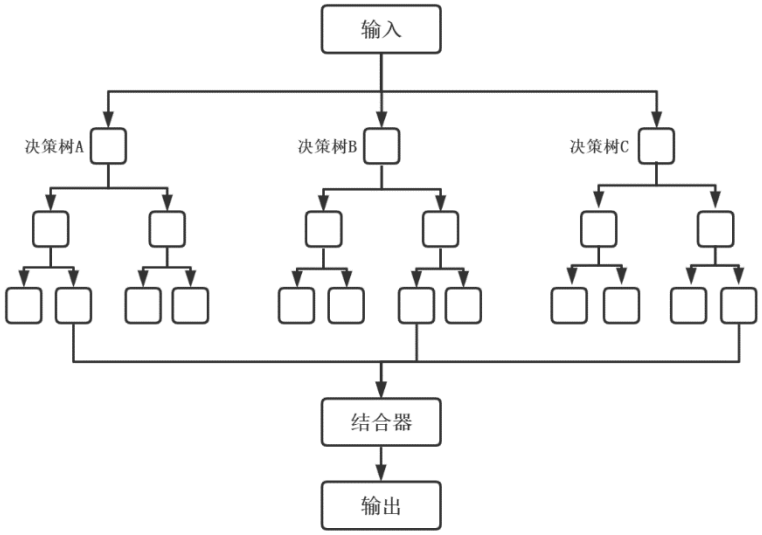

图 7：随机森林算法流程

上图为随机森林算法的具体流程，重复 *n* 次决策树的生成过程就能构建出 *n* 棵不同的决策树。构建完成后，再进行投票选择的流程，就能得出随机森林算法的最终分类结果。

6.2 玻璃分类模型的求解

我们通过 *Python* 的 *sklearn* 库实现这四种算法，表 13~16 是四种算法对于测试集的部分预测结果，完整结果见附录。

表 13：KNN 算法部分预测结果

高钾玻璃概率	铅钡玻璃概率	预测结果	原始标签
0	1	1	1
1	0	0	0
0	1	1	1
1	0	0	0

表 14：贝叶斯分类器部分预测结果

高钾玻璃概率	铅钡玻璃概率	预测结果	原始标签
2.55E-27	1	1	1
1	2.44E-26	0	0
2.20E-70	1	1	1
1	1.08E-28	0	0

表 15：MLP 算法部分预测结果

高钾玻璃概率	铅钡玻璃概率	预测结果	原始标签
0.07	0.93	1	1
0.98	0.02	0	0

5.88E-06	1	1	1
0.93	0.07	0	0

表 16：随机森林算法部分预测结果

高钾玻璃概率	铅钡玻璃概率	预测结果	原始标签
0.05	0.95	1	1
0.74	0.26	0	0
0.02	0.98	1	1
0.88	0.12	0	0

对比预测结果与原始标签，能够看出我们选用的四种模型对于高钾玻璃和铅钡玻璃的分类的准确度都是 100%。虽然四个最终预测结果是一致的，但是模型内部对于两种玻璃类型划分的概率还是存在细微的差距。由于使用 KNN 算法的分类模型无法得出具体的概率值，只有 0 与 1 两种极端值，因此我们在最终的玻璃分类模型中不考虑 KNN 算法，将剩余三种算法，即贝叶斯分类器、MLP 算法和随机森林算法中相加后取均值，概率均值就作为最终的玻璃分类结果。

6.3 高钾玻璃亚类划分模型与铅钡玻璃亚类划分模型的建立与求解

为了划分亚类，我们需要先提取高钾玻璃和铅钡玻璃的显著特征，这里我们选用熵权法对两种玻璃的数据集进行分析，得到差异性较大的一个或几个特征化学成分，根据这些成分进行亚类的划分。

熵权法（EWM）将指标值的变异性定义为信息熵，信息熵越小，指标变异程度越大，就认为它所反映的信息就越多。我们借助软件 *SPSSAU* 进行计算，得到两种玻璃各化学成分的信息效用值，具体数据如下表所示：

表 17：两种玻璃含有的各化学成分的信息效用值

化学成分	信息效用值 *d*	
	高钾玻璃	铅钡玻璃
二氧化硅(SiO2)	0.1245	0.0392
氧化钠(Na2O)	0.2647	0.1441
氧化钾(K2O)	0.1391	0.1007
氧化钙(CaO)	0.1496	0.0937
氧化镁(MgO)	0.1353	0.0559
氧化铝(Al2O3)	0.0937	0.0867
氧化铁(Fe2O3)	0.1752	0.1065
氧化铜(CuO)	0.1182	0.1629
氧化铅(PbO)	0.1770	0.0519
氧化钡(BaO)	0.2461	0.0670
五氧化二磷(P2O5)	0.1770	0.1579
氧化锶(SrO)	0.1949	0.0546
氧化锡(SnO2)	0.2152	0.0958
二氧化硫(SO2)	0.1107	0.2365

由上表可以看出，在高钾玻璃类别中，氧化钠(Na2O)的信息效用值最大，显著高于其他化学成分，因此氧化钠(Na2O)就作为高钾玻璃的亚类划分特征；在铅钡玻璃类别中，信息效用值最大与最小之间差距较大，中间值也较为分散，因此我们选择了三个化学成分作为下文亚类划分的依据，分别是二氧化硫(SO2)、五氧化二磷(P2O5)和氧化铜(CuO)。在铅钡玻璃的三个亚类划分依据中，有两个非金属氧化物和一个金属氧

化物，同时磷元素和硫元素的原子序数也邻近，因此本文以磷/硫（P/S）氧化物和氧化铜(CuO)作为铅钡玻璃亚类划分的特征，最终可以得到两种以不同特征划分的亚类划分方法。

总结出两类玻璃的三种划分特征后，我们使用了无监督算法 K-means 聚类，以达到尽量增大类间距离，缩小类内差距的目的，以便进一步求解出每一类的聚类中心和亚类的划分阈值。我们将本问上文得到的三个模型输出的类型概率再求均值后作为聚类的 x 轴，将化学成分含量比例按式（20）进行归一化处理后作为 y 轴。对于磷硫氧化物共同作为一个划分特征时，我们先将二氧化硫(SO2)和五氧化二磷(P2O5)的含量比例相加，再进行归一化处理。

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{20}$$

考虑到本身数据点就较少，若再分成多个类别可能会导致类间差距过下，因此我们就选定 k 值为 2。我们通过 *Python* 程序处理数据并且实现聚类算法，事先设定聚类参数为 2，下一步将数据拟合到 K-means 实例中，然后创建模型。*fit_predict* 运算这些值，获取聚类标签[3]。为了便于划分，我们将聚类后两个聚类中心的 y 值取平均作为划分类别的阈值。具体结果如下图所示：

