基于机器学习统计分类模式的玻璃制品成分分析鉴别

摘要

针对**问题**一，本文将问题分为两个小问进行求解，在第一小问中，针对多维度多分类数据，通过模糊数学相关性分析和岭回归的方法进行量化分析，得到岭回归方程。在第二小问中，本文结合第一小问岭回归模型对 14 种化学成分逐一进行回归，预测了未风化文物风化前各化学元素含量，并结合两个既有风化点又有非风化点的文物（49，50）进行验证，预测值和拟合值的误差为 0.7%。

针对**问题**二，本文首先对清洗后的数据进行 KMO 检验和 Bartlett 球形检验验证其适合做主成分分析，在提取主成分后得到高钾玻璃、铅钡玻璃的分类规律。然后将 63 个样本分成高钾玻璃和铅钡玻璃两类，对每类分别进行层次分析法得到具体的亚分类，分别是 BaO · CuO 系列、Na2O 系列、PbO 系列、K2O · Na2O 系列、PbO · Al2O3 系列、K2O 系列和 PbO · BaO 系列。

针对**问题**三，本文采用随机森林、AdaBoost 、XGboost 和 LightGBM 等分类预测算法对未知类别的的古代玻璃制品进行分类，以 1 值等作为重要标准，最终选择 XGboost 作为最优分类预测模型，以期实现对未知玻璃制品的正确分类。最终，本文运用分类预测模型确定 A1、A5、A6 和 A7 为高钾玻璃，A2、A3、A4 和 A8 为铅钡玻璃。对于分类结果，本文通过变更特定指标的方式来检验预测分类模型的敏感性们，将二氧化硅含量上下波动 5% 的区间进行敏感性分析，最终结果体现出分类预测模型具备较好的分类性能。

针对**问题四**，本文首先对亚分类中 7 个系列中玻璃化学成分含量分别做组内和组间的单因素方差分析，得到不同亚类的玻璃具有显著差异的结论。然后分别对高钾玻璃组内和铅钡玻璃组内做 14 种化学成分含量的相关性分析，得到热力图。最后定义了样品的化学组成向量，对 7 种亚变量的组间向量进行相关性分析，得到 K2O · Na2O 系列和 K2O 系列具有较强的正关联性，BaO · CuO 系列和 K2O · Na2O 系列具有负相关性等结论。

关键词: 玻璃文物　岭回归　主成分分析　层次聚类　XGboost

一、问题重述

1.1 问题背景

玻璃制品是中国古代对外贸易繁荣开放的重要物证，其中铅钡玻璃和钾玻璃均是是中国古代玻璃制品的工艺瑰宝。由于古代玻璃制品易受埋藏环境影响而风化，玻璃制品内部极易发生与外界元素反应的情况，从而影响对古代玻璃制品类别的判断。因此，在中国古代玻璃制品的文物保护工作中，玻璃文物样品化学成分的分析与鉴别十分重要。

1.2 需要解决的问题

问题一：根据现有玻璃文物的类型、纹饰与颜色等相关数据信息，分析其与玻璃文物风化情况之间的关联。在考虑玻璃类型的基础上，分析文物样品表面有无风化化学成分含量的统计规律，结合现阶段风化点采样点检测数据，预测在玻璃制品风化前的化学成分含量。

问题二: 根据表单 2 的相关数据，探索玻璃类型的分类规律，并且针对铅钡玻璃和高钾玻璃两类，进行更加细致的亚分类处理，并对分类结果的合理性与敏感性做出评价。

问题三: 通过分析未明确类别的玻璃文物化学成分组成，对 8 件玻璃文物的类别进行鉴定，并对分类预测结果进行敏感性分析。

问题四：在前面问题探讨的基础上，分析不同类别玻璃文物内部化学成分的相关性，同时比较不同类别玻璃文物化学成分的差异性。

二、模型假设

- 假设玻璃文物有且仅有 14 种化学成分；
- 假设玻璃文物在短时间内化学成分含量保持稳定；
- 假设所有玻璃文物处于稳定，同种的物理环境中；
- 假设测定的玻璃文物化学成分是准确的；
- 假设同一文物不同采样点之间不会发生化学物质交换。

三、符号说明

符号	说明
$T_i(i=1,2)$	玻璃类型
$D_i(i=1,2,3)$	玻璃花纹
$C_i(i=1,2,...,8)$	玻璃颜色

四、模型的建立与求解

4.1 问题一

4.1.1 问题分析

根据题意将本文分成两个小问，第一小问根据表单 1 提供的文物样本信息，分析玻璃制品面风化程度与玻璃类型、纹饰和颜色等关键特征指标可能存在的相关关系；第二小问需要结合表单 1 与表单 2 的文物样本信息对有无风化的玻璃制品化学成分进行统计与描述，并根据风化点检测数据对文物在风化作用发生前的化学含量进行预测。

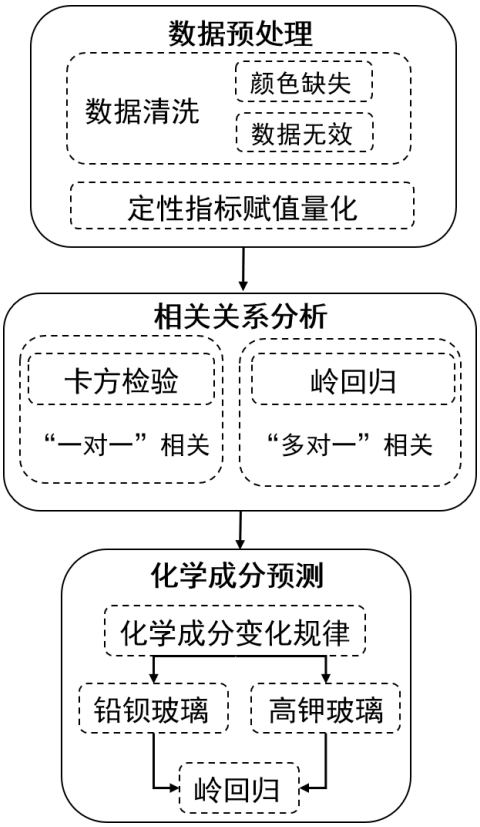


图 1: 问题一流程图

4.1.2 数据预处理

A. 数据清洗

由于第一小问需要研究玻璃制品风化程度与颜色之间的关系，由于 19 号、40 号、48 号和 58 号文物颜色信息缺失，因此在第一小问中应该将上述四组数据予以剔除；在第二小问中，题干规定文物采样样本成分比例累加合计应居于 85 % -105% 的区间，所以 19 号和 21 号两组数据为无效数据，本文进行了剔除。

B. 定性指标赋值量化

针对表单一中的文本数据，本文采取赋值量化的方式对定性指标进行处理。对纹饰、玻璃类型、颜色和表面是否风化等信息的定量赋值规则如下，

表 1: 编码规则

属性	分类	编码	属性	类别	编码
类型	铅钡玻璃	1	颜色	浅绿	1
				浅蓝	2
	高钾玻璃	2		蓝绿	3
				绿	4
纹饰	A	1		深绿	5
	B	2		深蓝	6
	C	3		紫	7
				黑	8

在制定编码规则后，本文对 54 组文物样本有效数据进行了重新编码，限于篇幅，在此仅列出数据处理后前 20 号文物的数据，全部数据详见附录 1。

表 2: 玻璃文物编码数据（部分）

文物编号	纹饰	类型	颜色	表面风化	文物编号	纹饰	类型	颜色	表面风化
1	3	1	3	0	11	3	2	2	1
2	1	2	2	1	12	2	1	3	1
3	1	1	3	0	13	3	1	2	0
4	1	1	3	0	14	3	1	5	0
5	1	1	3	0	15	3	1	2	0
6	1	1	3	0	16	3	1	2	0
7	2	1	3	1	17	3	1	2	0
8	3	2	7	1	18	1	1	6	0
9	2	1	3	1	19	1	2	2	0
10	2	1	3	1	20	1	1	3	0

