古代玻璃制品的成分分析与鉴别问题

摘要

本文对古代玻璃制品的成分分析与鉴别的模型问题进行研究。

在解决问题之前，我们先进行了数据清洗，检测到2个无效数据并进行了剔除。

针对问题一，我们采用了卡方检测算法，分析了玻璃特征与表面风化之间的关联性，得到影响玻璃风化的因素是**纹饰与类型**，其中易风化的玻璃特征组合为C纹饰-铅钡类型，B纹饰类型，铅钡类型和不易风化的玻璃特征组合为C纹饰-高钾，A纹饰-高钾类型。为了验证我们算法的合理性我们采用了*Apriori*算法进行分析，其结果与卡方模型结果基本一致，并且与现实情况相符。通过绘制风化前后的化学元素分布散点图，我们**定性得出了其风化前后两种玻璃的化合物含量变化趋势**，如高钾玻璃风化后二氧化硅含量会增加。最后，通过对各种化学元素进行正态化拟合，得出各个化学元素的正态分布规律，再对其中不符合正态分布规律的元素进行了修正，获取置信区间，分析化合物特征。**通过***Mapie* **算法**，对风化后的每种化合物，进行分析，获取风化前的区间预测值。

针对问题二，我们再次利用卡方检验寻找玻璃类型与化合物含量的关联性，并得出了鉴别高钾和铅钡两种玻璃类型的关键化合物是**氧化钙、氧化钡、氧化铅与氧化钾**。为了对玻璃类型亚分类，我们采用了系统聚类法进行聚类分析，聚类后，高钾玻璃亚分为三种类型，其分类依据取决于**氧化钙**和**氧化铝**含量的多少；铅钡玻璃亚分为三种类型，其分类依据取决于**氧化钠**和**氧化铜**含量的多少。为了检验模型的合理性，我们绘制了箱型图,同时对模型进行了**五折交叉验证评估**。不同亚类箱型图具有很强的规律性，**模型的评估得分为0.94**，准确度很高。为了分析模型敏感性，我们加入了**随机高斯噪音**，但对最后的分类结果没有影响，模型稳定性高。

针对问题三，我们做了模型融合，将原本的聚类问题转化为更易拟合有监督的分类问题。对数据处理后，我们分别在*lightgbm*,*svm*,*catboost*分类算法中进行了处理，比较采用了*catboost*算法，最后利用**卷积神经网络***CNN*进行深度学习，模型评价得分为**89.89372，**对其加入高斯噪声，结果具有稳定性，以此我们鉴定出了未知的文物玻璃类型。

针对问题四，我们采取了以**相关系数算法**为核心，通过相关性热图进行定性分析，得到了不同玻璃类型其化学成分含量间的关联性，并且我们利用*MVTEST*库进行独立性检验，得到了正负相关性。最后通过可视化的方式，观察他们的具体关联，并对找到的关联规则进行了分析。

。

关键字： 卡方检验　卷积神经网络　Mapie区间预测算法 系统聚类算法　catboost算法

目录

一、 问题背景与重述

党的十八大以来，“考古中国”这一项中华文明探源工程成果丰硕，我国已先后完成两次全国文物普查，实施了丝绸之路、长城等万余项重点文物保护工程。其中，考古人员需要挖掘文物背后的交织脉络，让我们对自身血脉与文化根基极深研几的追问，重塑中华民族的文化自信。在这其中，丝绸之路作为我国早期中西方文化交流的通道，其出土的玻璃文物是早期贸易往来的宝贵物证。然而由于年代的久远，玻璃的风化导致玻璃的化学元素占比发生了改变，考古人员难以通过已有的判断条件鉴定文物玻璃类型。

以本题为例，出土的玻璃有两种，一种是高钾玻璃，另一种是铅钡玻璃。由于风化的原因，其颜色、纹饰，尤其是化学成分，会发生显著的变化。

在这种情况下，我们需要对风化前与风化后的文物表征后的数据，建立一种玻璃文物的数学模型，通过我们的模型，对出土玻璃的种类进行鉴别，并给出其化学成分的关联性。

题目要求我们针对下面的具体情况：

1. 给出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已将其分为高钾玻璃和铅钡玻璃两种类型；
2. 给出古代玻璃制品的分类信息；
3. 给出这一批玻璃样品的成分组成。

完成以下任务：

任务 1：分析风化对玻璃特征的影响，并预测风化前的玻璃特征。

任务 2：对一般问题进行研究，给出高钾玻璃和铅钡玻璃的判断模型和相应分类方法；

任务 3：利用我们建立的模型对附件表单3中的未知类别的玻璃文物的化学成分进行分析，鉴别其所属种类。

任务 4：分析不同类别其化学成分含量之间的关联性和差异性，并给出分析。

二、 模型的假设

1. 假设玻璃文物在风化后化学含量的变化，都是由风化这个因素引起的。
2. 假设化学成分的占比为质量占比。
3. 在考古人员对玻璃类型进行分类时，其分类不存在错误。
4. 假设采样点足够随机，具有代表性
5. 假设采样仪器没有收到外界干扰，即采集的元素与玻璃实际元素一致
6. 假设未风化点，为该玻璃初始元素含量

三、 符号约定

符号	意义
H_0	卡方检验事件
X_i	玻璃特征组合
A_i	特征区间
f_i	特征频率
p_i	特征概率
χ^2	卡方检验统计量
P	置信度
H	聚类簇
K	聚类簇
D	聚类时样本点之间的距离

四、 问题分析

4.1 问题一分析

对于问题（1），我们需要对其的特征进行差异性检验，由于其特征均为离散型，我们可以采用卡方检验进行处理，并利用p值判断玻璃特征组合是否显著影响风化。

对于问题（2），为了能够定性的得到玻璃风化前后的统计规律，我们可以对数据进行散点图处理，并将他们的变化进行可视化处理，来分析其风化前后的化学成分含量的变化。

对于问题（3），由于数据样本量小、影响因素多、限制条件小，所以我们并不采取进行固定值的预测，这样的预测值往往是不准确的，我们认为，其化合物风化前的含量的值应该位于某置信一区间内，这样的区间预测方式更具有实际意义，所以我们采取$mapie$区间预测模型进行解决。

4.2 问题二分析

问题二中问题（1）要求我们对分析高钾玻璃和铅钡玻璃的分类依据。这要求我们需要分析化学成分含量与玻璃类型的关联性，我们将连续变量进行等距分箱，并进行卡方分析，来得到他们之间的关联性，并以此判断高钾玻璃和铅钡玻璃的分类主要取决于哪些化学成分含量。

在问题（2）中，题目要求我们对高钾玻璃和铅钡玻璃进行亚分类，通过观察数据特点，我们认为利用K-means聚类算法可以很好的对这两种玻璃进行再一次的亚分类，但由于传统的K-means算法不稳定性，其聚类效果并不如系统聚类法，所以我们利用系统聚类算法建立分类模型。

为了检验模型的合理性与灵敏性，我们可以分别模型进行五折交叉分析验证和添加高斯噪声，来进行合理性和灵敏度的分析。

4.3问题三分析

第三问要求我们鉴别未知类别玻璃文物的类型，我们采取的方法是先对其进行大类分类，在大类基础上进行亚类分类。

此题我们有三部分预测与拟合：根据未分化文物的数据预测基本类型，根据已分化数据预测其分化前的数据，最终选择卷积神经网络综合前两部分结果作为特征预测亚类，涉及到模型融合部分。

4.4 问题四分析

第四问要求我们寻找玻璃化学成分的关联关系，以及比较不同类别之间的化学成分关联关系的差异性，为了能直观的反应关联关系，我们利用$corr$函数采用皮尔逊方法求相关性,并通过热图的形式表示出来。但由于热图只能表示有贡献的数据，不能完整地表示所有相关联关系，因此，我们又引入独立性检验，通过调用$mvtest$库，对每组对象进行全面的相关性分析及差异性分析。

五、 模型的建立与分析

5.1 数据预处理

5.1.1数据清洗

由于检测手段等原因，采样点的化学成分比例的累加和会出现非100%的情况，在该问题中，题目认为成分比例累加和在85%~105%之间的算有效数据，所以我们需要对无效数据进行剔除。

对各采样点数据进行可视化处理：

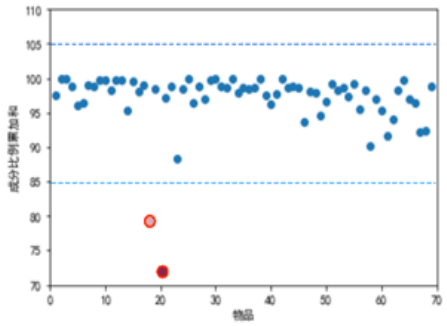


图5.1 化学成分累计和分布图

图中纵轴表示各采样点的化学成分比例累加和，横轴表示各采样点的编号。可以看出所有采样点中存在两个无效数据，对应表单2中的17与19采样点，我们对其进行了剔除。

5.1.2数据自然数编码

对数据中出现的纹饰、颜色和风化等符号数据进行编码，方便模型的分析。词典位于附录代码部分

5.2 问题一模型的建立与分析

5.2.1卡方检验

卡方检验就是统计样本的实际观测值与理论推断值之间的偏离程度，其检验步骤如下：

提出假设H_0：玻璃特征组合X与风化Y有关,其玻璃特征总体X的分布律P为

$$P\{X=x_i\}=p_{i,i=1,2,\ldots} \tag{1}$$

将玻璃特征总体X的取值范围分成k个互不相交的小区间；

当H_0为真时，根据所假设的理论分布，可算出总体X的值落入第i个小区间A_i的样本值的理论频数。n次实验中样本值落入第i个小区间A_i的概率f_i/n与概率p_i很接近，当H_0不为真时，则相差很大，所以我们可以得到检验统计量：

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_i - np_i)^2}{np_i} \tag{2}$$