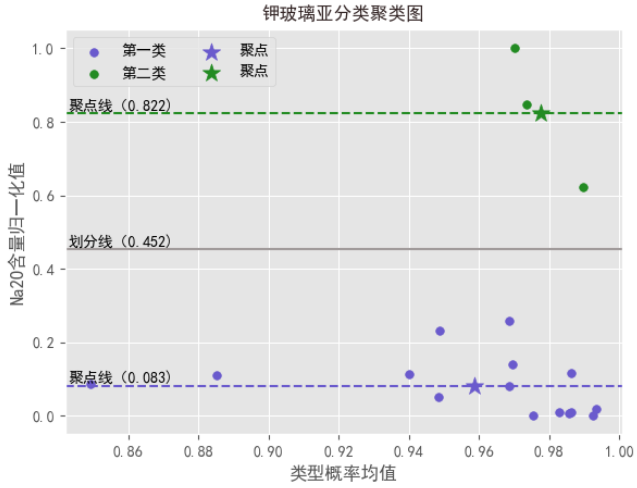

图 8：高钾玻璃—钠亚类聚类结果

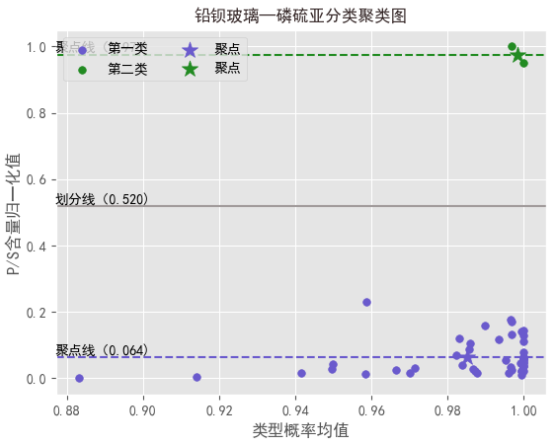

图 9：铅钡玻璃—磷硫亚类聚类结果

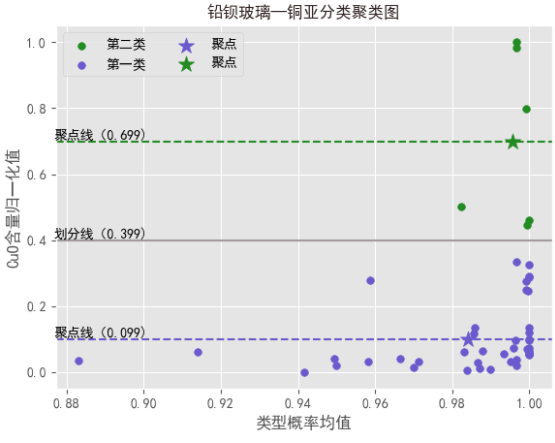

图 10：铅钡玻璃—铜亚类聚类结果

由图可看出，聚类的效果是较优的，数据点的聚集也较为明显且聚点之间的距离

也相对远。得到聚点和划分线的数据后，我们又通过归一化计算式（20）反推得到化学成分的实际含量比例，聚类结果数据和计算结果见下表。

表 18：聚类结果数据和计算结果

亚类	聚点坐标	划分线	实际含量比例阈值(%)
高钾玻璃—钠亚类	[0.9777, 0.8224]	0.452	1.5278
	[0.9587, 0.0825]		
铅钡玻璃—磷硫亚类	[0.9953, 0.5382]	0.520	8.5202
	[0.9813, 0.0772]		
铅钡玻璃—铜亚类	[0.9957, 0.6985]	0.399	4.2835
	[0.9841, 0.0986]		

由此，我们得到结论，在高钾玻璃的亚类划分中，当其氧化钠(Na2O)的含量比例低于 1.5278%时，将被分为高钾玻璃—低钠亚类，反之则被分为高钾玻璃—高钠亚类；在铅钡玻璃的亚类划分中，使用两种划分方式，一种是将二氧化硫(SO2)和五氧化二磷(P2O5)含量比例相加小于 8.5202%的玻璃分为铅钡玻璃—低磷硫亚类，反之被分为铅钡玻璃—高磷硫亚类；铅钡玻璃的另一种划分方式是将氧化铜(CuO)含量比例低于 4.2835%的玻璃分为铅钡玻璃—低铜亚类，反之被分为铅钡玻璃—高铜亚类。

6.4 亚分类模型的合理性和敏感性分析

亚类划分完毕后，还需分析该模型的合理性和敏感性，以证明该模型的实际可行性。对于合理性的证明，从前文的 K-means 聚类结果能直观看出三个聚类都能按化学成份很好地划分亚类型，分类效果明显。同时，划分线附近分布的数据点较少，数据点都集中在相应聚点附近，说明类间距离大而类内差距小，达到我们对该算法的预期效果。

我们通过波动划分线的方式来检测模型的敏感性，结果发现三种亚类划分方法均能在划分标准上下波动 10%的情况下，对结果不造成影响，由此证明我们的模型有较好的敏感性。敏感性分析图示见下图。

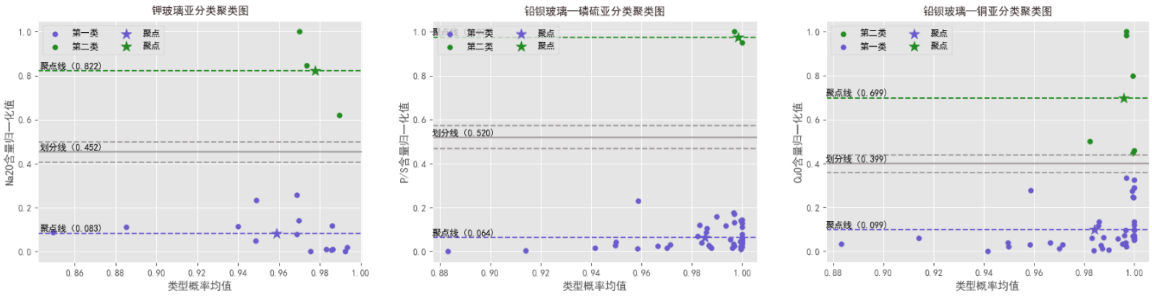

图 11：三种亚类划分方法敏感性分析

七、 问题三模型的建立与求解

根据问题二的分析，我们得到了高钾玻璃和铅钡玻璃的分类模型，以三种机器学习算法输出的类型均值作为最终类型概率。本文需要在问题二结论的基础上，分析未知类别的玻璃文物的化学成分，达到鉴别其类型的目的。

7.1 数据处理

在分类前，我们将附件表单 3 中未分类玻璃的化学成分比例数据进行处理，我们仍采用与前文相同的方式填充数据中空缺的部分，处理后的部分数据见下表，完整数

据见支撑材料“处理后的表单三数据.xlsx”。

表 19：表单 3 部分处理后数据（保留两位小数）

文物编号	SiO2	Na2O	K2O	CaO	MgO	Al2O3	…
A1	78.45	0.10	0.10	6.08	1.86	7.23	…
A2	37.75	0.41	0.41	7.63	0.41	2.33	…
A3	31.95	0.34	1.36	7.19	0.81	2.93	…
A4	35.47	1.33	0.79	2.89	1.05	7.07	…

7.2 分类结果

我们仍然借助上文的 *Python* 程序，通过训练后的机器学习算法得到对未知类别玻璃的鉴别结果，结果如下表所示。

表 20：鉴别结果

文物编号	预测结果			最终预测类型
	贝叶斯分类器	MLP 算法	随机森林	
A1	高钾	高钾	高钾	高钾
A2	铅钡	铅钡	铅钡	铅钡
A3	铅钡	铅钡	铅钡	铅钡
A4	铅钡	铅钡	铅钡	铅钡
A5	铅钡	铅钡	铅钡	铅钡
A6	高钾	高钾	高钾	高钾
A7	高钾	高钾	高钾	高钾
A8	铅钡	铅钡	铅钡	铅钡

根据上表可以看出，我们使用的三种机器学习算法最终的预测结果都是相同的，由此也能说明本文建立的模型是较为稳定的，具有可推广性。

7.3 敏感性分析

在本问中，我们通过改变输入参数，观察模型响应的相对变化的过程以进行敏感性分析。我们使用 *Python* 程序进行测试，发现三个算法都能在整体数据上下波动 40%的范围内，保持结果输出稳定不变。贝叶斯分类器和 MLP 算法能在每个数据点随机上下波动 30%范围内，保持结果稳定输出不变；随机森林算法能在每个数据点随机上下波动 10%范围内保持结果稳定输出不变。相对效果最好的是贝叶斯，能在每个数据随机波动 50%范围内，保持结果不变。本文使用的模型所得到的三个结果都较稳定，说明敏感度都较优。

八、 问题四模型的建立与求解

本问题要求根据附件表单 1 和表单 2 的数据，针对铅钡玻璃样品和高钾玻璃样品两种不同玻璃类型的样品，分别研究同类玻璃样品下，化学成分之间的关联关系；并在此基础上，分析和检验铅钡玻璃样品和高钾玻璃样品之间化学成分关联关系的差异性。在求解过程中，建立探究化学成分关联及关联差异性的统计分析模型。

化学成分关联及关联差异性的模型旨在分析检验不同玻璃样品之间化学成分关联关系的差异性。与 5.2（文物表面风化与否和化学成分含量关系的统计分析模型）类似，将全部数据划分成铅钡玻璃和高钾玻璃这 2 组，先对两组样本数据分别进行正态检验和方差齐性检验。检验满足正态分布和方差齐性之后，对 2 组玻璃文物样

品分别采用 *Pearson* 相关性分析，计算每组化学成分之间的 *Pearson* 相关系数，以得到化学成分之间的关联关系。将两组相关系数集按“成分 1-成分 2”的标签一一对应，作为 91 个标签组，用来做铅钡玻璃样品和高钾玻璃样品的配对样本 *t* 检验（*matched samples t- test*），用于检测不同玻璃类型间化学成分关联关系的差异性。

8.1 建立化学成分关联及关联差异性的模型

与上文 5.2 非常类似，将全部数据划分成铅钡玻璃样品和高钾玻璃样品这 2 组，每组样本量分别为 49 和 18，在分析检验关联及关联差异性之前，先对两组样本数据分别进行正态检验和方差齐性检验，两种检验的方法、公式、步骤与上文 5.2 一致。