4.1.3 问题一第（1）问模型建立与求解

Step1：卡方检验

本文通过卡方检验分别描述古代玻璃制品表面风化程度与玻璃纹饰、类型和颜色等定类变量之间的相关程度。

首先，本文对玻璃类型、纹饰和颜色等三个变量的量化数据进行了正态性检验，发现三类变量的数据都不满足正态性特征，其检验结果详见附录。因此，对于相关程度的分析需要采取非参数检验的方法，针对表单 1 均为定类变量的数据特征的情况，故采用**卡方检验**。

卡方检验中卡方值越大表示实际与期望的相关性越大，独立性越小；卡方值越小表示实际与期望的相关性越小，独立性越大。卡方检验的公式为:

$$\chi^2 = \sum \frac{(f_0 - f_e)^2}{f_e} \tag{1}$$

其中，f_0 表示实际频数，f_e 为期望频数，运用卡方检验描述玻璃制品表面风化程度与三类变量的相关程度，详细结果如下表所示。

表 3: 卡方检验

属性	名称	表面风化		合计	卡方值	P值	决策
		无风化	风化				
类型	铅钡玻璃（T1）	12	24	36	5.400	0.020	有显著差异
	高钾玻璃（T2）	12	6	18			
合计		24	30	54			
纹饰	纹饰A（D1）	11	9	20	5.747	0.056	有显著差异
	纹饰B（D2）	0	6	6			
	纹饰C（D3）	13	15	28			
合计		24	30	54			
颜色	浅绿（C1）	2	1	3	6.287	0.507	无显著差异
	浅蓝（C2）	8	12	20			
	蓝绿（C3）	6	9	15			
	绿（C4）	1	0	1			
	深绿（C5）	3	4	7			
	深蓝（C6）	2	0	2			
	紫（C7）	2	2	4			
	黑（C8）	0	2	2			
合计		24	30	54			

由表中信息可以得出，玻璃类型与纹饰对于风化程度均呈现显著性差异，而玻璃颜色对于风化程度统计不显著。因此，玻璃类型与纹饰对玻璃文物的风化程度具有显著影响，而颜色对风化程度的影响不明显。

Step2：岭回归

岭回归是一种能很好解决多重共线性的回归方法，它以牺牲部分信息和准确度为代价，换取更符合实际、更可靠的回归结果，能很好的拟合变态数据。岭回归实际是对最小二乘法的改良，在其基础上加入了惩罚项：

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda|\theta|_2^2=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2 \tag{2}$$

在上式中 $J(\theta)$ 为损失函数，λ 为惩罚系数，λ 越大，曲线拟合程度越低，忽略噪声的效果越好。建立回归模型，如下式所示：

$$Y=z\theta+\varepsilon \tag{3}$$

在上式中：Y 为因变量，z 为自变量，θ 为回归系数，ε 为误差项。岭回归系数 $\theta^{(k)}$ 的定量表达式为：

$$\widehat{\theta_{(k)}}=(Z^TZ+kl)^{-1}Z^TY=(Z^TZ+kl)^{-1}Z^T\widehat{\theta} \tag{4}$$

其中，l 为单位矩阵，k 是岭回归参数。k 的取值在 0-1 之间，当 k 取 0 时，即是最小二乘估计。

设定量化处理后的玻璃类型为 $T_i(i=1,2)$, 分别为铅钡玻璃和高钾玻璃；定量化处理后的玻璃纹饰为 $D_i(i=1,2,3)$, 分别为纹饰 A,B 和 C; 定量化处理后处理后的玻璃颜色为 $C_i(i=1,2,...,8)$, 分别对应由浅至深八种玻璃颜色。

同时，本文对表单 2 中的数据进行标准化处理，对岭回归系数表达式中的 Y 和 Z 都进行中心化和长度单位化处理，即：

$$Z_{ij} = \frac{X_{ij} - \bar{X}_j}{\sqrt{\sum (X_{ij} - \bar{x}_j^2)}} \tag{5}$$

运用 MATLAB 绘制岭迹图，其中横坐标表示岭回归参数，纵坐标表示岭回归系数。K 越大，消除共线性效果更好，但拟合方差越大，拟合精度越低，因此应在消除共线性和提高拟合精度的平衡中找到合适的 k 值。

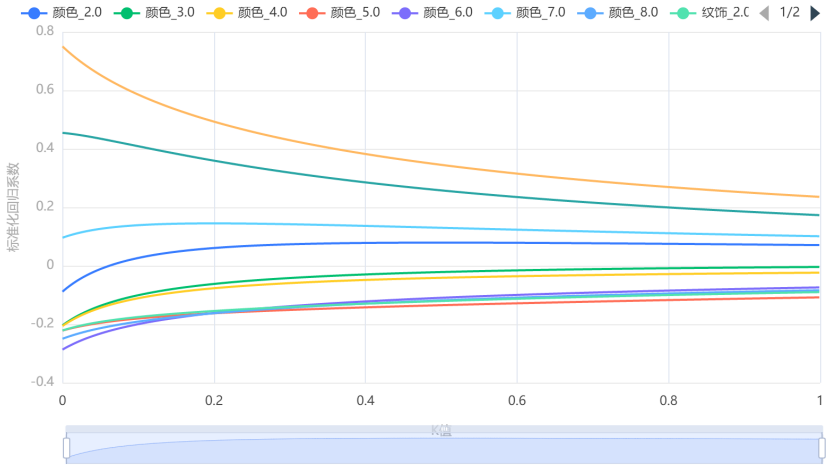


图 2: 岭回归

根据图可以观察到 k 增大时，曲线趋于平缓。根据方差扩大因子法，确定合适的 k 值为 0.144。模型的 F 统计量为 37.86，在显著性水平为 0.05 的水平下通过率检验，说明模型是有效的，调整后的 R^2 为 0.92，拟合优度较高，模型可以接受。岭回归拟合精度和模型参数如下：

表 4: 模型参数与拟合精度

模型参数	非标准化		标准化	T	显著性
	系数	标准误差	系数		
X1	0. 109	0. 020	0. 120	4. 518	0. 0011
X2	0. 144	0. 036	0. 120	2. 2508	0. 04812

R_{Mult}	R^2	R^2_{Arj}	标准差（SE）
0.997	0.994	0.977	0.118

根据上述分析，可得本小题的回归方程为：

$$\begin{aligned} W = {} & 27.013 + 1.974 * D_1 + 40.86 * D_2 - 15.821 * D_3 + 6.788 * T_1 \\ & +20.225 * T_2 + 16.834 * C_1 + 5.173 * C_2 + 15.416 * C_3 - 12.636 * C_4 \\ & -4.095 * C_5 + 8.405 * C_6 + 1.477 * C_7 - 3.563 * C_8 \end{aligned} \tag{6}$$

下图直观地展示了真实值与预测值之间的比较，从图中可以观察到模型具有良好的拟合效果。

图 3: 拟合效果展示

4.1.4 问题一第（2）问模型建立与求解

本文对文物样品风化前后化学成分变化情况进行汇总梳理，对化学成分特征指标变化规律进行总结，如下图所示，得出就铅钡类型玻璃而言，文物发生风化作用后，二氧化硅含量和氧化铅含量普遍下降，部分化学成分同时产生相应变动；就高钾类型玻璃而言，文物发生风化作用后，二氧化硅含量普遍上升，氧化钾含量普遍降低。