若χ^2值越大，则说明玻璃特征组合X与风化Y有关系的可能性越大。其中，玻璃特征的组合类型有6种。我们将其带入卡方检验模型，其χ^2值与P值如下

表5.1 玻璃特征组合与风化关联性的卡方检验结果

玻璃特征组合	χ^2	$p-value$
纹饰	1.14201269	0.28522744
类型	2.13529412	0.14394285
颜色	0.01210826	0.9123796
纹饰-类型	5.49575394	0.01906271
纹饰-颜色	0.0266301	0.87037096
类型-颜色	0.24677679	0.61935384

其中的玻璃特征组合中，颜色、纹饰-颜色与类型-颜色的P值均大于0.5，**可以认为其与玻璃是否风化没有显著性，而与玻璃是否风化具有显著性的玻璃特征为:纹饰-类型组合。同时也可以得到玻璃的纹饰,类型也能显著的影响玻璃是否风化，但是纹饰-类型组合与玻璃是否风化的关联性最大**。

为了更深入分析这三个组合如何影响风化，我们分析了这三个组合的特征-风化柱状图。

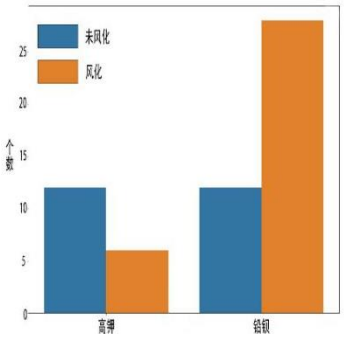


图5.2玻璃类型风化柱状图

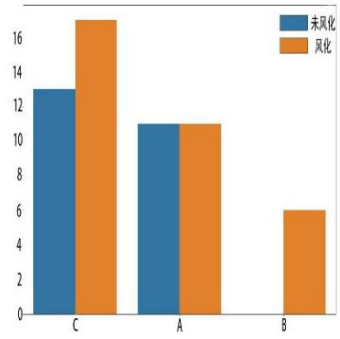


图5.3纹饰风化柱状图

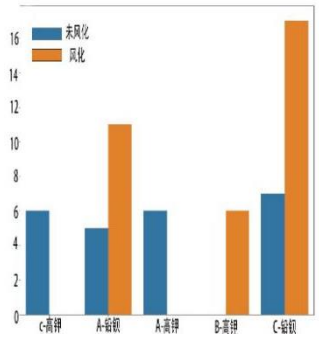


图5.4 纹饰-类型风化柱状图

可以看出其风化分布具有明显的规律，**其中易发生风化玻璃特征组合为：C纹饰-铅钡类型，B纹饰类型，铅钡类型。其中不易风化的玻璃特征组合为：C纹饰-高钾类型和A纹饰-高钾类型。**

5.2.2 Apriori算法分析

通过$Apriori$算法，找到数据中的频繁项集以及关联规则，通过分析物品各要素的出现频率和相关性，以下特征要素与表面风化同时出现的频率高，说明这种组合特征导致表面风化的可能性大

算法采用最小支持度为0.4，结果如下：

表5.2 Apriori算法分析结果

支持度	频繁项集	关联规则
0.483	铅钡，表面风化	铅钡 → 风化，风化 → 铅钡
0.30	纹饰C，纹饰类型C-铅钡，风化	纹饰C，纹饰C-铅钡 → 风化

这与我们卡方检验得到的结论一致，说明我们的关联性分析准确。

5 .2.3 统计规律分析

为了分析不同类型玻璃风化前后的化学成分比例，我们对不同玻璃的化学成分进行散点图处理，并对其前后的变化进行了定性的判断。

表5.3 风化前后化学成分变化的统计规律

风化后含量变化	化学成分	
	高钾玻璃	铅钡玻璃
增加	SiO_2	PbO
稳定	CuO, P_2O_5, SO_2	除PbO和Na_2O外的 12 种化合物
下降	除SiO_2, CuO, P_2O_5, SO_2外的 10 种化合物	Na_2O

我们对两种玻璃进行分析，将具有特征性变化的化学成分变化以散点图的形式如下给出，其中橙色表示风化的样品，蓝色表示未风化的样品，全部化学成分含量风化前后的散点图见支撑材料。

高钾玻璃：

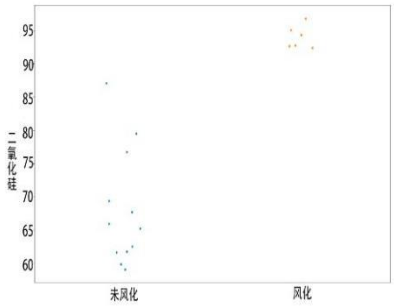


图5.5高钾玻璃二氧化硅含量风化前后散点图

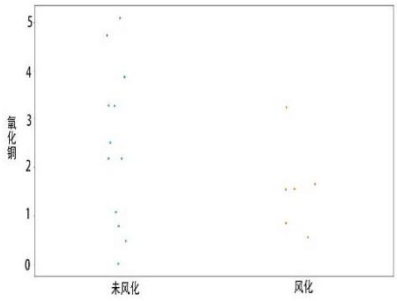


图5.6高钾玻璃氧化铜含量风化前后散点图

通过对散点图进行分析，**高钾玻璃的风化统计规律是：风化后的二氧化硅含量明显上升，并且具有较大比重，而其他化学成分均有不同程度的下降。**

铅钡玻璃：

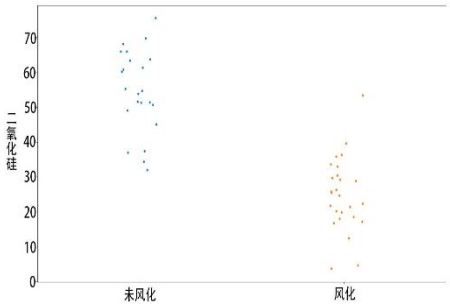


图5.7铅钡玻璃二氧化硅含量风化前后散点图

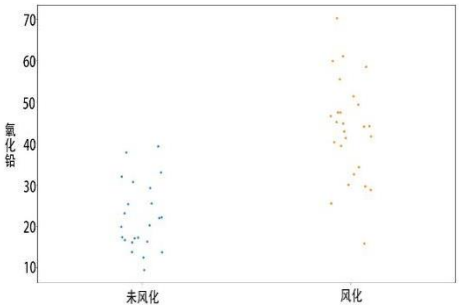


图5.8铅钡玻璃氧化铅含量风化前后散点图

通过对散点图进行分析，**铅钡玻璃的风化统计规律是：风化后二氧化硅的含量显著下降，而氧化铅的含量有所上升，其他化学成分都能保持一定的稳定性**。

5.2.4 Mapie区间预测算法进行化学成分含量预测

首先做出每一类化合物的未风化前的频数分布直方图，然后对其计算拟合分布，最后根据拟合的结果求出每一种元素的置信区间，下面给出部分化合物的置信区间图，其余图见支撑材料。

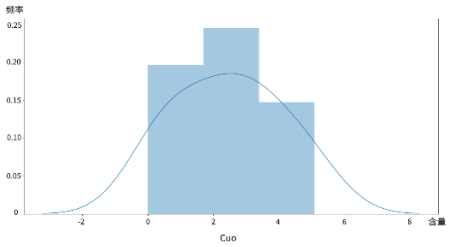


图5.9 氧化铜置信区间图

经验证，所有分布中仅有氧化钠和氧化锡无法拟合正态分布，所以单独对这两列拟合并对氧化纳和氧化锡的置信区间进行修正，以氧化钠为例，其修正前后图如下：

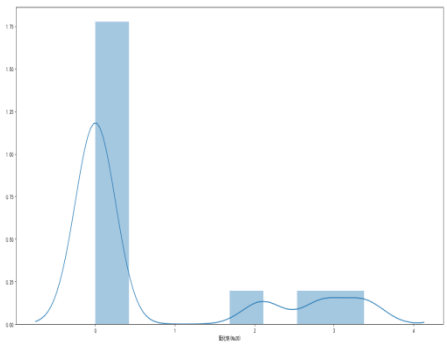


图5.10 修正前

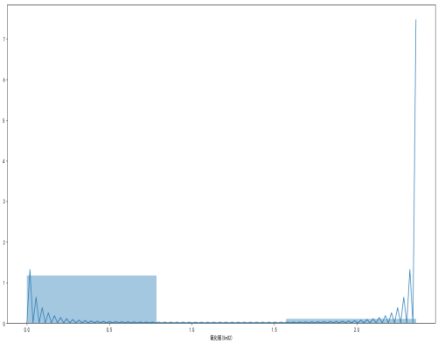
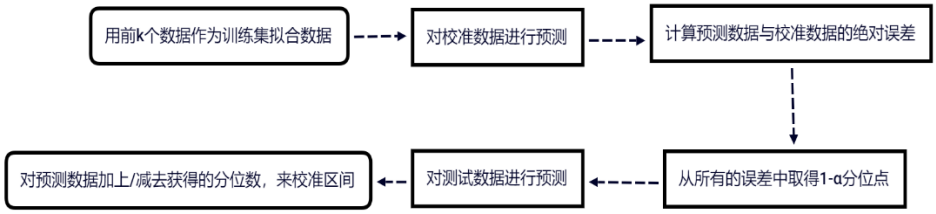


图5.11 修正后

随后利用*mapie*区间预测算法拟合占比变化率进而根据占比变化率求出风化前的化学成分含量。

*mapie*区间预测算法流程图为：



经过预测模型处理后，部分预测数据如下，化学成分风化前含量分在在预测数据的周围，并不是预测数据本身，预测数据本身为数据的平均值。

文物采样	二氧化硅	氧化钠(Na	氧化钾(K	氧化钙(Ca	氧化镁(Mg	氧化铝(Al	氧化铁(Fe	氧化铜(Cu	氧化铅(Pb	氧化钡(Ba	五氧化二	氧化锶(Sr	氧化锡(Sn	二氧化硫
2	44.2314	33.51962	0.383803	0.252603	1.580486	0.826481	2.803266	0.928344	2.589183	29.05581	8.097362	2.16979	0.272273	0
7	76.64389	0.463333	6.401667	3.845	0.785	5.056667	1.376111	2.155556	0.274444	0.398889	1.028333	0.027778	0.131111	0.067778
8	44.2314	33.51962	0.383803	0.252603	1.580486	0.826481	2.803266	0.928344	2.589183	29.05581	8.097362	2.16979	0.272273	0.119344
08严重风	33.51962	33.51962	0.383803	0.252603	1.580486	0.826481	2.803266	0.928344	2.589183	29.05581	8.097362	2.16979	0.272273	0.119344
9	76.64389	0.463333	6.401667	3.845	0.785	5.056667	1.376111	2.155556	0.274444	0.398889	1.028333	0.027778	0.131111	0.067778