检验满足正态分布和方差齐性之后，接下来对每组玻璃化学成分之间的关联关系进行分析，在这里我们使用 *Pearson* 相关性分析，求得每组玻璃的各化学成分间相关系数矩阵，并生成色阶热力图，*Pearson* 相关性分析的步骤和公式与上文 5.2 中一致。

通过计算得到两组玻璃样品的 *Pearson* 相关系数集，从而完成了对不同玻璃样品化学成分之间的关联关系进行分析的题目要求。

然而要探究不同玻璃样本间化学成分关联关系的差异性，需要借助配对样本 *t* 检验。先将两组相关系数集按“成分 1-成分 2”的标签一一配对，作为 91 个标签组，部分标签组展示如下表 21 所示。

表 21：部分标签组

相关系数名	高钾玻璃组	铅钡玻璃组
二氧化硅-二氧化硅	1	1
二氧化硅-氧化钠	-0.880078497	-0.349007111
二氧化硅-氧化钾	-0.835579006	-0.507545535
二氧化硅-氧化钙	-0.642749669	0.084954254
二氧化硅-氧化镁	-0.830732359	0.401109497

配对样本 *t* 检验旨在分析配对定量数据之间的差异对比关系。与独立样本 *t* 检验相比，配对样本 *t* 检验要求样本是配对的，两个样本的样本量相同，先后顺序一一对应。配对样本 t 检验的步骤与公式如下：

Step 1：提出原假设和备择假设：

H_0：两配对的样本存在显著差异影响

H_1：两配对的样本不存在显著差异影响

Step 2：根据式（22），构造并计算 *t* 统计量；

$$t=\frac{\bar{X}}{\frac{s}{\sqrt{n}}}\sim t(n-1) \tag{21}$$

Step 3：给定显著性水平 α；

Step 4：查表得到判断临界值t_α；

Step 5：若$t<t_\alpha$，按计算的显著性水平 α 拒绝原假设；若$t>t_\alpha$，接受原假设。（注：使用单侧检验）

8.2 求解化学成分关联及关联差异性的模型

采用 *Python* 编写 *Pearson* 相关性分析的代码，得到高钾玻璃组和铅钡玻璃组各自的化学成分关联关系的相关系数矩阵，并且将相关系数矩阵转化为色阶热力图进行辅助可视化。高钾玻璃组 *Pearson* 相关系数热力图如图 11 所示，铅钡玻璃组 *Pearson* 相

关系数热力图如图 12 所示，两组玻璃的具体相关系数数据见支撑材料。

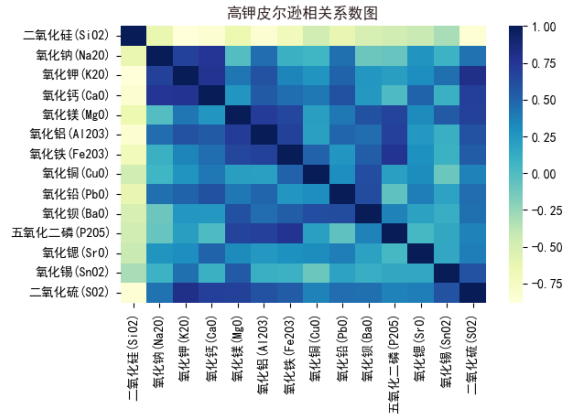

图 11：

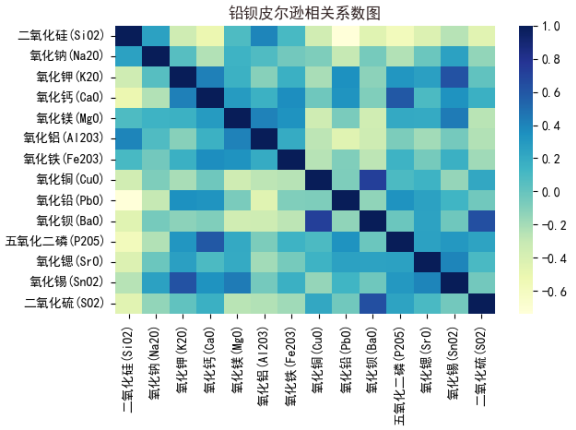

图 12：

根据两个相关系数矩阵热力图可以看出，高钾玻璃中，化学成分之间的正关联度较高，二氧化硅和其它化学成分呈负相关，其它化学成分之间呈正相关。在铅钡玻璃中，化学成分之间的负关联度较高，化学成分之间呈负相关的较多，也有部分化学成分之间呈相关系数绝对值不高的正相关。

将高钾玻璃组和铅钡玻璃组相关系数集的相关系数一一配对，使用 SPSSAU 进行配对样本 t 检验，检验不同组玻璃之间的化学成分关联关系的差异性。结果如表 22 所示。

表 22：配对 t 检验分析结果

配对编号	项	平均值	标准差	平均值差值	t	p
配对 1	高钾玻璃组	0.29	0.45	0.19	6.387	0.000***
	铅钡玻璃组	0.10	0.38			

* p<0.1 ** p<0.05 *** p<0.01

由上表可知，*p=0.000* 小于显著性水平 0.01，高钾玻璃组和铅钡玻璃组之间呈现出 0.01 水平的显著性，说明两组化学成分关联关系存在着显著差异性。

可以利用 Cohen's d 值研究差异幅度情况，其表示效应量大小（差异幅度大小），该值越大说明差异越大，值越小说明差异越小。效应量小、中、大的区分临界点分别是： 0.20、0.50、0.80。 配对 t 检验的效应量指标如下表 23 所示。

表 23：配对 t 检验效应量指标

名称	平均值差值	df	差值标准差	Cohen's d 值
高钾玻璃组-铅钡玻璃组	0.19	195	0.418	0.456

由上表可知，Cohen's d 值为 0.456，接近于 0.50，接近于中效应量水平，说明高钾玻璃组和铅钡玻璃组化学成分关联关系的差异幅度较明显，属于中等差异幅度，至此，不同类型玻璃之间的化学成分关联及关联差异性的探索结束。

九、 模型的评价与推广

本文基于各类机器学习，建立了分析玻璃文物表面风化与玻璃类型、纹饰和之间的关系模型，并且能基于统计规律预测风化前的化学成分含量；此外，根据现有的数据，本文还得到了高钾玻璃和铅钡玻璃两种玻璃的分类模型，能够基于化学成分对未知类别的玻璃文物进行鉴别，由于古代不同地区生产的玻璃制品的成分也不同，我们的结论对研究古代玻璃制品的来源地也起到了辅助的作用，对研究古代玻璃制品的发

展也有所帮助。

9.1 模型的优点

（1） 在进行模型的求解前，由于有些化学成分的差异性不明显，本文对原始数据进行了处理，提取出较为显著的特征，能够充分挖掘文物是否分化和玻璃分类的特征。

（2） 本文利用了四种有监督机器学习算法和一种无监督机器学习算法进行玻璃类别的判断，能够更充分地衡量玻璃分类的规律，模型的结果也更具推广性。

9.2 模型的缺点

本文分析所使用的数据量不够充分，在进行特征提取时可能会存在特征属性的数据局限性。若增加数据点，模型的泛用性会进一步提高。

9.3 模型的推广

总之，本文建立的模型与给定数据的适应性良好，若需要对实际文物的分类有参考作用，则需要对模型进行更深层次的分析，最终可以应用于实际的文物研究。

十、 参考文献

[1]郭羽含，陈虹，肖成龙编,面向新工科普通高等教育系列教材 Python 机器学习,机械工业出版社,2021.03,第 185 页.
[2]张春强，张和平，唐振,机器学习软件工程方法与实现,机械工业出版社,2021,第 289 页.
[3]（美）Sjbanjan Das，（美）Umit Mert Cakmak 著；谢琼娟译,自动机器学习入门与实践 使用 Python,华中科技大学出版社,2019.12,第 65 页.