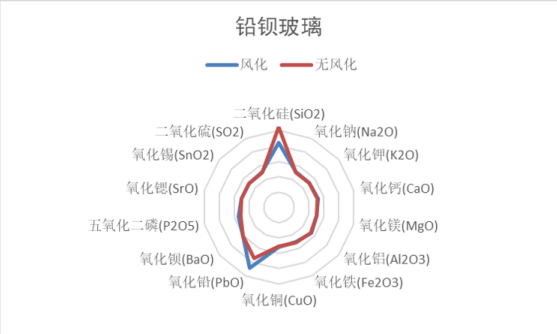


图 4: 铅钡玻璃

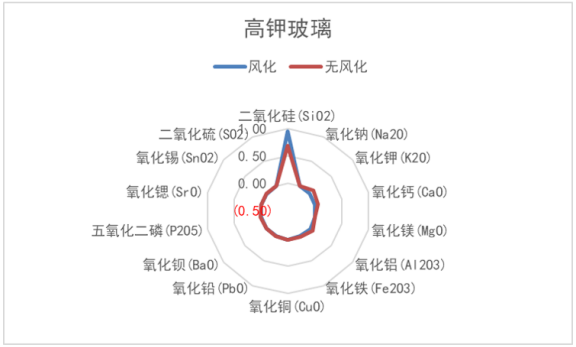


图 5: 高钾玻璃

基于此认识，本文结合第（1）问岭回归模型求解思路，采用岭回归对其他参数以及 14 个化学成分逐一进行回归，对风化点风化前的化学成分含量进行预测。下列为岭回归模型中有关各成分回归方程的岭迹图，篇幅有限，只展示部分回归可视化效果图，所有化学成分回归方程的岭迹图详见附录 2。

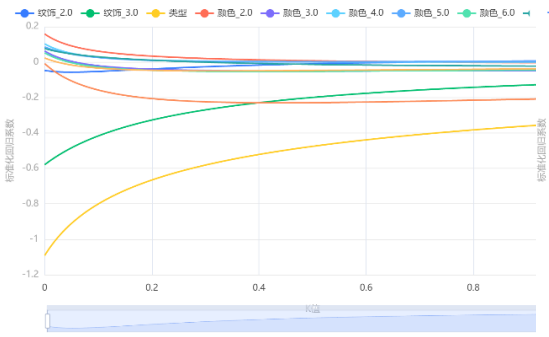


图 6: 岭迹图（氧化钾）

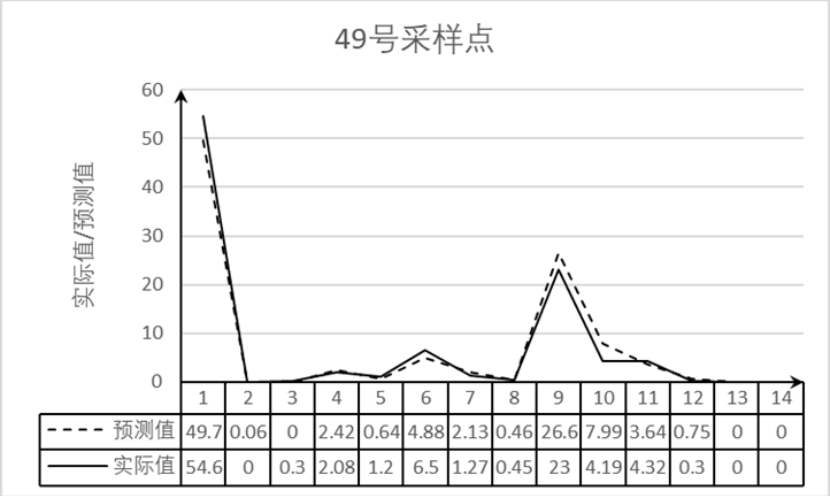
图 7: 岭迹图（二氧化硅）

以下为对无风化点风化前化学成分的预测，由于篇幅有限，此处仅展示 20 件文物的预测信息，全部玻璃文物制品化学成分预测信息详见附录 3。

表 5: 化学成分预测信息（部分）

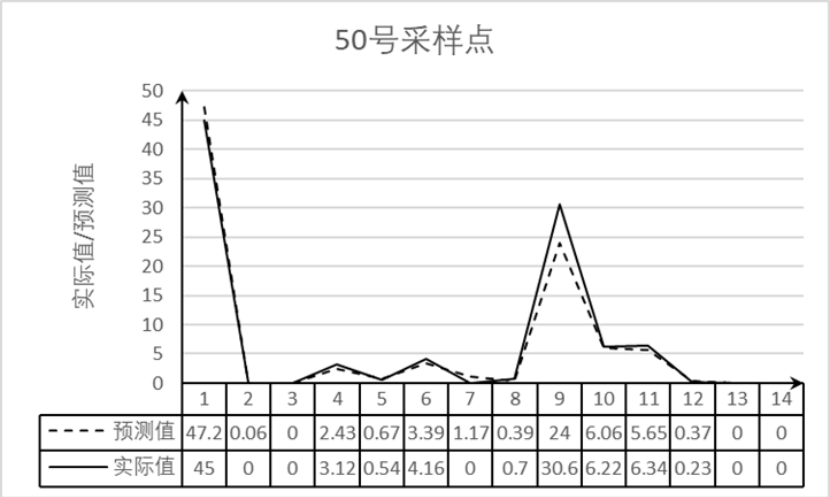
文物编号	二氧化硅 (Si02)	氧化钠 (Na20)	氧化钾 (K20)	氧化钙 (Ca0)	氧化镁 (Mg0)	氧化铝 (Al203)	氧化铁 (Fe203)	氧化铜 (Cu0)	氧化铅 (Pb0)	氧化钡 (Ba0)	五氧化二磷 (P205)	氧化锶 (Sr0)	氧化锡 (Sn02)	二氧化硫 (S02)
2	40.93	1.822	0	0.965	0.836	5.726	0.689	1.196	18.412	6.464	1.538	0.278	0.003	0
7	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
8	13.878	0.008	0	1.339	0	0.68	0.704	5.794	23.003	25.261	3.037	0.495	0	5.523
8	13.878	0.008	0	1.339	0	0.68	0.704	5.794	23.003	25.261	3.037	0.495	0	5.523
9	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
10	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
11	23.327	0.888	0.04	1.916	0.376	1.654	0.482	1.361	34.755	6.257	3.698	0.351	0	0
12	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
19	40.93	1.822	0	0.965	0.836	5.726	0.689	1.196	18.412	6.464	1.538	0.278	0.003	0
22	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
23	36.857	0.839	0	1.006	0.655	4.035	1.156	1.827	24.474	10.459	1.059	0.2	0	0
25	23.327	0.888	0.04	1.916	0.376	1.654	0.482	1.361	34.755	6.257	3.698	0.351	0	0
26	13.878	0.008	0	1.339	0	0.68	0.704	5.794	23.003	25.261	3.037	0.495	0	5.523
26	13.878	0.008	0	1.339	0	0.68	0.704	5.794	23.003	25.261	3.037	0.495	0	5.523
27	89.594	0	0.15	0.055	0	0.84	0.347	1.206	0	0	0	0	0	0
28	40.93	1.822	0	0.965	0.836	5.726	0.689	1.196	18.412	6.464	1.538	0.278	0.003	0
29	40.93	1.822	0	0.965	0.836	5.726	0.689	1.196	18.412	6.464	1.538	0.278	0.003	0
34	33.321	0.879	0.107	0.909	0.026	1.791	0.039	0.727	33.584	7.505	0.094	0.256	0	0.102
36	33.321	0.879	0.107	0.909	0.026	1.791	0.039	0.727	33.584	7.505	0.094	0.256	0	0.102