5.3 问题二模型的建立与分析

5.3.1 数据预处理

首先，为了分析高钾玻璃与铅钡玻璃的分类依据，已经风化的样本数据就不再具有统计意义，所以我们将所有已风化的数据进行了剔除，然后进行分析。

然后，由于是根据其化学成分含量这个连续变量探究玻璃类型这个离散变量之间的相关性，所以我们需要对化学成分含量这个连续变量进行等距分箱处理，这样可以把数据按特定的规则进行分组，实现数据的离散化，增强数据稳定性，很好的处理空值和缺失值。

基于以上两种处理，我们对数据进行了等距分箱处理，得到其研究特征和类型特征的分箱结果图，并给出部分分箱结果图。

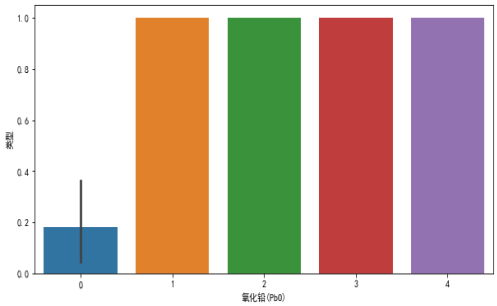


图5.12氧化铅分箱图

最后，根据柱状图去除冗余编码。以*图5.12*为例，横坐标编码为1,2,3,4起到的作用效果一致，所以我们可以将1,2,3,4归为一类，减少冗余的编码，减少分类，将连续数据转化为离散数据，方便比较，使结果更为精确。其分类后数据集见支撑材料。

5.3.2卡方分析

再次运用卡方检验，对每一种化学成分与玻璃类型进行关联性分析，得到χ^2值与p值。

表5.4 部分化学成分与玻璃类型关联性的卡方检验结果

化学成分	χ^2	$p-value$
氧化钾	2.27500000e+01	1.84504140e-06
氧化钙	2.41139601e+01	9.07996575e-07
氧化铅	1.10769231e+01	8.74087211e-04
氧化钡	8.30769231e+00	3.94775186e-03
...	...	...

结合χ^2值和p值可以看出，**氧化钾，氧化铅,氧化钡,氧化钙与玻璃类型有强相关性**。

5.3.3关联性分析

根据上述卡方分析，我们认为区分玻璃类型的关键在于氧化钾、氧化铅、氧化钡和氧化钙的化学成分含量的多少。根据这个结论，我们将这四种化学成分做出箱型图进行关联性分析。

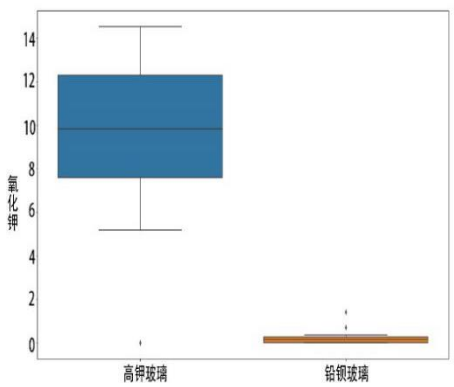


图5.13 氧化钾箱型图

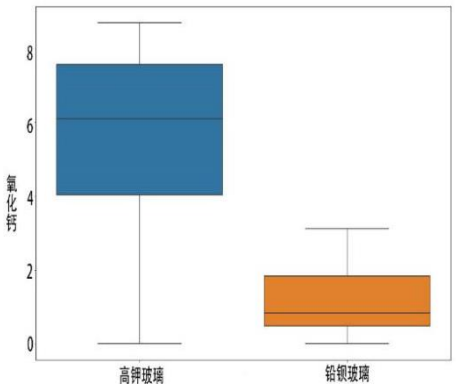


图5.14 氧化钙箱型图

如图5.13和图5.14所示，高钾玻璃类型氧化钾的含量与氧化钙含量远远高于铅钡玻璃类型，这非常符合高钾玻璃的特点。同时由于随着玻璃工艺的发展，铅钡玻璃逐渐用铅去替代高钾玻璃中的钙，所以铅钡玻璃中氧化钙的含量低于高钾玻璃也得到了很好的解释。

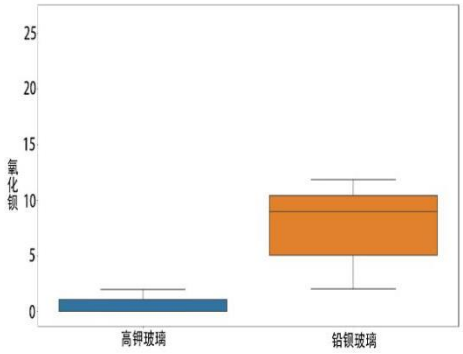


图5.15 氧化钡箱型图

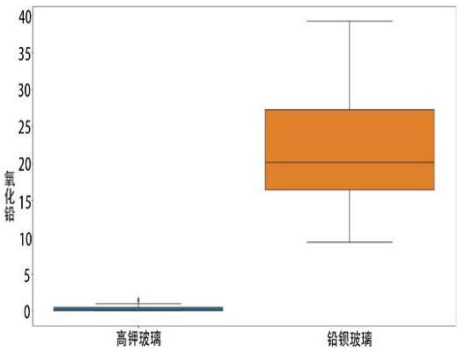


图5.16 氧化铅箱型图

可以看出在铅钡玻璃类型中，其氧化钡和氧化铅含量显著高于高钾玻璃类型，这也非常符合该类玻璃的特点。

经过上述的分析，我**们认为区分高钾玻璃和铅钡玻璃的关键在于氧化钾、氧化钡、氧化铅和氧化钙这四种化学物质。在高钾玻璃中，氧化钾和氧化钙的含量丰富；在铅钡玻璃中，则是氧化铅和氧化钡的含量丰富。**

通过箱型图，我们能很明显的得到化学成分含量的分布与两种玻璃的特点以及实际玻璃生产历史演化一一对应，这说明我们的模型具有很强的解释性，同时也可以说明我们的结论具有较高的可信度。

5.3.4 系统聚类法

数据处理

数据选择上，由于根据玻璃原始成分分类，所以我们选择**未风化的数据**作为数据集，并绘制其化学成分分布图。

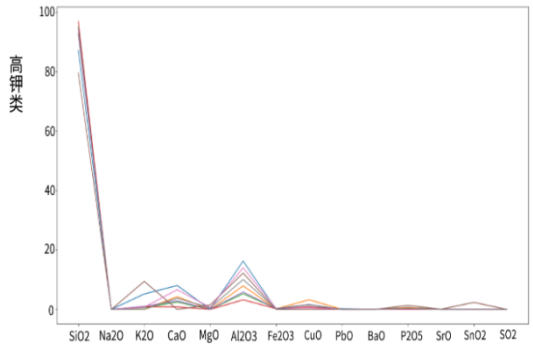


图5.17未风化高钾类样品化合物含量分布

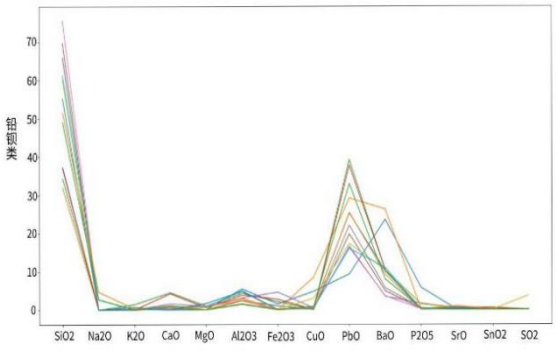


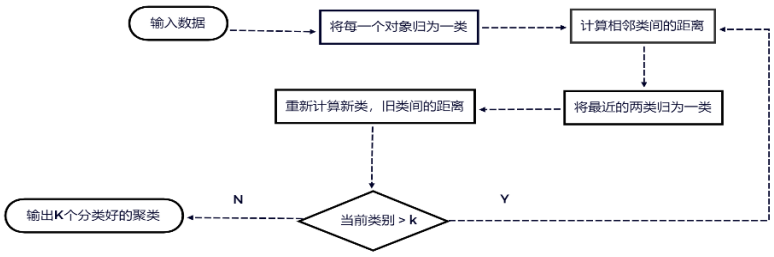
图5.18未风化铅钡类样品化合物含量分布

经上述图像分析发现，高钾类中氧化钙和氧化铝差异较大，为了提高精确度，故对全部数据进行归一化处理后，将该两列乘以权重值4，扩大其影响力，使实现效果更显著。同样的，在铅钡类中，氧化钠和氧化铜的含量差异比较大，我们将该两列乘以权重4。

模型的建立与求解

算法选择上,为了找到合适的化学成分来对高钾玻璃和铅钡玻璃进行亚分类,我们首先采用无监督学习中的K-means方法进行聚类，经过实验验证由于该算法只适用于大样本，对于数据比较少的样本而言效果并不太理想。因此我们采用系统聚类法，通过每次将相邻最近的样品分为一类，随着层次迭代加深，最终形成K类目标聚簇，经过对比发现，系统聚类法的簇内差值更小，分类效果更加明显

其算法设计流程图为：



在我们采用的系统聚类法模型中，计算两个类距离的方式采取最远距离法：

设H和K是两个聚类，则两类间的最长距离定义为：

$$D_{H,K}=\max\{d_{u,v}\},u\in H,v\in K \tag{3}$$

随后进行递推运算

$$\left.\begin{array}{l}D_{H,I}=\max\{d_{m,n}\},m\in H,n\in I\\D_{H,J}=\max\{d_{m,n}\},m\in H,n\in J\end{array}\right\}\Rightarrow D_{H,K}=\max\{D_{H,I},D_{H,J}\} \tag{4}$$