附录

附录一：支撑材料文件列表
附录二：完整数据表格
附录三：Python 代码

附录 1 支撑材料文件列表
数据处理相关代码和结果
数据预处理.py
数据处理补空值去无效合并文物编号类型.xlsx
datafenghua.xlsx
问题一相关代码和结果
逻辑回归区间预测.py
Spearman 相关系数矩阵.m
高钾四检验.spv
铅钡四检验.spv
全部数据的 ANOVA 分析.spv
逻辑回归.spv
风化.xlsx
无风化.xlsx
卡方分析.xlsx
ANOVA 分析处理后数据.xlsx
斯皮尔曼.xlsx
LDA 分析数据.xlsx
逻辑回归数据.xlsx
完全风化前化学成分.xlsx
化学成分相关性热力图.png
问题二相关代码和结果
KNN.py
贝叶斯分类.py
多层感知神经网络.py
随机森林.py
KMeans.py
高钾.xlsx
铅钡.xlsx
多层感知网络权重.xlsx
随机森林权重.xlsx
KNN 结果概率.xlsx
贝叶斯结果概率.xlsx
神经网络结果概率.xlsx
随机森林结果概率.xlsx
四算法结果概率.xlsx
高钾玻璃亚分类.xlsx
铅钡-磷硫亚分类.xlsx

铅钡-铜亚分类.xlsx
高钾玻璃亚分类.png
铅钡_磷硫亚分类图.png
铅钡-铜亚分类图.png
高钾玻璃亚分类合理敏感分析.png
铅钡玻璃-磷硫亚分类合理敏感分析.png
铅钡玻璃-铜亚分类合理敏感分析.png
KNNROC.png
贝叶斯 ROC.png
神经网络 ROC.png
随机森林 ROC.png

问题三相关代码和结果

三大算法预测文物类型.py
处理后的表单三数据.xlsx
三算法预测文物结果.xlsx

问题四相关代码和结果

皮尔逊相关系数.py
皮尔逊相关系数.xlsx
配对样本 t 检验.xlsx
高钾皮尔逊相关系数.xlsx
铅钡皮尔逊相关系数.xlsx
高钾皮尔逊相关系数热力图.png
铅钡皮尔逊相关系数热力图.png

附录 2 完整数据表格

贝叶斯分类器完整预测结果

高钾概率	铅钡概率	预测结果	原始标签	高钾概率	铅钡概率	预测结果	原始标签
1	6.1E-31	0	0	1.47E-19	1	1	1
1	4.21E-33	0	0	5.64E-80	1	1	1
1	2.18E-26	0	0	1	6.67E-30	0	0
1	9.6E-29	0	0	3.06E-94	1	1	1
1	1.27E-28	0	0	2.3E-115	1	1	1
1	4.39E-25	0	0	1	3.25E-31	0	0
1	5.38E-24	0	0	1	3.43E-32	0	0
1	7.68E-31	0	0	8.16E-64	1	1	1
1	1.03E-29	0	0	1	2.17E-31	0	0
1	9.51E-36	0	0	1.8E-84	1	1	1
1	2.97E-32	0	0	1	5.04E-31	0	0
4.8E-43	1	1	1	9.09E-97	1	1	1
1	4.24E-21	0	0	7.3E-117	1	1	1
1.88E-81	1	1	1	1	1.64E-29	0	0
9.51E-77	1	1	1	5.53E-92	1	1	1

1.19E-74	1	1	1	4E-83	1	1	1
2.7E-16	1	1	1	7.09E-98	1	1	1
8.25E-23	1	1	1	4.3E-119	1	1	1
4.7E-12	1	1	1	3.8E-139	1	1	1
6.61E-30	1	1	1	2.21E-97	1	1	1
5.08E-31	1	1	1	1.9E-120	1	1	1
2.86E-27	1	1	1	8.71E-90	1	1	1
6.81E-45	1	1	1	2.09E-23	1	1	1
6.79E-47	1	1	1	1.98E-70	1	1	1
2.5E-59	1	1	1	1.7E-101	1	1	1
2.93E-34	1	1	1	1.88E-85	1	1	1
9.75E-55	1	1	1	1.85E-98	1	1	1
1.03E-17	1	1	1	1.37E-97	1	1	1
1.02E-05	0.99999	1	1	1.2E-110	1	1	1
6.24E-42	1	1	1	9.7E-116	1	1	1
9.04E-39	1	1	1	1.32E-92	1	1	1
3.74E-14	1	1	1	1.6E-102	1	1	1
5.02E-35	1	1	1	7.04E-81	1	1	1
4.69E-56	1	1	1				

KNN 算法完整预测结果

高钾概率	铅钡概率	预测结果	原始标签	高钾概率	铅钡概率	预测结果	原始标签
1	0	0	0	0	1	1	1
1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0
0	1	1	1	0	1	1	1
1	0	0	0	0	1	1	1
0	1	1	1	1	0	0	0
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1

0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1
0	1	1	1				

MLP 算法完整预测结果

高钾概率	铅钡概率	预测结果	原始标签	高钾概率	铅钡概率	预测结果	原始标签
0.988777374	0.011222626	0	0	0.091422248	0.908577752	1	1
0.997224821	0.002775179	0	0	6.55043E-05	0.999934496	1	1
0.99559523	0.00440477	0	0	0.9488955	0.0511045	0	0
0.976491991	0.023508009	0	0	4.62849E-05	0.999953715	1	1
0.985657468	0.014342532	0	0	1.36352E-08	0.999999986	1	1
0.995586357	0.004413643	0	0	0.979121956	0.020878044	0	0
0.979872416	0.020127584	0	0	0.986521649	0.013478351	0	0
0.980549	0.019451	0	0	0.000463405	0.999536595	1	1
0.990169095	0.009830905	0	0	0.980376462	0.019623538	0	0
0.998652044	0.001347956	0	0	0.000453014	0.999546986	1	1
0.998247269	0.001752731	0	0	0.976394802	0.023605198	0	0
0.002078929	0.997921071	1	1	5.71326E-05	0.999942867	1	1
0.707741927	0.292258073	0	0	3.65806E-09	0.999999996	1	1
0.002232284	0.997767716	1	1	0.95544028	0.04455972	0	0
0.00013343	0.99986657	1	1	8.76696E-05	0.99991233	1	1
7.38037E-05	0.999926196	1	1	3.8349E-05	0.999961651	1	1
0.06626128	0.93373872	1	1	0.000162942	0.999837058	1	1
0.065419277	0.934580723	1	1	0.001993437	0.998006563	1	1
0.190821108	0.809178892	1	1	0.038600325	0.961399675	1	1
0.018258799	0.981741201	1	1	0.000371299	0.999628701	1	1
0.054197896	0.945802104	1	1	0.043270764	0.956729236	1	1
0.0703796	0.9296204	1	1	0.002174312	0.997825688	1	1
0.006568331	0.993431669	1	1	0.030416173	0.969583827	1	1
0.00050213	0.99949787	1	1	6.87056E-05	0.999931294	1	1
0.000125497	0.999874503	1	1	0.000772017	0.999227983	1	1

0.002697769	0.997302231	1	1	0.00032254	0.99967746	1	1
0.000260006	0.999739994	1	1	0.000959142	0.999040858	1	1
0.070204604	0.929795396	1	1	0.000258972	0.999741028	1	1
0.117674412	0.882325588	1	1	0.001882774	0.998117226	1	1
0.000845297	0.999154703	1	1	0.013293041	0.986706959	1	1
0.002276452	0.997723548	1	1	0.000131794	0.999868206	1	1
0.075213875	0.924786125	1	1	0.000393526	0.999606474	1	1
0.004459355	0.995540645	1	1	0.000202869	0.999797131	1	1
0.000371831	0.999628169	1	1				

随机森林算法完整预测结果

高钾概率	铅钡概率	预测结果	原始标签	高钾概率	铅钡概率	预测结果	原始标签
0.92	0.08	0	0	0.06	0.94	1	1
0.98	0.02	0	0	0.09	0.91	1	1
0.66	0.34	0	0	1	0	0	0
0.87	0.13	0	0	0.01	0.99	1	1
0.92	0.08	0	0	0	1	1	1
0.91	0.09	0	0	0.98	0.02	0	0
0.84	0.16	0	0	0.97	0.03	0	0
0.94	0.06	0	0	0	1	1	1
0.92	0.08	0	0	1	0	0	0
0.97	0.03	0	0	0	1	1	1
0.96	0.04	0	0	0.95	0.05	0	0
0	1	1	1	0.01	0.99	1	1
0.84	0.16	0	0	0.01	0.99	1	1
0	1	1	1	0.89	0.11	0	0
0.01	0.99	1	1	0	1	1	1
0.01	0.99	1	1	0	1	1	1
0.02	0.98	1	1	0	1	1	1
0.11	0.89	1	1	0.01	0.99	1	1
0.16	0.84	1	1	0	1	1	1
0.03	0.97	1	1	0.03	0.97	1	1
0.07	0.93	1	1	0.01	0.99	1	1
0.03	0.97	1	1	0.04	0.96	1	1
0.03	0.97	1	1	0.01	0.99	1	1
0	1	1	1	0.02	0.98	1	1
0	1	1	1	0.01	0.99	1	1
0	1	1	1	0	1	1	1
0	1	1	1	0.05	0.95	1	1
0.08	0.92	1	1	0	1	1	1
0.14	0.86	1	1	0	1	1	1
0	1	1	1	0.03	0.97	1	1
0	1	1	1	0	1	1	1