为检测模型预测效果，针对未纳入样本的 49 和 50 两组风化前后的采样点数据进行误差分析，可以看到误差均控制在合理范围 (如下图所示)，拟合效果良好。



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
预测值	49.7	0.06	0	2.42	0.64	4.88	2.13	0.46	26.6	7.99	3.64	0.75	0	0
实际值	54.6	0	0.3	2.08	1.2	6.5	1.27	0.45	23	4.19	4.32	0.3	0	0

表 6: 49 号采样点预测



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
预测值	47.2	0.06	0	2.43	0.67	3.39	1.17	0.39	24	6.06	5.65	0.37	0	0
实际值	45	0	0	3.12	0.54	4.16	0	0.7	30.6	6.22	6.34	0.23	0	0

表 7: 50 号采样点预测

4.2 问题二

4.2.1 问题分析

探究玻璃类型的分类规律即是要求充分运用表单 1 与表单 2 的相关数据对铅钡玻璃与高钾玻璃之间差异性进行探究，同时提出针对铅钡玻璃和高钾玻璃中的亚分类的划分方法，并得出分类结果，最终对结果的合理性与敏感性需要做出检验。

4.2.2 数据预处理

首先，针对表单 2 中玻璃文物化学成分含量等信息，根据题干中文物样本成分比例累加合计应居于 85%-105% 的区间，所以应将 19 号与 21 号两组不符合题意的无效数据进行剔除。然后针

对清洗后的数据进行归一化处理，处理公式为：

$$x_i^* = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{7}$$

4.2.3 问题二模型的建立与求解

对得到的 63 份玻璃文物化学成分比例样本数据进行 KMO 检验和 Bartlett 球形检验，结果显示 KMO 检验系数为 0.541，高于 0.5，适合做主成分分析，Bartlett 球形检验显著性水平低于 0.01，因此拒绝零假设，说明变量间存在较强相关性，主成分分析适用。

KMO值		0.541
Bartlett 球形度检验	近似卡方	421.263
	df	45
	p 值	0

表 8: KMO 和 Bartlett 的检验

由于某些化学成分空值太多或含量太少，我们对部分化学成分进行剔除后，对 63 份玻璃文物化学成分比例指标进行主成分分析，可得主成分分析的方差解释率和累计方差解释率。

方差解释率						
编号	特征根			主成分提取		
	特征根	方差解释率%	累积%	特征根	方差解释率%	累积%
1	3.779	37.792	37.792	3.779	37.792	37.792
2	2.043	20.43	58.222	2.043	20.43	58.222
3	1.506	15.063	73.285	1.506	15.063	73.285
4	0.779	7.789	81.074	0.779	7.789	81.074
5	0.587	5.872	86.945	–	–	–
6	0.465	4.648	91.594	–	–	–
7	0.385	3.849	95.443	–	–	–
8	0.245	2.454	97.897	–	–	–
9	0.195	1.953	99.849	–	–	–
10	0.015	0.151	100	–	–	–

表 9: 方差解释率

根据上表可以看出，主成分 1 的特征根为 3.779，方差解释率为 37.792，主成分 2 的特征根为 2.043，方差解释率为 20.43，主成分 3 的特征根为 1.506，方差解释率为 15.063。前 4 个主成分的累计方差解释率达到 81.074%，能够比较全面反映玻璃文物的化学成分特性。

名称	载荷系数				共同度(公因子方差)
	主成分1	主成分2	主成分3	主成分4	
二氧化硅(SiO2)	0.7	-0.658	-0.035	0.125	0.941
氧化钾(K2O)	0.726	0.22	0.41	-0.27	0.816
氧化钙(CaO)	0.474	0.637	0.281	-0.447	0.909
氧化镁(MgO)	0.431	0.573	-0.372	0.33	0.76
氧化铝(Al2O3)	0.654	0.357	-0.096	0.411	0.734
氧化铁(Fe2O3)	0.55	0.497	0.038	0.111	0.563
氧化铜(CuO)	-0.266	0.13	0.854	0.264	0.886
氧化铅(PbO)	-0.754	0.411	-0.374	-0.242	0.936
氧化钡(BaO)	-0.755	0.113	0.468	0.224	0.852
氧化锶(SrO)	-0.642	0.505	-0.147	0.149	0.711

备注：表格中数字若有颜色：蓝色表示载荷系数绝对值大于0.4，红色表示共同度(公因子方差)小于0.4。

表 10: 载荷系数

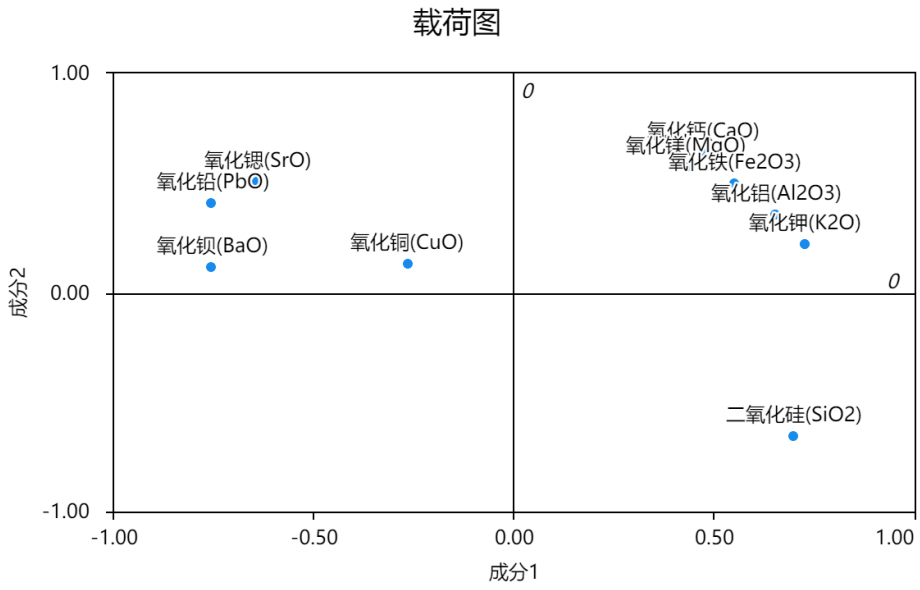


图 8: 载荷图

观察载荷图可以看到在一、二象限载荷集中分布，根据载荷矩阵可知第 1 主成分主要反映氧化钡、氧化铅、氧化钾、二氧化硅的特性，第 2 主成分主要反映氧化硅和氧化钙的特性，第 3 主成分主要反映氧化铜的特性，第 4 主成分主要反映氧化钙和氧化铝的性质。