将数据集置入算法模型，我们得到以下的聚类曲线。

高钾玻璃:

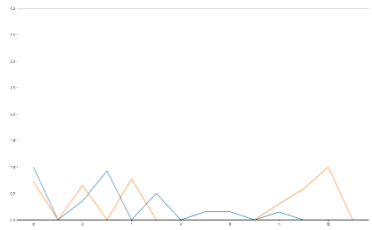


图5.19 低钙高铝型化合物含量分布

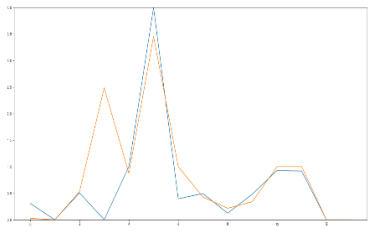


图5.20 中钙高铝型化合物含量分布

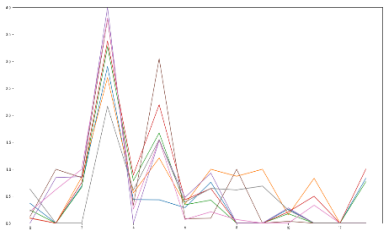


图5.21高钙低铝型化合物含量分布

图中横坐标为各化学成分的标号，纵坐标为含量。

可以看出经过聚类分析**高钾玻璃可以分成三种亚类，分别是中钙高铝型、低钙高铝型和第三种高钙低铝型，**并且分成的三类各类中，各样品化学成分分布曲线具有高度的相似性，这也充分说明高钾玻璃分成的这三种亚类是有实际的参考价值的。

铅钡玻璃：

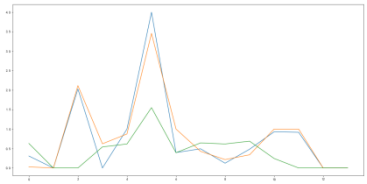


图5.22 低铜低钠类

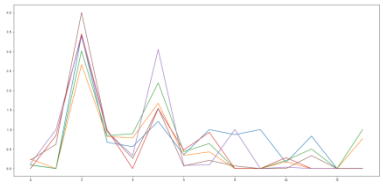


图5.23低铜中钠类

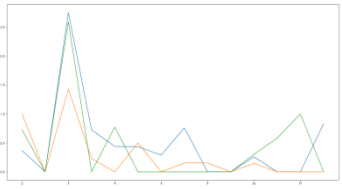


图5.24高铜低钠类

同样的，经过聚类分析，数据集被分为了**三种亚类，其分别对应高铜低钠类、低铜中钠类与低铜低钠类**，其分成的三类各类中，各样品化学成分分布曲线并不如高钾玻璃亚类分类后的相似性高，但也具有比较明显的相似性，对此我们进行箱型图分析。

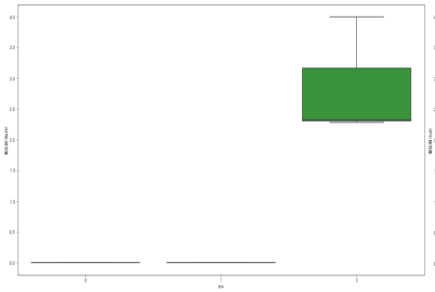


图5.25 三种分类下氧化钠的含量箱型图

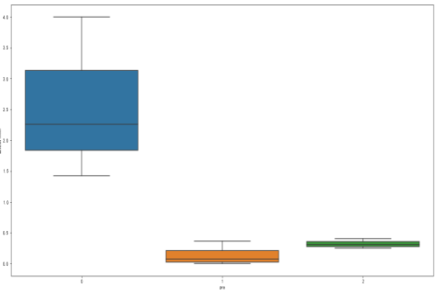


图5.26 三种分类下氧化铜的含量箱型图

可以看出这三类在氧化铜和氧化钠含量上具有明显的差异性，所以该模型的分类效果具有一定的可行性和准确性。

合理性检验：

使用SPSSPRO五折交叉验证对模型进行准确度评估，采用8:2的比例划分训练集和验证集，进行交叉验证。其**模型评价得分（*accuracy score*）为0.9411764705882353**。

其评价分数>0.8表示结果合理可信，此划分依据具有很强的可解释性，与实际上玻璃生产类别一致

敏感性检验：

通过给数据添加高斯噪声进行干扰，在多次添加随机高斯噪声的情况下，模型的预测结果始终与原结果保持一致，说明**模型抗干扰能力强，灵敏度低**。

```
In  [692]: sklearn.metrics.mean_squared_error(data_2_type0['pre'], data_2_type0['pre_y'])
Out[692]: 0.0
```

图5.27 高斯噪声干扰程序运行结果图

5.4 问题三模型的建立与分析

5.4.1 模型比较与模型优化

由于我们在问题二关于亚类的讨论中是基于大类型聚类的，所以此问题的预测我们也需要基于大类进行划分。

首先分别使用*Catboost*、*Lightgbm*和*svm*对已分类玻璃文物的化学成分比例进行拟合，并对验证集求精准率分数，最终验证得分为*Catboost*最高，故我们选择*Catboost*预测未知基本类型。

表5.4 预测算法验证得分

拟合算法	验证得分
Cathboost	1.0
Lightgbm	0.9411764705882353
svm	1.0

除此之外，由于我们上一题用于模型拟合的数据均为未风化的数据，故此题中我们利用第一题的*mapie*算法预测化学物质风化前化学成分含量的模型，对此题中已风化的玻璃求风化前的各化学成分含量，至此，我们得到了所有用于求亚类的特征。

结合上题的聚类结果，我们成功将原来的聚类分析转化成分类预测，将无监督学习变为有监督学习，模型表现力更好，拟合效果更强。

5.4.2 卷积神经网络CNN

根据已得数据，我们曾尝试过利用随机森林基于大类预测亚类，但由于数据量较少，验证集表现效果并不好，最终精准率得分仅0.6，所以我们尝试了卷积神经网络，效果表现要优于随机森林。

整个算法流程结合了模型选择和模型融合，多样性更强，更接近实际，其过程为：

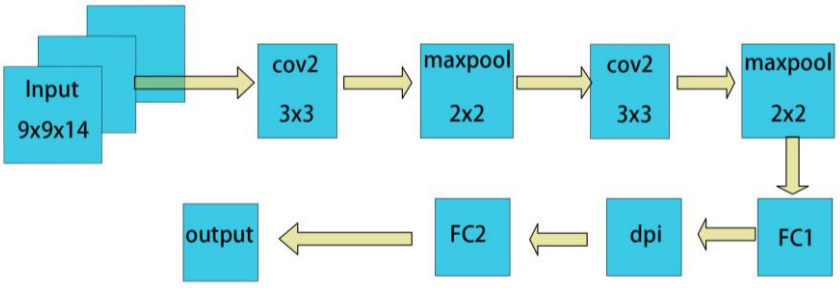


图5.28卷积神经网络结构

将训练集带入卷积神经网络中训练，并对模型进行评价，其准确率**89.89372**。

将模型代入附件表单3中进行预测，其预测结果为：

表5.5 预测结果

文物编号	类别	亚类
A1	高钾	高钙高铝
A2	铅钡	低钠低铜
A3	铅钡	低钠低铜
A4	铅钡	低钠低铜
A5	铅钡	低钠低铜
A6	高钾	高钙高铝
A7	高钾	高钙高铝
A8	铅钡	低钠高铜

同样对神经网络加入高斯噪声，其稳定性依然比较高，灵敏度较低。

5.5 问题四模型的建立与分析

5.5.1 皮尔逊相关系数模型

首先将所有的样品按照高钾类和铅钡类分为两类，再在每一类中按照是否发生风化，再细分为两类。然后，我们采用皮尔逊相关系数的方法在数据集中的各列之间找相关性，带入热图中，计算每一列两两之间的相关系数。得到

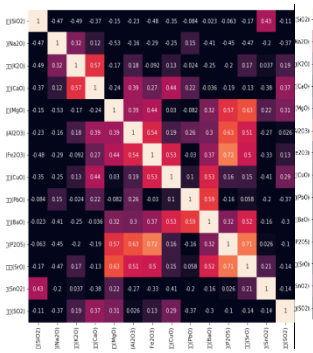


图5.29高钾-风化热图

图5.30铅钡-未风化热图

图5.31铅钡-风化热图

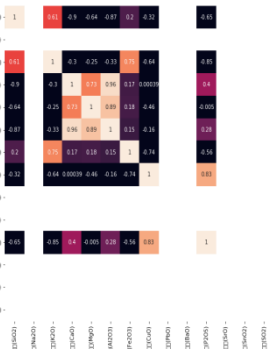


图5.32高钾-未风化热图

通过图像，我们得到相关系数>0.5的化合物组合，认为其具有关联性：

表5.6 具有关联性的化合物组合

分类	化合物组合
高钾-风化	CaO与K_2O,BaO与CuO,...等13种
高钾-未风化	K_2O与SiO_2,MgO与CaO,...等6种
铅钡-风化	Al_2O_3与K_2O,Fe_2O_3与CaO,...等6种
铅钡-未风化	Na_2O与SiO_2,BaO与CuO,...等6种

其全部化合物组合见支撑材料。

5.5.2 MVTEST库独立性检验

通过Python封装的MVTEST库，我们首先分为风化和未风化两类进行了独立性检验，再到计算相关性分析的时候，把数据集分为4类，分别进行独立性检验，计算每组数据的置信度p。经过分析我们找出其中置信度为99%的特征组，画它们的散点图进行统计分析，下面给出部分散点图。