0.05	0.95	1	1	0	1	1	1
0.01	0.99	1	1	0	1	1	1
0	1	1	1				

附录 3 Python 代码

数据预处理.py

```
import numpy as np
import pandas as pd
import math
def is_number(s):
    try:
        float(s)
        return True
    except:
        return False
#表单一
Forms_one=pd.DataFrame(pd.read_excel("附件.xlsx",sheet_name=0))
#表单二
Forms_two=pd.DataFrame(pd.read_excel("附件.xlsx",sheet_name=1))
Glass_type=[]
Artifact={}
Artifact['文物编号']= {}
Artifact['类型']= {}
Artifact['表面风化']={}
for i in range(len(Forms_one)):
    Artifact['文物编号'][i]=Forms_one.loc[i,'文物编号']
    Artifact['类型'][i]=Forms_one.loc[i,'类型']
    Artifact['表面风化'][i] = Forms_one.loc[i, '表面风化']
chemical_composition=Forms_two.to_dict('dict')

chemical_biaohao=[]
for i in chemical_composition:
    if(i !='文物采样点'):
        chemical_biaohao.append(i)

chemical_composition['文物编号']={}
chemical_composition['类型']={}
chemical_composition['表面风化']={}
for i in range(len(Forms_two)):
    muber=int(chemical_composition['文物采样点'][i][:2])
    chemical_composition['文物编号'][i]=Artifact['文物编号'][muber-1]
```

```
        chemical_composition['类型'][i] = Artifact['类型'][muber - 1]
        chemical_composition['表面风化'][i] = Artifact['表面风化'][muber -
1]
    for i in range(len(Forms_two)):
        count_sum=0
        nan_number=0
        nan_index=[]
        for s in chemical_biaohao:
            if(math.isnan(Forms_two.loc[i,s])):
                nan_number+=1
                nan_index.append(s)
            else:
                count_sum+=Forms_two.loc[i,s]
        if(count_sum<85 or count_sum>105):
            for s in chemical_composition:
                print(s)
                chemical_composition[s].pop(i)
        else:
            count_sum=100-count_sum
            for s in nan_index:
                chemical_composition[s][i]=count_sum/nan_number
    chemical_composition['类型数字化']={}
    for i in chemical_composition['类型']:
        if(chemical_composition['类型'][i]=='高钾'):
            chemical_composition['类型数字化'][i]=0
        elif(chemical_composition['类型'][i]=='铅钡'):
            chemical_composition['类型数字化'][i] =1
    data=pd.DataFrame(chemical_composition)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('datafenghua0.xlsx',sheet_name='data',in
dex=False)
```

贝叶斯分类.py

```
    from sklearn.naive_bayes import MultinomialNB
    import pandas as pd
    from sklearn.model_selection import train_test_split
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    lis=['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)',
        '风化数字化']
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='类型数字化'):
                datay.append(data['类型数字化'][i])
            elif (j in lis):
                datax[i].append(data[j][i])
    x_train, x_test, y_train, y_test = train_test_split(datax, datay,
test_size=0.2, random_state=10)
    clf = MultinomialNB()
    clf.fit(x_train,y_train)
    print("结果预测准确度：")
    print(clf.score(x_test,y_test))
    pre=clf.predict_proba(datax)

    pre_dict={}
    pre_dict['高钾概率']=[]
    pre_dict['铅钡概率']=[]
    pre_dict['预测结果']=clf.predict(datax)
    pre_dict['原始标签']=datay
    for i in range(len(pre)):
        pre_dict['高钾概率'].append(pre[i][0])
        pre_dict['铅钡概率'].append(pre[i][1])
    data=pd.DataFrame(pre_dict)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('贝叶斯结果概
率.xlsx',sheet_name='data',index=0)

    y_score =clf.predict_proba(x_test)
    from sklearn import metrics
    from pylab import *
    import matplotlib.pyplot as plt
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure()
```

```
    name=["高钾","铅钡"]
    for i in range(2):
        fpr, tpr, thresholds = metrics.roc_curve(y_test, y_score[:,
i],pos_label=i)
        auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr,
                 lw=2, label="贝叶斯_ROC "+name[i]+" (area = %0.2f)" %
auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

KNN.py

```
    from sklearn.neighbors import KNeighborsClassifier
    import pandas as pd
    from sklearn.model_selection import train_test_split
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    lis=['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)',
         '风化数字化']
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='类型数字化'):
                datay.append(data['类型数字化'][i])
```

```
            elif(j in lis):
                datax[i].append(data[j][i])
    x_train, x_test, y_train, y_test = train_test_split(datax, datay,
test_size=0.2, random_state=10)
    #构建knn分类模型，并指定 k 值
    KNN=KNeighborsClassifier(n_neighbors=3)
    #使用训练集训练模型
    model=KNN.fit(x_train,y_train)
    print("结果预测准确度：")
    print(model.score(x_test,y_test))

    pre=model.predict_proba(datax)
    pre_dict={}
    pre_dict['高钾概率']=[]
    pre_dict['铅钡概率']=[]
    pre_dict['预测结果']=model.predict(datax)
    pre_dict['原始标签']=datay
    for i in range(len(pre)):
        pre_dict['高钾概率'].append(pre[i][0])
        pre_dict['铅钡概率'].append(pre[i][1])
    data=pd.DataFrame(pre_dict)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('KNN结果概
率.xlsx',sheet_name='data',index=0)

    y_score =model.predict_proba(x_test)
    from sklearn import metrics
    from pylab import *
    import matplotlib.pyplot as plt
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure()
    name=["高钾","铅钡"]
    for i in range(2):
        fpr, tpr, thresholds = metrics.roc_curve(y_test, y_score[:,
i],pos_label=i)
        auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr,
                 lw=2, label="KNN_ROC "+name[i]+" (area = %0.2f)" %
auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

```
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

多层感知神经网络.py

```
    from sklearn import neural_network
    from sklearn.model_selection import train_test_split
    import pandas as pd
    import warnings
    warnings.filterwarnings('ignore')
    mlp=neural_network.MLPClassifier(hidden_layer_sizes=(10), #隐藏层
                                     activation='relu',  #激活函数
                                     solver='adam',
                                     alpha=0.0001,  #正则化项系数
                                     batch_size='auto',
                                     learning_rate='constant',  #学习率
                                     learning_rate_init=0.001,
                                     power_t=0.5,
                                     max_iter=450,#迭代次数
                                     tol=1e-4)
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    lis=['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)',
         '风化数字化']
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='类型数字化'):
                datay.append(data['类型数字化'][i])
            elif(j in lis):
```

```
                datax[i].append(data[j][i])
    x_train, x_test, y_train, y_test = train_test_split(datax, datay,
test_size=0.2, random_state=10)
    mlp.fit(x_train,y_train)
    print("预测准确度：")
    print(mlp.score(x_test,y_test))

    pre=mlp.predict_proba(datax)
    pre_dict={}
    pre_dict['高钾概率']=[]
    pre_dict['铅钡概率']=[]
    pre_dict['预测结果']=mlp.predict(datax)
    pre_dict['原始标签']=datay
    for i in range(len(pre)):
        pre_dict['高钾概率'].append(pre[i][0])
        pre_dict['铅钡概率'].append(pre[i][1])
    data=pd.DataFrame(pre_dict)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('神经网络结果概
率.xlsx',sheet_name='data',index=0)

    data=pd.DataFrame(mlp.coefs_)
    data_list=mlp.coefs_[0].tolist()
    cl=[]
    for i in range(10):
        cl.append(mlp.coefs_[1].tolist()[i][0])
    data_dict={}
    data_dict['神经网络层级']=[i+1 for i in range(10)]
    data_dict['二氧化硅(SiO2)']=data_list[0]
    data_dict['氧化钠(Na2O)']=data_list[1]
    data_dict['氧化钾(K2O)']=data_list[2]
    data_dict['氧化钙(CaO)']=data_list[3]
    data_dict['氧化镁(MgO)']=data_list[4]
    data_dict['氧化铝(Al2O3)']=data_list[5]
    data_dict['氧化铁(Fe2O3)']=data_list[6]
    data_dict['氧化铜(CuO)']=data_list[7]
    data_dict['氧化铅(PbO)']=data_list[8]
    data_dict['氧化钡(BaO)']=data_list[9]
    data_dict['五氧化二磷(P2O5)']=data_list[10]
    data_dict['氧化锶(SrO)']=data_list[11]
    data_dict['氧化锡(SnO2)']=data_list[12]
    data_dict['二氧化硫(SO2)']=data_list[13]
    data_dict['常量']=cl
    data=pd.DataFrame(data_dict)#将字典转换成为数据框
    pd.DataFrame(data_dict).to_excel('多层感知网络权
```