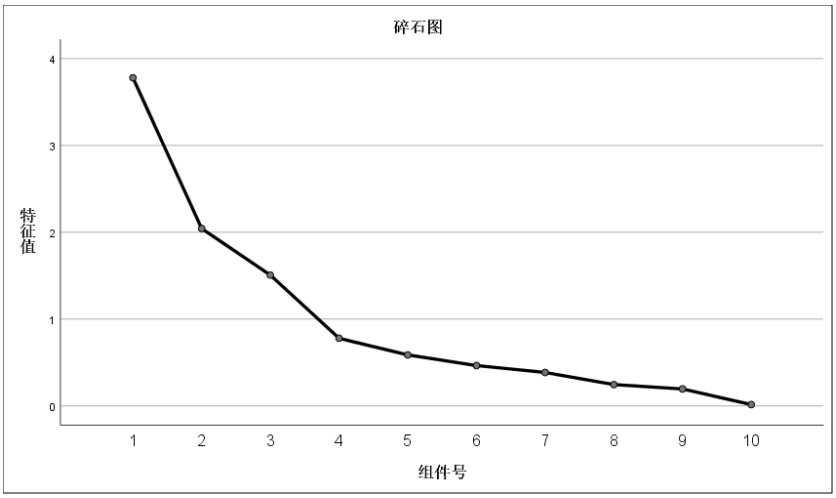


图 9: 碎石图

观察碎石图，特征值在分类为 1-4 处下降较陡，分类 4 以后下降趋于平缓，因此提取 4 个主成分是合理的。结合主成分的特征向量，可以得出主成分的方程：

$$\begin{aligned} F_1 = &-0.074SO_2 - 0.083P_2O_5 + 0.054SnO_2 - 0.17SrO - 0.191BaO \\ &+0.191SiO_2 - 0.206PbO - 0.077CuO + 0.123Fe_2O_3 + 0.156Al_2O_3 \\ &+0.097MgO + 0.113CaO + 0.184K_2O + 0.013Na_2O \end{aligned} \tag{8}$$

$$\begin{aligned} F_2 = &-0.081SO_2 + 0.282P_2O_5 + 0.022SnO_2 + 0.164SrO - 0.058BaO \\ &-0.23SiO_2 + 0.161PbO - 0.038CuO + 0.209Fe_2O_3 + 0.137Al_2O_3 \\ &+0.261MgO + 0.231CaO + 0.055K_2O - 0.12Na_2O \end{aligned} \tag{9}$$

$$\begin{aligned} F_3 = &\ 0.359SO_2 + 0.003P_2O_5 + -0.107SnO_2 - 0.052SrO + 0.215BaO \\ &-0.044SiO_2 - 0.154PbO + 0.45CuO - 0.154Fe_2O_3 + 0.45Al_2O_3 \\ &+0.129MgO + 0.026CaO + 0.055K_2O - 0.07Na_2O \end{aligned} \tag{10}$$

$$\begin{aligned} F_4 = &-0.095SO_2 + 0.047P_2O_5 + 0.023SnO_2 - 0.57SrO - 0.591BaO \\ &+0.791SiO_2 - 0.706PbO - 0.037CuO + 0.113Fe_2O_3 + 0.136Al_2O_3 \\ &+0.077MgO + 0.213CaO + 0.184K_2O + 0.083Na_2O \end{aligned} \tag{11}$$

由上可以得到：

$$F = (0.274/0.587)F_1 + (0.179/0.587)F_2 + (0.134/0.587)F_3 \tag{12}$$

针对高钾玻璃和铅钡玻璃分别做层次分析法，有如下结果，

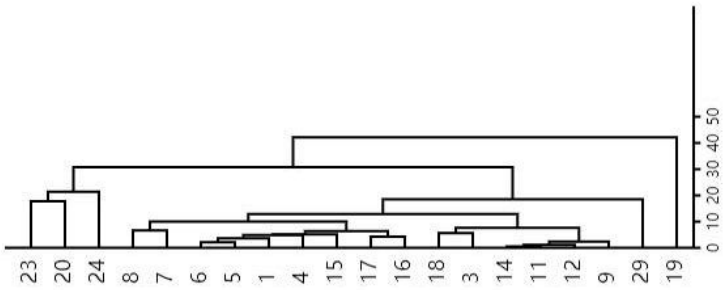


图 10: 高钾玻璃聚类树状图

样本类别	编号
第一类	20、24
第二类	19
第三类	1、4、5、6、7、8、15、16、17、23、29
第四类	3、9、11、12、14、18

图 11: 高钾玻璃分类

观察可知，第一类、第二类含有较高氧化钾，第三类含有较多的氧化钠，第四类中含有较多的氧化钠和氧化钾

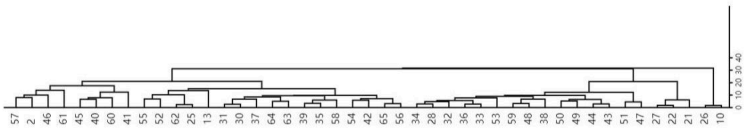


图 12: 铅钡玻璃聚类树状图

样本类别	编号
第一类	25、28、32、33、34、36、38、43、44、47、48、49、50、51、53、55、59、62、
第二类	21、22、27
第三类	2、13、30、31、35、37、39、40、41、42、45、46、52、54、56、57、58、60、61、63、64、65、
第四类	10、26

图 13: 铅钡玻璃分类

观察可知，第一类中含有较多的氧化铅和氧化钡，第二类中含有较多的氧化铝和氧化铅，第三类中含有较多的氧化铅，第四类中含有更高的氧化铜和氧化钡。综上所述，针对文物样品的亚分类如下所示：

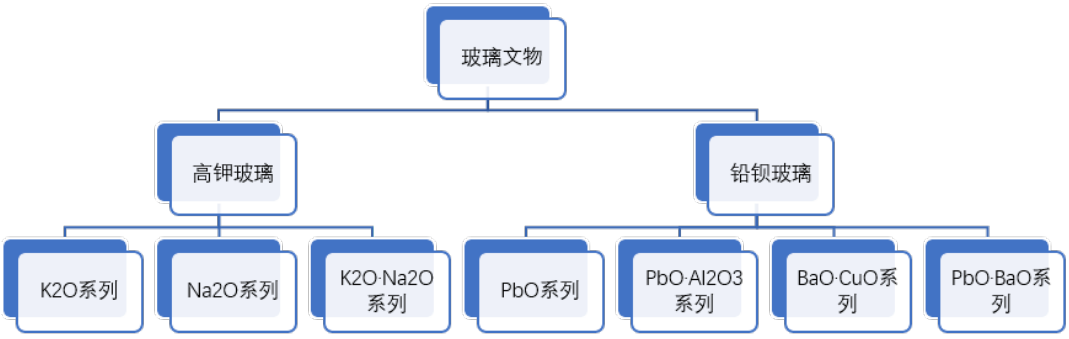


图 14: 分类结果

4.3 问题三

4.3.1 问题分析

基于第二问对古代玻璃制品类别归属进行了明确的判别，本文计划分别采用随机森林、AdaBoost 、XGboost 和 LightGBM 等分类预测算法对未知类别的的古代玻璃制品进行分类，以 F_1 值等作为重要标准，来选择最优分类预测模型，以期实现对未知玻璃制品的正确分类。对于分类结果，本文计划通过缩小样本容量的方式，变更特定指标的方式来检验预测分类模型的敏感性。

4.3.2 算法选择

为获得有效的分类结果，本文选择了随机森林、AdaBoost 、XGboost 等三种分类预测算法进行实证分析，计划在基于 F_1 值的评价模型中比较选择出最优的预测算法来完成玻璃制品分类的任务。

（1）随机森林

首先，本文针对表单 2 中已明确类别的玻璃制品集中抽取 70 % 的样本作为训练子集，并对训练子集进行决策树建模训练，然后利用测试集对各决策树进行测试，综合多棵决策树的测试结果，通过投票得出最终的用电量预测模型。本文随机森林预测模型如下图所示。