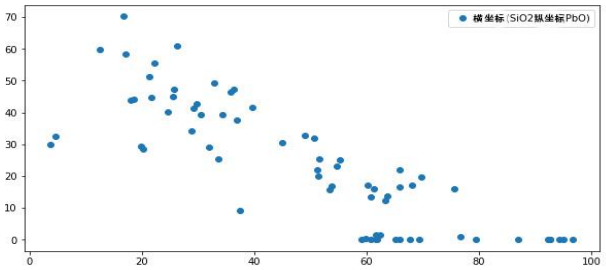


图5.30部分化合物组散点图

经过观察，我们可以得到在不同类别中不同化合物组合的正负相关性，如图5.30中，在其所属类别中，当二氧化硅含量升高时，其氧化铅的含量将会相应的降低，**所以在其所属类别中，二氧化硅与氧化铅呈负相关。经过我们查阅资料，呈现这种相关性的原因是由于铅类玻璃易受风化，这会导致二氧化硅的富集，这与我们得到的相关性相符。**

5.5.3 不同类别之间的化学成分关联关系的差异性分析

通过5.5.1中不同类别的热图，我们可以直观的看到不同类别中其化学成分组合的关联性差异，存在很多组在高钾类玻璃中关联性密切而在铅钡中毫无关联的数据，同理，风化前后其关联性也会发生变化。

例如，Al_2O_3和CaO在高钾型未风化类型中存在很强的相关性，但在高钾已风化中却很微弱，类似的数据还存在很多。

我们认为，造成这种差异性现象的原因是不同类型玻璃在制造时对于化学物质的需求不同，而不同的风化程度会造成其流失或产生的化合物含量不同。例如，若不考虑低含量化学物质会受高含量化学成分变化而引起本身成分占比变化，则基于**未风化的**特定类型玻璃来看，若某两种化学成分之间存在“此消彼长”的关系，则我们可以合理猜测这两种化学成分存在可互相替代性， 例如同为着色剂；若基于**已风化**的特定类型玻璃来看，若某两种化学成分之间存在“此消彼长”的关系，则我们应当猜测两者存在某种反应，其中一物会受益于风化影响而另一物会受损于风化影响。

六、 模型的评价与改进

模型的评价

在解决所有问题的过程中。我们根据不同类型的问题建立的不同的模型。

在第一题中，我们的以卡方检验算法为核心，分析了风化与玻璃特征的关联性，并对这个模型得到的结果进行了验证，结果不仅符合现实实际情况，而且也与Apriori算法得到的结果一致，这说明我们第一问采取的模型具有很高的准确率，我们的模型设计具有合理性。

同时，在预测数据的问题中，我们并没有采用传统的拟合算法，这是因为样本数据小，影响因素多，无法得到点对点的准确预测数据，所以我们采用了mapie区间预测算法，预测其风化前化合物含量的分布区间，并对该区间取均值，这样一来我们预测的结果便有着比较高的稳定性，并且容错率比较高。

在第二题要求我们对其进行亚分类时，我们对K-means聚类算法与系统聚类算法进行了比较。我们通过K-means聚类算法的得到的其中一个聚类如下：

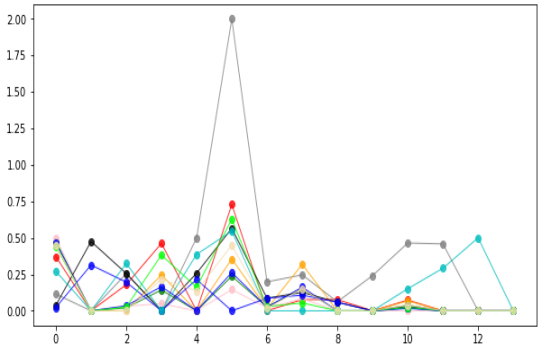


图6.1 K-means聚类算法结果

可以明显的看出，通过K-means算法得到的聚类冗杂，规律性差，不具有参考意义。这是由于样本量太小，K-means聚类效果并不好，于是我们换取了更适用于小样本的系统聚类法，聚类效果得到了显著的提高。

在第三题中，由于涉及训练模型，我们采取了多个机器学习算法进行比较，最后采取了拟合效果最好卷积神经网络模型，并且具有很强的解释性。其拟合效果也能够达到89%以上，这说明我们对其类型的鉴别有比较高的准确度。

模型的改进

在分类问题中，我们仅仅根据化学成分含量进行分类，并未深入挖掘化学成分以及颜色直接的关系以及颜色对于分类结果的影响，这有可能会使得分类结果不够精准。

由于数据集太小，我们认为没有必要尝试其他的深度学习模型，但在机器学习中，一种有效抑制过拟合的方法就是自制数据集，我们认为这对于我们这个赛题是有效且可行的，或许可以成为改进模型的一个方法。

参考文献

[1] 李青会,周虹志,黄教珍,干福熹,张平.一批中国古代镶嵌玻璃珠化学成分的检测报告[J].江汉考古,2005,(04):79-86+93.

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

[3] 成倩,张建林.北周武帝孝陵出土玻璃珠的科学分析与研究[J].考古与文物,2011,(01):107-112.

[4] 吴宗道,周福征,史美光.几个古玻璃的显微形貌、成分及其风化的初步研究[J].电子显微学报,1986,(04):65-71

[5] 王承遇,李松基,陶瑛.中国古代琉璃所用乳浊剂的演变[J].玻璃,2017,44(05):3-7.

[6] 付强,邝桂荣,吕良波,莫慧旋,李青会,干福熹.广州出土汉代玻璃制品的无损分析[J].硅酸盐学报,2013,41(07):994-1003..

附录

支撑材料:

```
D:.
├──第一问到三问
│   ├──mapie区间预测算法
│   │   └──预测结果
│   ├──picture
│   │   ├──元素风化箱式图
│   │   ├──表1风化特征柱状图
│   │   ├──预处理图
│   │   └──风化特征散点图
│   ├──数据集
│   └──源程序
│       ├──.ipynb_checkpoints
│       ├──dataset
│       └──新建文件夹
└──第四问
    ├──数据结果
    │   ├──总体
    │   ├──铅钡未风化
    │   ├──风化前
    │   ├──风化后
    │   └──高钾_风化
    └──源代码
        ├──.ipynb_checkpoints
        └──dataset
```