```
重.xlsx',sheet_name='data',index=False)

    y_score =mlp.predict_proba(x_test)
    from sklearn import metrics
    from pylab import *
    import matplotlib.pyplot as plt
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure()
    name=["高钾","铅钡"]
    for i in range(2):
        fpr, tpr, thresholds = metrics.roc_curve(y_test, y_score[:,
i],pos_label=i)
        auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr,
                 lw=2, label="多层感知网络_ROC "+name[i]+" (area
= %0.2f)" % auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

随机森林.py

```
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    import pandas as pd
    import warnings
    warnings.filterwarnings('ignore')
    lis=['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)',
        '风化数字化']
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='类型数字化'):
                datay.append(data['类型数字化'][i])
            elif(j in lis):
                datax[i].append(data[j][i])
    x_train, x_test, y_train, y_test = train_test_split(datax, datay,
test_size=0.2, random_state=10)
    clf = RandomForestClassifier()
    clf.fit(x_train,y_train)
    print("预测准确度：")
    print(clf.score(x_test,y_test))
    pre=clf.predict_proba(datax)
    pre_dict={}
    pre_dict['高钾概率']=[]
    pre_dict['铅钡概率']=[]
    pre_dict['预测结果']=clf.predict(datax)
    pre_dict['原始标签']=datay
    for i in range(len(pre)):
        pre_dict['高钾概率'].append(pre[i][0])
        pre_dict['铅钡概率'].append(pre[i][1])
    data=pd.DataFrame(pre_dict)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('随机森林结果概
率.xlsx',sheet_name='data',index=0)

    data=pd.DataFrame(clf.feature_importances_)
    data_dict={}
    count=0
    for i in lis:
        data_dict[i]=[]
        data_dict[i].append(clf.feature_importances_[count])
        count+=1
    data=pd.DataFrame(data_dict)#将字典转换成为数据框
    pd.DataFrame(data_dict).to_excel('随机森林权
重.xlsx',sheet_name='data',index=False)

    y_score =clf.predict_proba(x_test)
```

```
    from sklearn import metrics
    from pylab import *
    import matplotlib.pyplot as plt
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure()
    name=["高钾","铅钡"]
    for i in range(2):
        fpr, tpr, thresholds = metrics.roc_curve(y_test, y_score[:,
i],pos_label=i)
        auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr,
                 lw=2, label="随机森林_ROC "+name[i]+" (area = %0.2f)" %
auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

逻辑回归区间预测.py

```
    import pandas as pd
    #spss逻辑回归系数
    coef=[-0.333,-0.943,-0.760,-0.541,-1.679,-0.266,-0.516,-0.068,-
0.127,33.512]
    lis=['二氧化硅(SiO2)',
         '氧化钠(Na2O)',
         '氧化钾(K2O)',
         '氧化铝(Al2O3)',
         '氧化铁(Fe2O3)',
         '氧化铅(PbO)',
         '氧化钡(BaO)',
         '五氧化二磷(P2O5)',
         '二氧化硫(SO2)',
         ]
    def Logistic(lis1):
        index=[0,1,2,5,6,8,9,10,13]
        y=0
        for i in range(len(index)):
            y+=lis1[index[i]]*coef[i]
        y+=coef[len(coef)-1]
```

```
        return y
    lis2=['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)',
         '类型数字化']
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='风化数字化'):
                datay.append(data['风化数字化'][i])
            elif(j in lis2):
                datax[i].append(data[j][i])
    y=[]
    for i in range(len(datax)):
        if(Logistic(datax[i])<0.5):
            y.append(0)
        else:
            y.append(1)
    count=0
    for i in range(len(y)):
        if(y[i]==datay[i]):
            count+=1
    print("逻辑回归预测正确率：")
    print(count/len(y))

    ##############风化数据预测##############################
    lis=['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)',
        '类型数字化']
Forms=pd.DataFrame(pd.read_excel("风化.xlsx",sheet_name=0))
data=Forms.to_dict('dict')
datafh=[[] for j in range(len(Forms))]
for i in range(len(Forms)):
    for j in data:
        if(j in lis):
            datafh[i].append(data[j][i])
print(datafh)
def qjian(lis):
    flag2 = 999
    index = [0, 1, 2, 5, 6, 8, 9, 10, 13]
    lis2 = [lis[i] for i in range(len(lis))]
    n = 600
    lis1 = [lis[i] for i in range(len(lis))]
    while (n > 0):
        n-=1
        for i in range(len(index)):
            lis1[index[i]] -= lis1[index[i]]*coef[i]/10 #按系数递增或递减
        lis3 = []
        for j in range(len(lis)-1):
            lis3.append(lis1[j]/(sum(lis1)-lis1[len(lis1)-1])*100)
        lis3.append(lis1[len(lis1)-1])
        flag1=Logistic(lis3)
        if(flag1 < 0.05):
            lis2=lis3
            break
    return lis2
print(data)
datax={}
datax['文物采样点']=data['文物采样点']
for i in range(len(lis)-1):
    datax[lis[i]]={}
datax['逻辑回归结果值']={}
```

```
    for i in range(len(datafh)):
        print("第%d次\n"%(i+1))
        print(Logistic(datafh[i]))
        lis1=qjian(datafh[i])
        for j in range(len(lis1)-1):
            datax[lis[j]][i]=lis1[j]
        datax['逻辑回归结果值'][i]=Logistic(lis1)
        print(lis1)
        print(Logistic(lis1))
    data = pd.DataFrame(datax)  # 将字典转换成为数据框
    pd.DataFrame(datax).to_excel('逻辑回归预测风化前化学成分.xlsx',
sheet_name='data', index=False)
```

皮尔逊相关系数.py

```
    import pandas as pd
    from pylab import *
    import matplotlib.pyplot as plt
    import seaborn as sb

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

    lis=['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)']

    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))

    from scipy.stats import stats
    correlation_lis= {}
    for i in lis:
        correlation_lis[i]=[]
        for j in lis:
```

```
            a = pd.Series(Forms[i].to_list())
            b = pd.Series(Forms[j].to_list())
            r,p=stats.pearsonr(a,b)
            r=float(r)
            correlation_lis[i].append(r)
    data=pd.DataFrame(correlation_lis,index =lis)#将字典转换成为数据框
    pd.DataFrame(data).to_excel('皮尔逊相关系
数.xlsx',sheet_name='data',index=lis)
    print(data)
    sb.heatmap(data = data,cmap="YlGnBu")#cmap设置色系
    plt.show()
```

KMeans.py

```
    from sklearn.cluster import KMeans
    import pandas as pd
    from pylab import *
    import matplotlib.pyplot as plt
    import numpy as np
    bys_p=pd.DataFrame(pd.read_excel("四算法结果概率.xlsx",sheet_name=0))#
贝叶斯结果概率
    CNN_p=pd.DataFrame(pd.read_excel("四算法结果概率.xlsx",sheet_name=2))#
多层神经网络概率
    ran_p=pd.DataFrame(pd.read_excel("四算法结果概率.xlsx",sheet_name=3))#
随机森林概率
    sum_p=(bys_p+CNN_p+ran_p)/3
    pre_dict={}
    pre_dict['高钾']=[]
    pre_dict['铅钡']=[]
    count=0
    for i in bys_p['预测结果']:
        if(i):
            pre_dict['铅钡'].append(sum_p.loc[count, '铅钡概率'])
        else:
            pre_dict['高钾'].append(sum_p.loc[count, '高钾概率'])
        count+=1

    datax=[]
    datay=[]
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    Na2O=[]
    P2O5=[]
    SO2 =[]
    CuO=[]
    k_dict={}
```