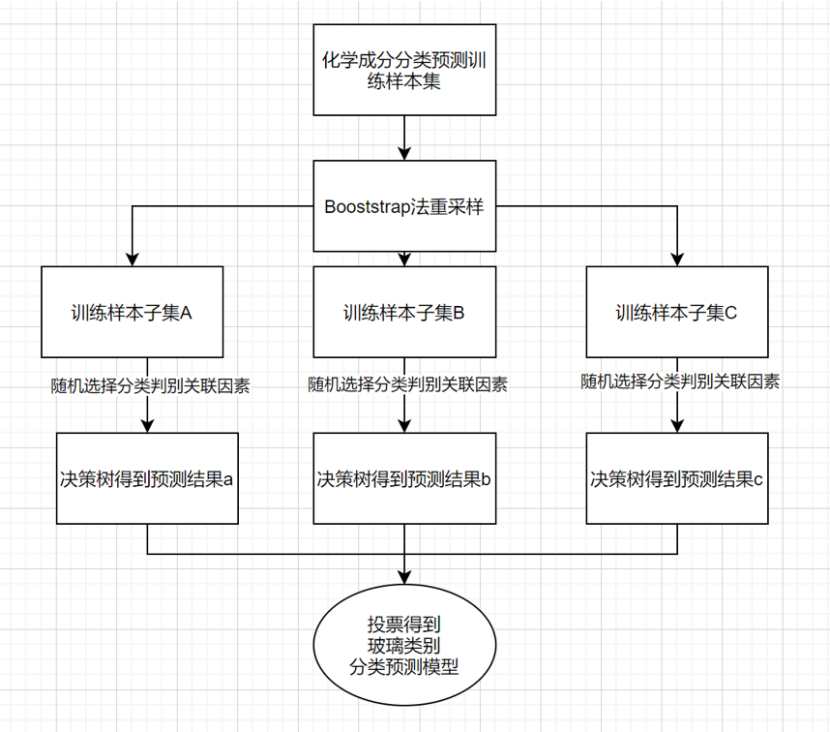


表 11: 基于随机森林算法的玻璃制品分类预测模型

（2）AdaBoost

AdaBoost 是 Boosting 族算法中较为典型的代表算法，AdaBoost 通过上一轮分类正确与否作为考量动态决定其权值，从而充分发挥判别学习能力强的学习器的作用。其分类器线性组合公式为：

$$\begin{cases} f(x) = \sum_{m=1}^{M} \alpha_m G_m(x), \\ G(x) = sign(f(x)). \end{cases} \tag{13}$$

其中 $f(x)$ 为所有学习器加权的结果，而 Gx 为样本的判定类别结果

（3）XGboost

XGboost 算法在 GBDT 算法的基础上进行优化，通过同时使用一阶导数和二阶导数并借鉴随机森林列抽样的方法，具有速度快，处理能力强等优良特性。其表达式为：

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i), f_k \in F, \tag{14}$$

其中，K 是判定类别的总数，F 为所有可能的结果，f 为其中一个具体的判定结果。

4.3.3 算法比较

本文主要借助 F_1 值对化学成分分类预测模型进行比较分析。其中 F_1 值为精确率和召回率的调和平均数，各模型指标数据对比如下所示，

表 12: 不同算法指标对比

模型	准确率	召回率	精确率	F1
随机森林	0.9	0.9	0.912	0.891
AdaBoost	0.85	0.85	1	0.919
XGboost	1	1	1	1

由上可知，XGboost 的 F_1 值最高，在玻璃制品化学成分分类分类预测中具有较强的分类性能，因此，本文选择 XGboost 作为预测方法，对表单 3 中未分类的玻璃文物进行类型鉴定。

4.3.4 分类结果

XGboost 的预测分类结果如下表所示：

文物编号	主分类	文物编号	主分类
A1	高钾	A2	铅钡
A5	高钾	A3	铅钡
A6	高钾	A4	铅钡
A7	高钾	A8	铅钡

图 15: 分类结果

进一步，我们通过将二氧化硅含量上下波动 5% 的区间进行敏感性分析，由下图可知，模型对数据变动的敏感性不强，分类性能较好。

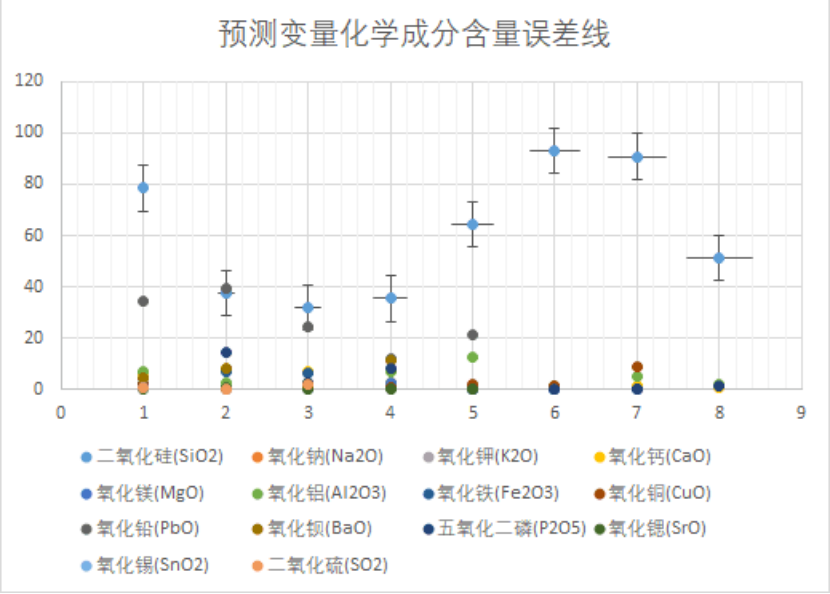


图 16: 化学成分含量误差线

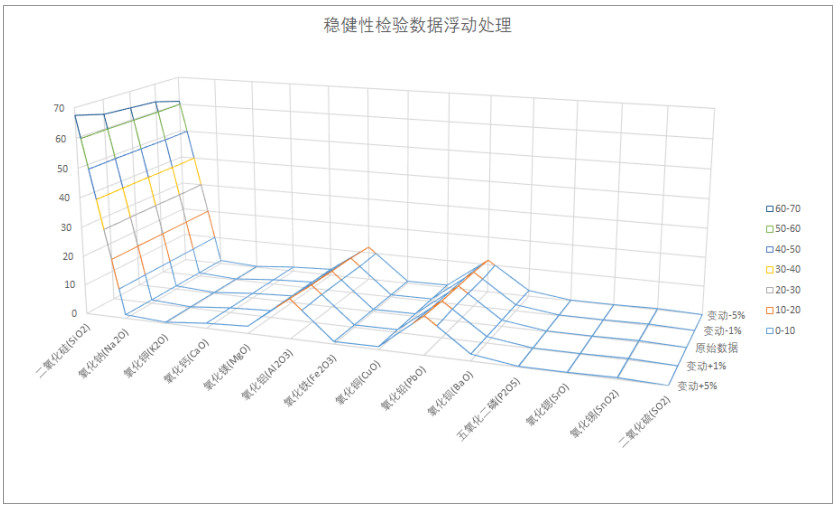


图 17: 稳健性检验数据浮动处理

4.4 第四问

4.4.1 问题分析

为研究玻璃文物化学成分的相关性以及差异性。本文主要借助单因素方差分析和 Pearson 相关系数法，对高钾和铅钡两种玻璃文物样品及其亚分类比较了他们之间不同化学成分含量的差异性，并且通过热力图分析了它们之间化学成分的关联关系。

4.4.2 差异性分析

单因素方差的前提是样本满足正态性检验。在大样本的前提下，化学元素含量分布可以认为服从正态分布。假设玻璃文物从大样本中抽取，玻璃文物化学成分的含量在自然状态下满足正态分布。对 7 种亚变量组化学成分含量分别做单因素方差分析，结果如下表。据表可知不同亚类的玻璃具有显著的差异 $p < 0,001$。