第一问代码

```
import pandas as pd
import numpy as np
import os
import gc
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import *
# 核心模型使用第三方库
from catboost import CatBoostClassifier
from sklearn.linear_model import SGDRegressor, LinearRegression, Ridge
import lightgbm as lgb
# 交叉验证所使用的第三方库
from sklearn.model_selection import StratifiedKFold, KFold
# 评估指标所使用的的第三方库
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, log_loss
# 忽略报警所使用的第三方库
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', 1000)
pd.set_option('display.max_row', 1000)
data_1 = pd.read_csv("D:/浏览器/数模/C 题/附件/表单 1.csv", encoding="ANSI")
data_2 = pd.read_csv("D:/浏览器/数模/C 题/附件/表单 2.csv", encoding="ANSI")
data_3 = pd.read_csv("D:/浏览器/数模/C 题/附件/表单 3.csv", encoding="ANSI")
data_t['纹饰'].unique()#C:0 A:1 B:2
data_t['类型'].unique()#'高钾':0, '铅钡':1
data_t['颜色'].unique()#'蓝绿':0, '浅蓝':1, '紫':2, '深绿':3, '深蓝':4, nan, '浅绿':6, '黑':7, '绿':8
data_t['纹饰_类型']=data_t['纹饰']+'_'+data_t['类型']
data_t['纹饰_颜色']=data_t['纹饰']+'_'+data_t['颜色']
data_t['类型_颜色']=data_t['类型']+'_'+data_t['颜色']
data_t['纹饰_类型'].unique()
data_t['纹饰_颜色'].unique()
data_t['类型_颜色'].unique()
def label_encoder(series):
    ls = list(series.unique())
    length = len(ls)
    cat_dict = dict(zip(ls, range(length)))
    return series.map(cat_dict)
data_1['纹饰_类型']=data_1['纹饰']+'_'+data_1['类型']
data_1['纹饰_颜色']=data_1['纹饰']+'_'+data_1['颜色']
```

```
data_1['类型_颜色']=data_1['类型']+'_'+data_1['颜色']
cols = [col for col in data_1.columns if col != '文物编号']
for i in tqdm(cols):
    data_1[i] = label_encoder(data_1[i])
data_1 = pd.get_dummies(data_1, columns=['纹饰','表面风化','类型', '颜色','纹饰_类型', '纹饰_颜色', '纹饰_颜色','类型_颜色'])
from mlxtend.frequent_patterns import apriori

apriori(data_1, min_support=0.2, use_colnames=True).sort_values(by='support', ascending=False)
cols = [col for col in data_1.columns if col not in ['文物编号', '表面风化']]
x_train = data_1[cols]
y_train = data_1['表面风化']
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
model1 = SelectKBest(chi2, k=6)#选择k个最佳特征
model1.fit_transform(x_train, y_train)
model1.scores_
model1.pvalues_
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = [u'SimHei']
plt.rcParams['axes.unicode_minus'] = False
col = '纹饰_颜色_类型'
plt.figure(figsize=(20, 10))
sns.countplot(x=col,hue ='表面风化' ,data=data_1)
plt.savefig("D:/浏览器/数模/C题/附件/{}_风化_count.png".format(col))
for col in cols:
    plt.figure(figsize=(20, 10))
    sns.countplot(x=col,hue ='表面风化' ,data=data_1)
    plt.savefig("D:/浏览器/数模/C题/附件/{}_风化_count.png".format(col))
for col in cols:
    plt.figure(figsize=(20, 10))
    sns.barplot(x = data_t[col], y = data_t['表面风化'] )
    plt.savefig("D:/浏览器/数模/C题/附件/{}_风化.png".format(col))
data_2['文物采样点'] = data_2['文物采样点'].astype(str)
def get_dig(str):
    return str[:2]
#data_2【17】【19】
data_2['文物编号'] = data_2['文物采样点'].apply(get_dig)
data_2['文物编号'] = data_2['文物编号'].astype(int)
data_2 = pd.merge(data_2,data_1[['文物编号', '表面风化', '类型']], how='left', on='文物编号')
data_2 = data_2.fillna(0)
data_2.drop(labels=[17, 19], axis=0, inplace = True)#删除17， 19
```

```
data_2 = data_2.reset_index()
del data_2['index']
for row in data_2.index:
    if '未风化点' in data_2.loc[row,'文物采样点']:
        data_2.loc[row,'表面风化']=0
    elif '严重风化点' in data_2.loc[row,'文物采样点']:
        data_2.loc[row,'表面风化']=1
data_2_type0 = data_2[data_2['类型']==0]
data_2_type1 = data_2[data_2['类型']==1]
chem_cols = [col for col in data_2.columns if col not in ['文物采样点', '文物编号', '表面风化', '类型']]
for col in chem_cols:
    plt.figure(figsize=(20, 10))
    sns.stripplot(x= '表面风化',y = col, data=data_2_type0)
    plt.savefig("D:/浏览器/数模/C 题/附件/高钾类_{}_风化_striplot.png".format(col))
for col in chem_cols:
    plt.figure(figsize=(20, 10))
    sns.stripplot(x= '表面风化',y = col, data=data_2_type1)
    plt.savefig("D:/浏览器/数模/C 题/附件/铅钡类_{}_风化_striplot.png".format(col))
data_changed = data_2.iloc[[9, 10, 26, 27, 53, 54, 55, 56], :]
 data_changed = data_changed.sort_values(by = ['文物编号','表面风化'])
 for col in chem_cols:
     data_changed[col+'_delta'] = data_changed[col]-data_changed.groupby('文物编号').shift(-1)[col]
 for col in chem_cols:
     data_changed[col+'_rate'] = data_changed[col+'_delta']/data_changed[col]
 rate_cols=[col for col in data_changed.columns if 'rate' in col]
 data_changed.loc[[9, 26,54,56],rate_cols]
 rate_dict={'二氧化硅(SiO2)':0.66,
  '氧化钠(Na2O)':0,
  '氧化钾(K2O)':0,
  '氧化钙(CaO)':-0.87,
  '氧化镁(MgO)':0,
  '氧化铝(Al2O3)':0.05,
  '氧化铁(Fe2O3)':-1.15,
  '氧化铜(CuO)':0,
  '氧化铅(PbO)':-0.25,
  '氧化钡(BaO)':-0.425,
  '五氧化二磷(P2O5)':-0.875,
  '氧化锶(SrO)':-0.75,
  '氧化锡(SnO2)':-0.775,
  '二氧化硫(SO2)':-5.9}
```

```
#对未风化数据进行假设性检验
data_2_type0 = data_2[(data_2['类型']==0) & (data_2['表面风化']==0)]
data_2_type1 = data_2[(data_2['类型']==1) & (data_2['表面风化']==0)]
for col in chem_cols:
    plt.figure(figsize=(20, 10))
    sns.distplot(data_2_type1[col],hist=True, kde=True)
    plt.show()
from fitter import Fitter
f = Fitter(data_2_type0['二氧化硅(SiO2)'], bins=20)
f.fit()
f.summary()
f.get_best()
data_2_fenhua1 = data_2[data_2['表面风化']==1]
from scipy import stats
llen = data_2_type0["二氧化硅(SiO2)"].shape[0]-1
mu = np.mean(data_2_type0["二氧化硅(SiO2)"])
std = np.std(data_2_type0["二氧化硅(SiO2)"], ddof=1)
se = std/np.sqrt(data_2_type0["二氧化硅(SiO2)"].shape[0])
interval = stats.t.interval(0.95, llen, mu, se)
ls0 = []
for col in chem_cols:
    llen = data_2_type0[col].shape[0]-1
    mu = np.mean(data_2_type0[col])
    std = np.std(data_2_type0[col], ddof=1)
    se = std/np.sqrt(data_2_type0[col].shape[0])
    interval = stats.t.interval(0.95, llen, mu, se)
    ls0.append(interval)
    print(col,interval)
ls1 = []
for col in chem_cols:
    llen = data_2_type1[col].shape[0]-1
    mu = np.mean(data_2_type1[col])
    std = np.std(data_2_type1[col], ddof=1)
    se = std/np.sqrt(data_2_type1[col].shape[0])
    interval = stats.t.interval(0.95, llen, mu, se)
    ls1.append(interval)
    print(col,interval)
data_2_fenhua1[data_2_fenhua1['类型']==0]
chosen0 = data_2_fenhua1[data_2_fenhua1['类型']==0]
chosen1 = data_2_fenhua1[data_2_fenhua1['类型']==1]
#根据rate 以及置信区间求之前
for i in range(len(chem_cols)):
    chosen0[chem_cols[i]] = (ls0[i][0]+ls0[i][1])/2
```

```
        chosen1[chem_cols[i]] = data_2_fenhua1[chem_cols[i]]/(1-rate_dict[chem_cols[i]])
    #修正铅钡型：
    for i in range(len(data_2_fenhua1[data_2_fenhua1['类型']==1])):
        for j in range(len(chem_cols)):
            if chosen1.iloc[i, j+1] < ls1[j][0]:
                chosen1.iloc[i, j+1] = ls1[j][0]
            elif chosen1.iloc[i, j+1] >ls1[j][1]:
                chosen1.iloc[i, j+1] = ls1[j][1]
    data_2_fenhua1 = pd.concat([chosen0, chosen1]).sort_values(by='文物采样点')
    data_2_fenhua1.to_csv("D:/浏览器/数模/C 题/1.3.csv", encoding="utf-8")
    rate_dict['二氧化硅(SiO2)']
    for col in chem_cols:
        plt.figure(figsize=(20, 10))
        sns.boxplot(x='类型',y=col,data = data_2[data_2['表面风化']==0])
        plt.savefig("D:/浏览器/数模/C 题/附件/type_{}_box.png".format(col))
    data_2 = pd.merge(data_2, data_1[['文物编号','类型']], how='left', on='文物编号')
    chem_cols = [col for col in data_2.columns if col not in ['文物采样点', '文物编号', '类型', '表面风化']]
    cut_list = []
    for col in chem_cols:
        data_2[col],cut_bins=pd.cut(data_2[col],5,labels = range(5),retbins=True)
        cut_list.append(cut_bins)
    cut_list
    for col in chem_cols:
        plt.figure(figsize=(10, 5))
        sns.barplot(x=col, y='类型', data=data_2)
    #根据上修改
    data_2['二氧化硅(SiO2)'] = data_2['二氧化硅(SiO2)'].replace([0], [1])
    data_2['二氧化硅(SiO2)'] = data_2['二氧化硅(SiO2)'].replace([0], [1])
    data_2['氧化钠(Na2O)'] = data_2['氧化钠(Na2O)'].replace([4], 3)
    data_2['氧化钾(K2O)'] = data_2['氧化钾(K2O)'].replace([3,4], [2, 2])
    data_2['氧化镁(MgO)'] = data_2['氧化镁(MgO)'].replace([0], 1)
    data_2['氧化铅(PbO)'] = data_2['氧化铅(PbO)'].replace([2, 3, 4], [1, 1, 1])
    data_2['氧化钡(BaO)'] = data_2['氧化钡(BaO)'].replace([2, 3, 4], [1, 1, 1])
```

```
data_2['五氧化二磷(P2O5)'] = data_2['五氧化二磷(P2O5)'].replace([3,4], [2, 2])
data_2['氧化锶(SrO)'] = data_2['氧化锶(SrO)'].replace([2, 3, 4], [1, 1,1])
data_2['氧化锡(SnO2)'] = data_2['氧化锡(SnO2)'].replace([4], 3)
data_2['二氧化硫(SO2)'] = data_2['二氧化硫(SO2)'].replace([2], 3)
x_train = data_2[data_2['表面风化']==0][chem_cols]
y_train = data_2[data_2['表面风化']==0]['类型']
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
model1 = SelectKBest(chi2, k=6)#选择k个最佳特征
model1.fit_transform(x_train, y_train)
model1.scores_
model1.pvalues_
data_2_type0 = data_2[(data_2['类型']==0)&(data_2['表面风化']==0)].reset_index()
data_2_type1 = data_2[(data_2['类型']==1)&(data_2['表面风化']==0)].reset_index()
# 归一化函数
def Normalization(x):
    return (x-min(x))/(max(x)-min(x))
for col in xcols:
    data_2_type0[col] = Normalization(data_2_type0[col])
    data_2_type1[col] = Normalization(data_2_type1[col])
#高钾类:
#氧化钙(CaO)  氧化铝(Al2O3)
data_2_type0['氧化钙(CaO)'] = data_2_type0['氧化钙(CaO)']*4
data_2_type0['氧化铝(Al2O3)'] = data_2_type0['氧化铝(Al2O3)']*4
#铅钡类:
#氧化钠(Na2O)  五氧化二磷(P2O5)
data_2_type1['氧化钠(Na2O)'] = data_2_type1['氧化钠(Na2O)']*4
data_2_type1['氧化铜(CuO)'] = data_2_type1['氧化铜(CuO)']*4