```
k_dict['文物编号']=[]
k_dict['文物采样点']=[]
Pb_dict={}
Pb_dict['文物编号']=[]
Pb_dict['文物采样点']=[]
for i in range(len(bys_p)):
    if(bys_p['预测结果'][i]):
        Pb_dict['文物编号'].append(Forms.loc[i,'文物编号'])
        Pb_dict['文物采样点'].append(Forms.loc[i, '文物采样点'])
        P2O5.append(Forms.loc[i,'五氧化二磷(P2O5)'])
        SO2.append(Forms.loc[i,'二氧化硫(SO2)'])
        CuO.append(Forms.loc[i,'氧化铜(CuO)'])
    else:
        k_dict['文物编号'].append(Forms.loc[i,'文物编号'])
        k_dict['文物采样点'].append(Forms.loc[i, '文物采样点'])
        Na2O.append(Forms.loc[i,'氧化钠(Na2O)'])
#数据标准化
def data_stand(sr):
    sc=[]
    for i in range(len(sr)):
        sc.append((sr[i] - min(sr)) / (max(sr) - min(sr)))
    return sc

print(min(Na2O))
print(max(Na2O))
print(min(CuO))
print(max(CuO))
Na2O=data_stand(Na2O)
P2O5=data_stand(P2O5)
P2O5_SO2=[]
for i in range(len(P2O5)):
    P2O5_SO2.append(P2O5[i]+SO2[i])
print(min(P2O5_SO2))
print(max(P2O5_SO2))
P2O5_SO2=data_stand(P2O5_SO2)

K_Na=[]
for i in range(len(Na2O)):
    K_Na.append([pre_dict['高钾'][i],Na2O[i]])

Pb_sp=[]
for i in range(len(P2O5_SO2)):
    Pb_sp.append([pre_dict['铅钡'][i],P2O5_SO2[i]])
```

```
    Pb_Cu=[]
    for i in range(len(P2O5_SO2)):
        Pb_Cu.append([pre_dict['铅钡'][i],CuO[i]])

    ##########################################################################
##钾玻璃-
Na2O##########################################################################
###
    estimator = KMeans(n_clusters=2)#聚类分类数
    estimator.fit(K_Na)#聚类
    label_pred = estimator.labels_
    print("聚类标签:")
    print(label_pred)
    centroids = estimator.cluster_centers_
    print("聚类中心:")
    print(centroids)

    k_dict['亚分类']=label_pred
    data=pd.DataFrame(k_dict)#将字典转换成为数据框
    pd.DataFrame(k_dict).to_excel('钾玻璃亚分
类.xlsx',sheet_name='data',index=False)

    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.style.use('ggplot')  # 美化

    color = ["#6A5ACD", '#228B22', '#B8860B', '#B22222', '#FF69B4',
            '#1E90FF', '#4B0082', "#00FF7F", '#FFFACD', '#0000FF']

    K_Na=np.array(K_Na)
    flag0=0
    flag1=0
    x_z=(centroids[0][0]+centroids[1][0])/2
    y_z=(centroids[0][1]+centroids[1][1])/2
    #plt.axvline(x_z,c='#A19B9B')#竖线
    plt.axhline(centroids[0][1],c="#6A5ACD", linestyle='--')  # 虚线
    plt.axhline(centroids[1][1],c='#228B22', linestyle='--')  # 虚线
    plt.axhline(y_z,c='#A19B9B')#横线
    plt.axhline(y_z+y_z/10,c='#A19B9B', linestyle='--')  # 虚线
    plt.axhline(y_z-y_z/10,c='#A19B9B', linestyle='--')  # 虚线
    x_min=1
    for i in range(len(label_pred)):
        x_min=min(x_min,K_Na[i][0])
        if(flag0==0 and label_pred[i]==0):
```

```
            plt.scatter(K_Na[i][0],K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第一类")
            flag0=1
        elif(flag1==0 and label_pred[i]==1):
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第二类")
            flag1 = 1
        else:
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30)

    plt.scatter(centroids[0][0],centroids[0][1], c=color[0], s=150,
label=f"聚点",marker ='*')
    plt.scatter(centroids[1][0],centroids[1][1], c=color[1],
s=150,label=f"聚点" , marker ='*')

    plt.title('钾玻璃亚分类聚类图',fontsize=12,c='#322424')
    plt.text(x_min-0.0063,y_z,"划分线
("+str(format(y_z,'.3f'))+")",verticalalignment='bottom',horizontalalig
nment='left')
    plt.text(x_min-0.0063,centroids[0][1],"聚点线
("+str(format(centroids[0][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.text(x_min-0.0063,centroids[1][1],"聚点线
("+str(format(centroids[1][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.ylabel("Na2O含量归一化值",fontsize=12)
    plt.xlabel("类型概率均值",fontsize=12)
    plt.legend(loc=0, ncol=2,)  # 加上图列
    plt.show()

    ##################################################################铅钡-磷
硫
##########################################################################
##
    estimator = KMeans(n_clusters=2)#聚类分类数
    estimator.fit(Pb_sp)#聚类
    label_pred = estimator.labels_
    print("聚类标签:")
    print(label_pred)
    centroids = estimator.cluster_centers_
    print("聚类中心:")
    print(centroids)
```

```
    Pb_dict['亚分类']=label_pred
    data=pd.DataFrame(Pb_dict)#将字典转换成为数据框
    pd.DataFrame(Pb_dict).to_excel('铅钡-磷硫亚分
类.xlsx',sheet_name='data',index=False)


    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.style.use('ggplot')  # 美化

    color = ["#6A5ACD", '#228B22', '#B8860B', '#B22222', '#FF69B4',
            '#1E90FF', '#4B0082', "#00FF7F", '#FFFACD', '#0000FF']

    K_Na=np.array(Pb_sp)
    flag0=0
    flag1=0
    x_z=(centroids[0][0]+centroids[1][0])/2
    y_z=(centroids[0][1]+centroids[1][1])/2
    #plt.axvline(x_z,c='#A19B9B')#竖线
    plt.axhline(centroids[0][1],c="#6A5ACD", linestyle='--')  # 虚线
    plt.axhline(centroids[1][1],c='#228B22', linestyle='--')  # 虚线
    plt.axhline(y_z,c='#A19B9B')#横线
    plt.axhline(y_z+y_z/10,c='#A19B9B', linestyle='--')  # 向上10%
    plt.axhline(y_z-y_z/10,c='#A19B9B', linestyle='--')  # 向下10%
    x_min=1
    for i in range(len(label_pred)):
        x_min=min(x_min,K_Na[i][0])
        if(flag0==0 and label_pred[i]==0):
            plt.scatter(K_Na[i][0],K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第一类")
            flag0=1
        elif(flag1==0 and label_pred[i]==1):
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第二类")
            flag1 = 1
        else:
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30)

    plt.scatter(centroids[0][0],centroids[0][1], c=color[0], s=150,
label=f"聚点",marker ='*')
    plt.scatter(centroids[1][0],centroids[1][1], c=color[1],
s=150,label=f"聚点" , marker ='*')
```

```
    plt.title('铅钡玻璃—磷硫亚分类聚类图',fontsize=12,c='#322424')
    plt.text(x_min-0.0063,y_z,"划分线
("+str(format(y_z,'.3f'))+")",verticalalignment='bottom',horizontalalig
nment='left')
    plt.text(x_min-0.0063,centroids[0][1],"聚点线
("+str(format(centroids[0][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.text(x_min-0.0063,centroids[1][1],"聚点线
("+str(format(centroids[1][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.ylabel("P/S含量归一化值",fontsize=12)
    plt.xlabel("类型概率均值",fontsize=12)
    plt.legend(loc=0, ncol=2,)  # 加上图列
    plt.show()

    ##################################################################铅钡-铜
##########################################################################
    estimator = KMeans(n_clusters=2)#聚类分类数
    estimator.fit(Pb_Cu)#聚类
    label_pred = estimator.labels_
    print("聚类标签:")
    print(label_pred)
    centroids = estimator.cluster_centers_
    print("聚类中心:")
    print(centroids)

    Pb_dict['亚分类']=label_pred
    data=pd.DataFrame(Pb_dict)#将字典转换成为数据框
    pd.DataFrame(Pb_dict).to_excel('铅钡-铜亚分
类.xlsx',sheet_name='data',index=False)

    mpl.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.style.use('ggplot')  # 美化

    color = ["#6A5ACD", '#228B22', '#B8860B', '#B22222', '#FF69B4',
            '#1E90FF', '#4B0082', "#00FF7F", '#FFFACD', '#0000FF']

    K_Na=np.array(Pb_Cu)
    flag0=0
    flag1=0
    x_z=(centroids[0][0]+centroids[1][0])/2
    y_z=(centroids[0][1]+centroids[1][1])/2
    #plt.axvline(x_z,c='#A19B9B')#竖线
```