化学成分		平方和 (sum of squares	自由度 (degree	均方 (mean square)	F	显著水平 (significance) (P<0.001)
BaO・CuO系列	组间(between groups	10.616	3	3.539 0.093	37.976	0.00
	组内(within groups)	3.354	36			
	总数(total)	13.97	39			
				1.037 0.006	171.055	0.00
PbO系列	组间(between groups	3.111	3			
	组内(within groups)	0.218	36			
	总数(total)	3.33	39	3.161	53.133	0.00
PbO・Al2O3系列	组间(between groups)	9.484	3			
	组内(within groups)	2.142	36	0.059		
	总数(total)	11.625	39	0.003	12.376	0.00
PbO・BaO系列	组间(between groups	0.008	3			
	组内(within groups)	0.007	36	0		
	总数(total)	0.015	39			
Na2O系列	组间(between groups)	0.323	3	0.108 0.001	155.674	0.00
	组内(within groups)	0.025	36			
	总数(total)	0.348	39			
K2O・Na2O系列	组间(between groups	0.204	3	0.068 0.001	49.252	0.00
	组内(within groups)	0.05	36			
	总数(total)	0.254	39			
				0.000 0.000	4.54	0.01
K2O系列	组间(between groups	0.001	3			
	组内(within groups)	0.002	36			
	总数(total)	0.003	39	0.000 0.000	19.505	0.00

表 13: 差异性分析

4.4.3 相关性分析

对高钾玻璃组内做 14 种化学成分含量的相关性分析，得热力图如下。从图可知，高钾玻璃组内氧化钙和氧化钾具有较强的正关联性，氧化钡和氧化铜、氧化锶具有较强的正关联性，氧化铅和二氧化硅具有较强的负相关性。

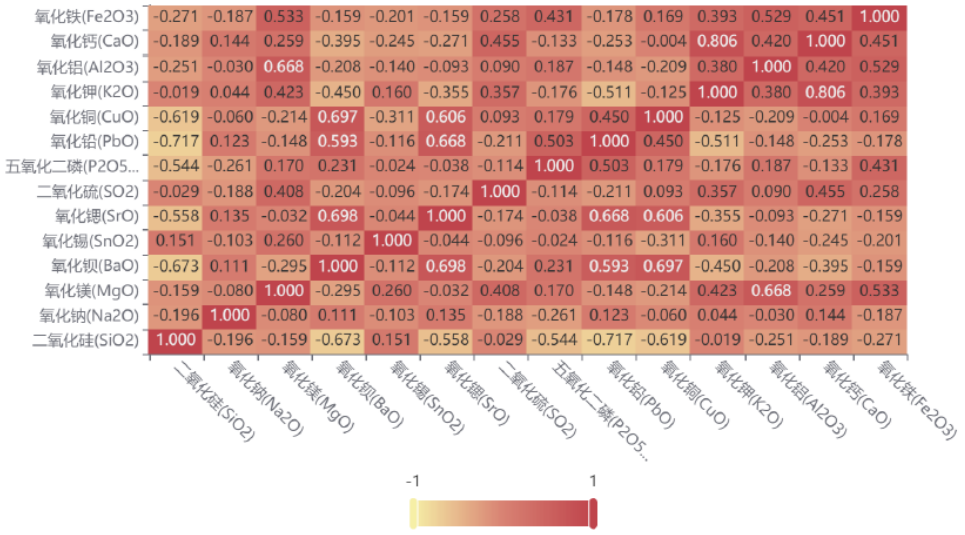


图 18: 相关性分析（1）

对铅钡玻璃组内进行 14 种化学成分含量的相关性分析，得热力图如下。分析图表可知铅钡玻

璃组内氧化钡、二氧化硫和氧化铜具有较强的正关联性，氧化铝和氧化锡具有较强的正关联性，氧化铅和二氧化硅具有较强的负相关性。

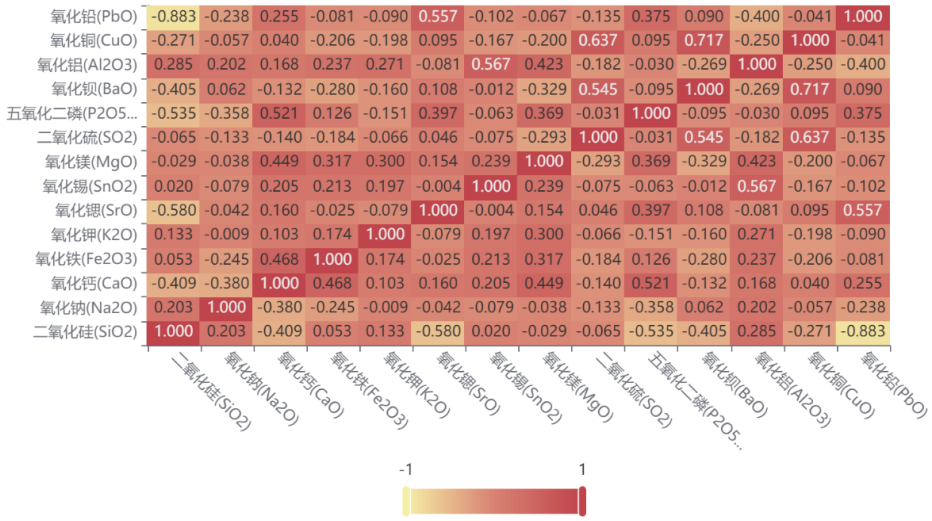


图 19: 相关性分析（2）

五、模型的综合评价与改进

5.1 模型的优点

（1）**问题**一通过采用岭回归的方法预测玻璃文物在风化前的化学成分含量，相较于传统的线性回归方法进行预测，本文充分考虑到由于风化过程中文物内部元素与外界环境元素产生置换而导致文物化学成分出现异常值的可能性，预测结果更具准确性与科学性；

（2）**问题**二采用主成分分析与层次分析相结合的方法，通过对表单 2 冗杂的数据进行降维凝练，在有效解决定类问题的同时避免了由于文物化学成分复杂所可能带来的多重共线性，分类模型可靠性强。

（3）**问题**三通过四种分类预测模型的比较分析，最终选择针对玻璃化学成分分类性能最强的 XGboost 算法开展对未知类别玻璃文物的分类预测，整体可行性强，为后期玻璃文物分类鉴别工作具有极强的参考意义。

5.2 模型的缺点

（1）**问题**二出于对实验结果的考量，对部分化学成分指标进行了筛选，忽略了这部分数据可能带来的影响，不够全面；

（2）**问题四**中针对化学成分之间的相关性时方法较为单一，可以进行模型的进一步优化。

5.3 模型的改进

（1）对玻璃制品化学成分的判别可以考虑采用其余的模型进行解决；

（2）在分析化学成分之间的关联性与差异性时可以考虑进一步考虑埋藏环境等外界因素的影响，使模型更加贴合现实。

参考文献

[1] 李越, 龚铭. 多元统计分析方法在古陶瓷鉴定中的应用 [J]. 文物鉴定与鉴赏,2019(06):46-57+75.

[2] 伏修锋, 干福熹. 基于多元统计分析方法对一批中国南方和西南地区的古玻璃成分的研究 [J]. 文物保护与考古科学,2006(04):6-13.DOI:10.16334/j.cnki.cn31-1652/k.2006.04.002.