```

第二问代码

```
from sklearn.cluster import KMeans
from sklearn import metrics
k_means = KMeans(n_clusters=2, random_state=10)
k_means.fit(x)
y_predict = k_means.predict(x)
y_predict
plt.scatter(x.loc[:,'氧化钙(CaO)'], x.loc[:,'氧化铝(Al2O3)'], c=y_predict)
x['y_pred'] = y_predict
```

```
for col in xcols:
    data_2_type0.astype(float)
for i in range(3):
    plt.figure(figsize=(20, 10))
    t = x[x['y_pred']==i]
    for j in range(len(t)):
        plt.plot(range(len(xcols)), t[xcols].iloc[j])
from sklearn.cluster import AgglomerativeClustering
from itertools import cycle
ac = AgglomerativeClustering(linkage='complete', n_clusters=3)
ac.fit(data_2_type0[xcols])
labels = ac.labels_
data_2_type0['pre'] = labels
ac = AgglomerativeClustering(linkage='complete', n_clusters=3)
ac.fit(data_2_type1[xcols])
labels = ac.labels_
data_2_type1['pre'] = labels
data_2_type0['亚类'] = data_2_type0['pre']
data_2_type1['亚类'] = data_2_type1['pre']
data_2_type = pd.concat([data_2_type0, data_2_type1], axis=0)
del data_2_type['index']
data_2_type.to_csv("D:/浏览器/数模/C 题/data_2_type_predict.csv")
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
for i in range(3):
    plt.figure(figsize=(20, 10))
    plt.ylim(0, 4)
    t = data_2_type0[data_2_type0['pre']==i]
    for j in range(len(t)):
        plt.plot(range(len(xcols)), t[xcols].iloc[j])
    plt.savefig("D:/浏览器/数模/C 题/附件/{}_agglome_高
钾.png".format(i))
for i in range(3):
    plt.figure(figsize=(20, 10))
    plt.ylim(0, 4)
    t = data_2_type1[data_2_type1['pre']==i]
    for j in range(len(t)):
        plt.plot(range(len(xcols)), t[xcols].iloc[j])
    plt.savefig("D:/浏览器/数模/C 题/附件/{}_agglome_铅
钡.png".format(i))
data_2_type = pd.concat([data_2_type0,data_2_type1], axis=0).reset
_index()
data_2_type.to_csv("D:\\浏览器\\数模\\C 题
\\data_2_type_predict.csv")
import random
data_2_type0[xcols]+= random.gauss(0, 0.1)
from sklearn.cluster import AgglomerativeClustering
```

```
from itertools import cycle
ac = AgglomerativeClustering(linkage='complete', n_clusters=3)
ac.fit(data_2_type0[xcols])
labels = ac.labels_
sklearn.metrics.mean_squared_error(data_2_type0['pre'], data_2_type0['pre_y'])
```

第三问代码

```
#首先预测大类型：高钾？铅钡？
X_cols = [col for col in data_2.columns if col not in ['文物编号', '表面风化', '类型', '文物采样点']]
X = data_2[X_cols]
y = data_2['类型']
from sklearn.model_selection import train_test_split
X_train, x_var, y_train, y_var = train_test_split(
X, y, test_size=0.2, random_state=10)
from lightgbm import LGBMClassifier
params = {'feature_fraction':0.6,'learning_rate': 0.1, 'n_estimators':100,'early_stop_round':100, 'max_depth': 4, 'l2_leaf_reg': 10, 'bootstrap_type':'Bernoulli','random_seed':2022,
                    'od_type': 'Iter', 'od_wait': 50, 'random_seed': 11, 'allow_writing_files': False}
lgbm = LGBMClassifier(iterations=200, **params, eval_metric='AUC')
lgbm.fit(X_train, y_train, eval_set=(x_var, y_var),
                    verbose=1)
y_pre=lgbm.predict(x_var)
f1_score(y_pre, y_var)
cat =  CatBoostClassifier(iterations=100, depth=5,learning_rate=0.5, loss_function='Logloss',
                                logging_level='Verbose')
cat.fit(X_train, y_train, eval_set=(x_var, y_var),
                    verbose=1)
y_pre = cat.predict(x_var)
f1_score(y_var, y_pre)
from lightgbm import plot_importance
plot_importance(lgbm)
from sklearn.svm import SVC
svc = SVC(kernel='rbf', probability=True)
svc.fit(X_train, y_train)
y_pre = svc.predict(x_var)
f1_score(y_var, y_pre)
x_test = data_3[chem_cols]
data_3['类型'] = cat.predict(x_test)
data_3 = data_3.fillna(0)
data_3_needs = data_3[data_3['表面风化']=='风化' ]
data_3_needs_0 = data_3_needs[data_3_needs['类型']==0]
```

```
data_3_needs_1 = data_3_needs[data_3_needs['类型']==1]
#根据rate 以及置信区间求之前
for i in range(len(chem_cols)):
    data_3_needs_0[chem_cols[i]] = (ls0[i][0]+ls0[i][1])/2
    data_3_needs_1[chem_cols[i]] = data_3_needs_1[chem_cols[i]]/(1-rate_dict[chem_cols[i]])
 #修正铅钡型:
for i in range(len(data_3_needs_1)):
    for j in range(len(chem_cols)):
        if data_3_needs_1.iloc[i, j+2] < ls1[j][0]:
            data_3_needs_1.iloc[i, j+2] = ls1[j][0]
        elif data_3_needs_1.iloc[i, j+2] >ls1[j][1]:
            data_3_needs_1.iloc[i, j+2] = ls1[j][1]
data_3_needs = pd.concat([data_3_needs_0, data_3_needs_1])
data_3 = pd.concat([data_3_needs, data_3[data_3['表面风化']=='无风化']]).sort_values(by = '文物编号')
data_2_type = data_2[data_2['亚类'].isnull()==False]
data_2_type = pd.merge(data_2[data_2['亚类'].isnull()==False], data_2_type[['类型', '文物采样点']],how = 'left', on ='文物采样点' )
data_2_type = data_2_type.fillna(0)
from sklearn.ensemble import RandomForestClassifier
rfc1 = RandomForestClassifier()
rfc2 = RandomForestClassifier()
data_2_type0 = data_2_type[data_2_type['类型']==0]
data_2_type1 = data_2_type[data_2_type['类型']==1]
X_0 = data_2_type0[chem_cols]
y_0 = data_2_type0['亚类']
X_1 = data_2_type1[chem_cols]
y_1 = data_2_type1['亚类']
from sklearn.model_selection import train_test_split
X_train_0, x_var_0, y_train_0, y_var_0 = train_test_split(
X_0, y_0, test_size=0.2, random_state=10)
from sklearn.model_selection import train_test_split
X_train_1, x_var_1, y_train_1, y_var_1 = train_test_split(
X_1, y_1, test_size=0.2, random_state=10)
rfc1.fit(X_train_0, y_train_0)
y_pre_0=rfc1.predict(x_var_0)
accuracy_score(y_pre_0, y_var_0)
rfc2.fit(X_train_1, y_train_1)
y_pre_1 = rfc2.predict(x_var_1)
accuracy_score(y_pre_1, y_var_1)
y_var_1
data_3_needs_0 = data_3[data_3['类型']==0]
test0=rfc1.predict(data_3_needs_0[chem_cols])
data_3_needs_1 = data_3[data_3['类型']==1]
```

```
test1 = rfc2.predict(data_3_needs_1[chem_cols])
data_3_needs_0['亚类'] = test0
data_3_needs_1['亚类'] = test1
data_3_needs = pd.concat([data_3_needs_0, data_3_needs_1]).sort_values(by = '文物编号')
data_3_needs.to_csv("D:/浏览器/数模/C 题/data_2Yatype_random_tree.csv")
x = torch.tensor(data_2_type[chem_cols].values).reshape(35, 14)
x = x=x.float()
x = x.unsqueeze(1)
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size = 3,stride=1,padding = 1) #2*2*32
        self.mp1 = nn.MaxPool2d(kernel_size=2,stride=2)#1132
        self.conv2 = nn.Conv2d(16,64, kernel_size=3,stride =1)#  2 2 64
        self.mp2 = nn.MaxPool2d(kernel_size=2,stride=2) #1 1 64
        self.fc1= nn.Linear(32,32)#32 32 32
        self.dp1 = nn.Dropout(p=0.2)
        self.fc2 = nn.Linear(14*4*4,10)

    def forward(self, x):
        out = self.conv1(x)
        #out shape(batch,16,5,5)
        out = out.view(-1, 14*4*4)     #out shape()
        out = self.fc2(out)                  #out shape(batch,10)
        return out
net = CNN()
data_2_type0 = data_2_type[data_2_type['类型']==0]
X_0 = data_2_type0[chem_cols]
y_0 = data_2_type0['亚类']
from sklearn.model_selection import train_test_split
X_train_0, x_var_0, y_train_0, y_var_0 = train_test_split(
X_0, y_0, test_size=0.2, random_state=10)
X_train_0 = torch.tensor(X_train_0.values)
X_train_0 = X_train_0.float()
X_train_0 = X_train_0.unsqueeze(1)
X_train_0.shape
X_train_0
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
#常用优化方法有
```

```
#1.Stochastic Gradient Descent (SGD)
#2.Momentum
#3.AdaGrad
#4.RMSProp
#5.Adam (momentum+adaGrad)   效果较好
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), std =(0.5, 0.5, 0.5))])
inputs, labels = X_train_0,torch.tensor(y_train_0.values)
# zeros the paramster gradients
optimizer.zero_grad()

# forward + backward + optimize
outputs = net(inputs)
loss = criterion(outputs, labels)  # 计算loss
loss.backward()     # loss 求导
optimizer.step()    # 更新参数

# print statistics
running_loss += loss.item()  # tensor.item()  获取tensor 的数值

print('Finished Training')
data_2_type0['亚类']
```