```
    plt.axhline(centroids[0][1],c="#6A5ACD", linestyle='--')  # 虚线
    plt.axhline(centroids[1][1],c='#228B22', linestyle='--')  # 虚线
    plt.axhline(y_z,c='#A19B9B')#横线
    plt.axhline(y_z+y_z/10,c='#A19B9B', linestyle='--')  # 虚线
    plt.axhline(y_z-y_z/10,c='#A19B9B', linestyle='--')  # 虚线
    x_min=1
    for i in range(len(label_pred)):
        x_min=min(x_min,K_Na[i][0])
        if(flag0==0 and label_pred[i]==0):
            plt.scatter(K_Na[i][0],K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第一类")
            flag0=1
        elif(flag1==0 and label_pred[i]==1):
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30, label=f"第二类")
            flag1 = 1
        else:
            plt.scatter(K_Na[i][0], K_Na[i][1], c=color[label_pred[i]],
s=30)

    plt.scatter(centroids[0][0],centroids[0][1], c=color[0], s=150,
label=f"聚点",marker ='*')
    plt.scatter(centroids[1][0],centroids[1][1], c=color[1],
s=150,label=f"聚点" , marker ='*')

    plt.title('铅钡玻璃—铜亚分类聚类图',fontsize=12,c='#322424')
    plt.text(x_min-0.0063,y_z,"划分线
("+str(format(y_z,'.3f'))+")",verticalalignment='bottom',horizontalalig
nment='left')
    plt.text(x_min-0.0063,centroids[0][1],"聚点线
("+str(format(centroids[0][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.text(x_min-0.0063,centroids[1][1],"聚点线
("+str(format(centroids[1][1],'.3f'))+")",verticalalignment='bottom',ho
rizontalalignment='left')
    plt.ylabel("CuO含量归一化值",fontsize=12)
    plt.xlabel("类型概率均值",fontsize=12)
    plt.legend(loc=0, ncol=2,)  # 加上图列
    plt.show()
```

三大算法预测文物类型.py

```
    import pandas as pd
    import math
    import warnings
```

```
    from matplotlib import pyplot as plt

    warnings.filterwarnings('ignore')
    #表单三
    Forms_three=pd.DataFrame(pd.read_excel("附件.xlsx",sheet_name=2))
    chemical_composition=Forms_three.to_dict('dict')
    for i in range(len(Forms_three)):
        count_sum=0
        nan_number=0
        nan_index=[]
        for s in Forms_three:
            if(s !='文物编号' and s!= '表面风化' and
math.isnan(Forms_three.loc[i,s])):
                nan_number+=1
                nan_index.append(s)
            elif(s !='文物编号' and s!= '表面风化'):
                count_sum+=Forms_three.loc[i,s]
        count_sum=100-count_sum
        for s in nan_index:
            chemical_composition[s][i]=count_sum/nan_number
    data=pd.DataFrame(chemical_composition)#将字典转换成为数据框
    try:
        pd.DataFrame(data).to_excel('处理后的表单三数
据.xlsx',sheet_name='data',index=False)
    except:
        print("文件已打开！")
    yuce_data=chemical_composition
    chemical_composition=Forms_three.to_dict('dict')
    yuce_data.pop('文物编号')
    yuce_data['风化数字化']={}
    for i in range(len(yuce_data['表面风化'])):
        if(yuce_data['表面风化']=='风化'):
            yuce_data['风化数字化'][i]=1
        else:
            yuce_data['风化数字化'][i] = 0
    yuce_data.pop('表面风化')
    yuce_datax=[[] for j in range(len(Forms_three))]
    for i in range(len(Forms_three)):
        for j in yuce_data:
            yuce_datax[i].append(yuce_data[j][i])
    print(yuce_datax)
    #波动测试使用
    '''import numpy as np
```

```
    yuce=np.array(yuce_datax)
    import random
    for i in range(len(yuce)):
        for j in range(len(yuce[i])):
            yuce+=yuce*random.randint(-100,100)*0.001 #-100,100 表示上下10%
的波动
    #yuce=yuce+yuce*0.4 #整体上下波动
    #yuce=yuce-yuce*0.4
    yuce_datax=yuce.tolist()'''
    '''无波动结果：
    贝叶斯预测结果：
    [0 1 1 1 1 0 0 1]
    神经网络预测结果：
    [0 1 1 1 1 0 0 1]
    随机森林预测结果：
    [0 1 1 1 1 0 0 1]'''
    pre_pro=[[0 for i in range(8)] for j in range(3)]
    ##########################################################贝叶斯预测
##########################################################
    from sklearn.naive_bayes import MultinomialNB
    import pandas as pd
    Forms=pd.DataFrame(pd.read_excel("datafenghua.xlsx",sheet_name=0))
    lis=['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)',
         '风化数字化']
    data=Forms.to_dict('dict')
    datax=[[] for j in range(len(Forms))]
    datay=[]
    for i in range(len(Forms)):
        for j in data:
            if(j=='类型数字化'):
                datay.append(data['类型数字化'][i])
```

```
            elif (j in lis):
                datax[i].append(data[j][i])
    clf = MultinomialNB()
    clf.fit(datax,datay)
    print("贝叶斯预测结果：")
    print(clf.predict(yuce_datax))
    pre=clf.predict(yuce_datax)
    pro=clf.predict_proba(yuce_datax)
    for i in range(len(pre)):
        pre_pro[0][i]=pro[i][pre[i]]
    chemical_composition['贝叶斯预测结果']= {}
    for i in range(len(pre)):
        if(pre[i]):
            chemical_composition['贝叶斯预测结果'][i]='铅钡'
        else:
            chemical_composition['贝叶斯预测结果'][i] = '高钾'
    ##############################################################神经网络预测
##########################################
    from sklearn import neural_network
    mlp=neural_network.MLPClassifier(hidden_layer_sizes=(20), #隐藏层
                                     activation='relu',  #激活函数
                                     solver='adam',
                                     alpha=0.0001,  #正则化项系数
                                     batch_size='auto',
                                     learning_rate='constant',  #学习率
                                     learning_rate_init=0.001,
                                     power_t=0.5,
                                     max_iter=300,#迭代次数
                                     tol=1e-4)
    mlp.fit(datax,datay)
    print("神经网络预测结果：")
    print(mlp.predict(yuce_datax))
    pre=mlp.predict(yuce_datax)
    pro=mlp.predict_proba(yuce_datax)
    for i in range(len(pre)):
        pre_pro[1][i]=pro[i][pre[i]]
    chemical_composition['神经网络预测结果']= {}
    for i in range(len(pre)):
        if (pre[i]):
            chemical_composition['神经网络预测结果'][i] = '铅钡'
        else:
            chemical_composition['神经网络预测结果'][i] = '高钾'
    ##################################################随机森林
##############################################
```

```
    from sklearn.ensemble import RandomForestClassifier
    clf = RandomForestClassifier()
    clf.fit(datax,datay)
    print("随机森林预测结果：")
    print(clf.predict(yuce_datax))
    pre=clf.predict(yuce_datax)
    pro=clf.predict_proba(yuce_datax)
    for i in range(len(pre)):
        pre_pro[2][i]=pro[i][pre[i]]
    chemical_composition['随机森林预测结果']= {}
    for i in range(len(pre)):
        if (pre[i]):
            chemical_composition['随机森林预测结果'][i] = '铅钡'
        else:
            chemical_composition['随机森林预测结果'][i] = '高钾'
    data=pd.DataFrame(chemical_composition)#将字典转换成为数据框
    try:
        pd.DataFrame(data).to_excel('三算法预测文物结
果.xlsx',sheet_name='data',index=False)
    except:
        print("文件已打开！")

    # 设置X,Y的范围
    x =[]
    for i in range(8):
        x.append("文物"+str(i+1))
    y_1 =pre_pro[0]
    y_2 =pre_pro[1]
    y_3 =pre_pro[2]
    # 设置图形大小
    plt.rcParams['font.sans-serif'] =["KaiTi"]
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure(figsize=(15,20), dpi=120)

    plt.plot(x, y_1, label='贝叶斯分类', color='red', linestyle=':',
marker='.', markersize=20)
    plt.plot(x, y_2, label='神经网络', color='black', linestyle='--',
marker='.', markersize=20)
    plt.plot(x, y_3, label='随机森林', color='#228B22', linestyle='--',
marker='.', markersize=20)
    # 设置x刻度
    _xtick_labels = [i for i in x]
    plt.xticks(x, _xtick_labels, rotation=45,fontsize=20)
    plt.yticks(fontsize=20)
```

```
    # 设置X，Y轴标签
    #plt.xlabel('预测结果权重',fontsize=20)
    plt.ylabel('预测结果权重',fontsize=20)
    plt.title('三个算法预测文物结果权重',fontsize=20)
    plt.grid(alpha=0.8)
    plt.legend(loc='upper left')
    # 展示图形
    plt.show()
```