第四问代码1

```
import pandas as pd
import numpy as np
import os
import gc
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import *
# 核心模型使用第三方库
from catboost import CatBoostClassifier
from sklearn.linear_model import SGDRegressor, LinearRegression, Ridge
import lightgbm as lgb
# 交叉验证所使用的第三方库
from sklearn.model_selection import StratifiedKFold, KFold
# 评估指标所使用的的第三方库
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, log_loss
# 忽略报警所使用的第三方库
import warnings
warnings.filterwarnings('ignore')
import mvtpy
data_2 = pd.read_csv("t2.csv")
```

```
data_1 = pd.read_csv("t1.csv")
def get_dig(str):
    return str[:2]
# data_2 = pd.read_csv("t2.csv")

#data_2【17】【19】
data_2['文物编号'] = data_2['文物采样点'].apply(get_dig)

data_2['文物编号'] = data_2['文物编号'].astype(int)

data_2 = pd.merge(data_2,data_1[['文物编号','类型','表面风化
    ']], how='left', on='文物编号')
data_2 = data_2.fillna(0)
def label_encoder(series):
    ls = list(series.unique())
    length = len(ls)
    cat_dict = dict(zip(ls, range(length)))
    return series.map(cat_dict)
data_2['类型'] = label_encoder(data_2['类型'])
data_2['表面风化'] = label_encoder(data_2['表面风化'])
del data_2['文物采样点']
del data_2['文物编号']
data_2_type0 = data_2[data_2['类型']==0]
data_2_type1 = data_2[data_2['类型']==1]
data_2_type0_f = data_2_type0[data_2_type0['表面风化']==0]
data_2_type0_uf = data_2_type0[data_2_type0['表面风化']==1]
data_2_type1_f = data_2_type1[data_2_type1['表面风化']==0]
data_2_type1_uf = data_2_type1[data_2_type1['表面风化']==1]
plt.figure(figsize=(20, 10))
sns.heatmap(data_2_type0_f.corr(),vmin=0,vmax=1,square=True,annot=Tru
    e)
data_2_type0_uf.corr()
del data_2_type0_uf['类型']
del data_2_type0_uf['表面风化']
plt.figure(figsize=(20, 10))
sns.heatmap(data_2_type0_uf.corr(),vmin=0,vmax=1,square=True,annot=Tr
    ue)
del data_2_type1_f['类型']
del data_2_type1_f['表面风化']
plt.figure(figsize=(20, 10))
sns.heatmap(data_2_type1_f.corr(),vmin=0,vmax=1,square=True,annot=Tru
    e)
del data_2_type1_uf['类型']
del data_2_type1_uf['表面风化']
plt.figure(figsize=(20, 10))
```

```
sns.heatmap(data_2_type1_uf.corr(),vmin=0,vmax=1,square=True,annot=True)
data = pd.read_csv('t2.csv')
from mvtpy.mvtest import mvtest
model = mvtest()
model.test(data['二氧化硅(SiO2)'], data['氧化钠(Na2O)'])
del data['文物采样点']
cols = [col for col in data.columns]
data = data.fillna(0)
for i in range(14):
    for j in range(i+1,14):
        tmp = model.test(data[cols[i]], data[cols[j]])
        ls_tmp = model._quantiles_transformer(tmp['Tn'])
#         print(tmp)
#         print(ls_tmp)
        if ls_tmp[1] <= 0.01:
            plt.figure(figsize = (10,5))
#             plt.plot(data[cols[i]],data[cols[j]], 'ro-', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
            plt.scatter(data[cols[i]],data[cols[j]],label = cols[i]+"_"+cols[j])
            plt.legend(loc="upper right")
            plt.savefig("D:/Desktop/数模 gs/CUMCM2022Problems/C 题/no4_picture/总体/总体{}_{}.png".format(cols[i],cols[j]))
#         dep_dict.append(model.test(data[cols[i]], data[cols[j]]))

```

第四问代码2

```
import pandas as pd
import numpy as np
import os
import gc
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import *
# 核心模型使用第三方库
from catboost import CatBoostClassifier
from sklearn.linear_model import SGDRegressor, LinearRegression, Ridge
import lightgbm as lgb
# 交叉验证所使用的第三方库
from sklearn.model_selection import StratifiedKFold, KFold
# 评估指标所使用的的第三方库
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, log_loss
# 忽略报警所使用的第三方库
```

```
import warnings
warnings.filterwarnings('ignore')
import mvtpy
data_2 = pd.read_csv("t2.csv")
data_1 = pd.read_csv("t1.csv")
def get_dig(str):
    return str[:2]
# data_2 = pd.read_csv("t2.csv")

#data_2【17】【19】
data_2['文物编号'] = data_2['文物采样点'].apply(get_dig)

data_2['文物编号'] = data_2['文物编号'].astype(int)

data_2 = pd.merge(data_2,data_1[['文物编号','类型','表面风化
    ']], how='left', on='文物编号')
data_2 = data_2.fillna(0)
def label_encoder(series):
    ls = list(series.unique())
    length = len(ls)
    cat_dict = dict(zip(ls, range(length)))
    return series.map(cat_dict)
data_2['类型'] = label_encoder(data_2['类型'])
data_2['表面风化'] = label_encoder(data_2['表面风化'])
del data_2['文物采样点']
del data_2['文物编号']
data_2_type0 = data_2[data_2['类型']==0]
data_2_type1 = data_2[data_2['类型']==1]
data_2_type0_f = data_2_type0[data_2_type0['表面风化']==0]
data_2_type0_uf = data_2_type0[data_2_type0['表面风化']==1]
data_2_type1_f = data_2_type1[data_2_type1['表面风化']==0]
data_2_type1_uf = data_2_type1[data_2_type1['表面风化']==1]
del data_2_type0_f['类型']
del data_2_type0_f['表面风化']
from mvtpy.mvtest import mvtest
model = mvtest()
cols = [col for col in data_2_type0.columns]
for i in range(14):
    for j in range(i+1,14):
        tmp = model.test(data_2_type0_f[cols[i]], data_2_type0_f[cols
    [j]])
        ls_tmp = model._quantiles_transformer(tmp['Tn'])
        print(tmp)
        print(ls_tmp)
        if ls_tmp[1] <= 0.01:
            plt.figure(figsize = (10,5))
```

```
#             plt.plot(data_2_type0[cols[i]],data_2_type0[cols[j]], 'ro-
', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
            plt.scatter(data_2_type0_f[cols[i]],data_2_type0_f[cols[j]],label = cols[i]+"_"+cols[j])
            plt.legend(loc="upper right")
#         dep_dict.append(model.test(data[cols[i]], data[cols[j]]))
del data_2_type0_uf['类型']
del data_2_type0_uf['表面风化']
for i in range(14):
    for j in range(i+1,14):
        tmp = model.test(data_2_type0_uf[cols[i]], data_2_type0_uf[cols[j]])
        ls_tmp = model._quantiles_transformer(tmp['Tn'])
        print(tmp)
        print(ls_tmp)
        if ls_tmp[1] <= 0.01:
            plt.figure(figsize = (10,5))
#             plt.plot(data_2_type1[cols[i]],data_2_type1[cols[j]], 'ro-
', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
            plt.scatter(data_2_type0_uf[cols[i]],data_2_type0_uf[cols[j]],label = cols[i]+"_"+cols[j])
            plt.legend(loc="upper right")
#         dep_dict.append(model.test(data[cols[i]], data[cols[j]]))
del data_2_type1_f['类型']
del data_2_type1_f['表面风化']
for i in range(14):
    for j in range(i+1,14):
        tmp = model.test(data_2_type1_f[cols[i]], data_2_type1_f[cols[j]])
        ls_tmp = model._quantiles_transformer(tmp['Tn'])
        print(tmp)
        print(ls_tmp)
        if ls_tmp[1] <= 0.01:
            plt.figure(figsize = (10,5))
#             plt.plot(data_2_type1[cols[i]],data_2_type1[cols[j]], 'ro-
', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
            plt.scatter(data_2_type1_f[cols[i]],data_2_type1_f[cols[j]],label = cols[i]+"_"+cols[j])
            plt.legend(loc="upper right")
#         dep_dict.append(model.test(data[cols[i]], data[cols[j]]))
```

```
del data_2_type1_uf['类型']
del data_2_type1_uf['表面风化']
for i in range(14):
    for j in range(i+1,14):
        tmp = model.test(data_2_type1_uf[cols[i]], data_2_type1_uf[cols[j]])
        ls_tmp = model._quantiles_transformer(tmp['Tn'])
        print(tmp)
          print(ls_tmp)
          if ls_tmp[1] <= 0.01:
              plt.figure(figsize = (10,5))
  #             plt.plot(data_2_type1[cols[i]],data_2_type1[cols[j]], 'ro-', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
              plt.scatter(data_2_type1_uf[cols[i]],data_2_type1_uf[cols[j]],label = cols[i]+"_"+cols[j])
              plt.legend(loc="upper right")
              plt.savefig("D:/Desktop/数模 gs/CUMCM2022Problems/C 题/no4_picture/铅钡未风化/铅钡未分化_{}_{}.png".format(cols[i],cols[j]))
  #         dep_dict.append(model.test(data[cols[i]], data[cols[j]]))
  # 风化前
  del data_2['类型']
  data_2_type0 = data_2[data_2['表面风化']==0]
  data_2_type1 = data_2[data_2['表面风化']==1]
  from mvtpy.mvtest import mvtest
  model = mvtest()
  for i in range(14):
      for j in range(i+1,14):
          tmp = model.test(data_2_type0[cols[i]], data_2_type0[cols[j]])
          ls_tmp = model._quantiles_transformer(tmp['Tn'])
          print(tmp)
          print(ls_tmp)
          if ls_tmp[1] <= 0.01:
              plt.figure(figsize = (10,5))
  #             plt.plot(data_2_type0[cols[i]],data_2_type0[cols[j]], 'ro-', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j])
              plt.scatter(data_2_type0[cols[i]],data_2_type0[cols[j]],label = cols[i]+"_"+cols[j])
              plt.legend(loc="upper right")
              plt.savefig("D:/Desktop/数模 gs/CUMCM2022Problems/C 题/no4_picture/风化前/风化前_{}_{}.png".format(cols[i],cols[j]))
```

```
126.  #         dep_dict.append(model.test(data[cols[i]], data[cols[j]])
    )
127.  for i in range(14):
128.      for j in range(i+1,14):
129.          tmp = model.test(data_2_type1[cols[i]], data_2_type1[cols[
    j]])
130.          ls_tmp = model._quantiles_transformer(tmp['Tn'])
131.          print(tmp)
132.          print(ls_tmp)
133.          if ls_tmp[1] <= 0.01:
134.              plt.figure(figsize = (10,5))
135.  #             plt.plot(data_2_type0[cols[i]],data_2_type0[cols[j]]
    , 'ro-
    ', color='#4169E1', alpha=0.8, linewidth=1, label=cols[i]+'_'+cols[j]
    )
136.              plt.scatter(data_2_type1[cols[i]],data_2_type1[cols[j]
    ],label = cols[i]+"_"+cols[j])
137.              plt.legend(loc="upper right")
138.              plt.savefig("D:/Desktop/数模 gs/CUMCM2022Problems/C 题
    /no4_picture/风化后/风化后_{}_{}.png".format(cols[i],cols[j]))
139.  #         dep_dict.append(model.test(data[cols[i]], data[cols[j]])
    )
140.
```