[3] 王爱民, 宋强, 李华, 张运素, 徐蕾. 小样本贫信息条件下高炉冶炼烧结终点组合预测法 [J]. 重庆大学学报,2011,34(05):123-129.

[4] 张文宽, 尹国龙, 徐洪恩, 杨本锦. 关于利用岩石化学成分预测岩浆型铜镍 (铂) 矿方法的讨论 [J]. 四川地质学报,2006(04):248-251.

[5] 王宇斌, 彭祥玉, 张小波, 李帅, 朱新峰. 从高铅玻璃中回收铅的试验研究 [J]. 矿产综合利用,2016(04):90-93.

附 录

附录 1：古代玻璃制品编码数据

文物编号	纹饰	类型	颜色	表面风化	文物编号	纹饰	类型	颜色	表面风化
1	3	1	3	0	28	1	2	2	1
2	1	2	2	1	29	1	2	6	0
3	1	1	3	0	30	3	2	7	0
4	1	1	3	0	31	3	2	1	0
5	1	1	3	0	32	3	2	5	0
6	1	1	3	0	33	3	2	5	1
7	2	1	3	1	34	3	2	1	0
8	3	2	7	1	35	3	2	5	1
9	2	1	3	1	36	3	2	5	0
10	2	1	3	1	37	3	2	5	1
11	3	2	2	1	38	3	2	5	1
12	2	1	3	1	39	3	2	1	1
13	3	1	2	0	40	1	2	2	1
14	3	1	5	0	41	3	2	2	1
15	3	1	2	0	42	1	2	2	1
16	3	1	2	0	43	1	2	2	0
17	3	1	2	0	44	1	2	2	0
18	1	1	6	0	45	1	2	2	0
19	1	2	2	0	46	1	2	8	1
20	1	1	3	0	47	1	2	8	1
21	2	1	3	1	48	3	2	2	1
22	1	2	3	1	49	3	2	2	1
23	3	2	7	0	50	1	2	2	1
24	3	2	2	1	51	3	2	2	1
25	3	2	7	1	52	3	2	4	0
26	2	1	3	1	53	3	2	3	1
27	1	2	2	1	54	3	2	3	1

附录 2：文物化学成分比重预测

文物编号	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)
2	40.93	1.822	0	0.965	0.836	5.726	0.689
7	89.594	0	0.15	0.055	0	0.84	0.347
8	13.878	0.008	0	1.339	0	0.68	0.704
8	13.878	0.008	0	1.339	0	0.68	0.704
9	89.594	0	0.15	0.055	0	0.84	0.347
10	89.594	0	0.15	0.055	0	0.84	0.347
11	23.327	0.888	0.04	1.916	0.376	1.654	0.482
12	89.594	0	0.15	0.055	0	0.84	0.347
19	40.93	1.822	0	0.965	0.836	5.726	0.689
22	89.594	0	0.15	0.055	0	0.84	0.347
23	36.857	0.839	0	1.006	0.655	4.035	1.156
25	23.327	0.888	0.04	1.916	0.376	1.654	0.482
26	13.878	0.008	0	1.339	0	0.68	0.704
26	13.878	0.008	0	1.339	0	0.68	0.704
27	89.594	0	0.15	0.055	0	0.84	0.347
28	40.93	1.822	0	0.965	0.836	5.726	0.689
29	40.93	1.822	0	0.965	0.836	5.726	0.689
34	33.321	0.879	0.107	0.909	0.026	1.791	0.039
36	33.321	0.879	0.107	0.909	0.026	1.791	0.039
38	33.321	0.879	0.107	0.909	0.026	1.791	0.039
39	33.321	0.879	0.107	0.909	0.026	1.791	0.039
40	23.327	0.888	0.04	1.916	0.376	1.654	0.482
41	34.782	0.059	0	1.697	0.848	1.816	0.782
42	40.93	1.822	0	0.965	0.836	5.726	0.689
42	40.93	1.822	0	0.965	0.836	5.726	0.689
43	23.327	0.888	0.04	1.916	0.376	1.654	0.482
43	23.327	0.888	0.04	1.916	0.376	1.654	0.482
44	40.93	1.822	0	0.965	0.836	5.726	0.689
48	40.93	1.822	0	0.965	0.836	5.726	0.689
51	23.327	0.888	0.04	1.916	0.376	1.654	0.482
51	23.327	0.888	0.04	1.916	0.376	1.654	0.482
52	23.327	0.888	0.04	1.916	0.376	1.654	0.482
53	40.93	1.822	0	0.965	0.836	5.726	0.689
54	23.327	0.888	0.04	1.916	0.376	1.654	0.482
54	23.327	0.888	0.04	1.916	0.376	1.654	0.482
56	19.254	0	0	1.957	0.195	0	0.949
57	19.254	0	0	1.957	0.195	0	0.949
58	23.327	0.888	0.04	1.916	0.376	1.654	0.482

问题一第（1）问代码：

```
import pandas as pd
import numpy as np
import warnings
```

```
import os
from spsspro.algorithm import descriptive_analysis
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font='simhei')
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
data1=pd.read_excel('附件.xlsx',sheet_name='表单1')
data1_process = data1.dropna(axis=0,how="any")
data1_process.to_csv("processed.csv",encoding="utf-8",index=False)
data=pd.read_csv("processed.csv")

data = data1[["纹饰","类型","颜色","表面风化"]]
data ["纹饰"] = LabelEncoder().fit_transform(np.array(data["纹饰"])).reshape(-1,1)
data ["表面风化"] = LabelEncoder().fit_transform(np.array(data["表面风化"])).reshape(-1,1)
data ["类型"] = LabelEncoder().fit_transform(np.array(data["类型"])).reshape(-1,1)
data ["颜色"] = LabelEncoder().fit_transform(np.array(data["颜色"])).reshape(-1,1)

print(data.corr('spearman'))
spearman_matrix = data.corr('spearman')
fig = plt.figure()
ax  = fig.add_subplot(1,1,1)
ax.set_title('相关性分析热力图')
ax.set_xlabel('')
ax.set_ylabel('')
figure1=sns.heatmap(spearman_matrix, annot=True)
s1 = figure1.get_figure()
s1.savefig('HeatMap.pdf',dpi=600,bbox_inches='tight')
```

问题一第（2）问代码：

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import os
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from spsspro.algorithm import statistical_model_analysis

data=pd.read_excel('附件.xlsx',sheet_name='表单1')
data.head()
data_merge.columns
```

```
data_merge['sum']=0
for i in ['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)']:
data_merge['sum']+=data_merge[i]
data_merge[(data_merge['sum']<85)|(data_merge['sum']>105)]
data_merge=data_merge[(data_merge['sum']>85)&(data_merge['sum']<105)]
data_merge.reset_index(inplace=True,drop=True)
X=pd.get_dummies(data_merge[['纹饰', '类型', '颜色', '表面风化']])
for i in ['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)']:
y=data_merge[i]
result = statistical_model_analysis.linear_regression(data_y=y, data_x1=X)
print('\n')
print(i)
print(result['linear_regression_form']['r2'])
print(result['linear_regression_analysis_description'])#进行岭回归

X1=data_merge[data_merge['表面风化']=='风化'][['文物编号', '纹饰', '类型', '颜色', '表面风化']]
X1.reset_index(inplace=True,drop=True)
X2=X.loc[data_merge[data_merge['表面风化']=='风化'].index][X.columns[:-1]]
=27.054+1.926*X2['纹饰_A']+40.805*X2['纹饰_B']+(-15.677)*X2['纹饰_C']+6.598*X2['类型_铅钡']
+20.456*X2['类型_高钾']+16.807*X2['颜色_浅绿']
+5.352*X2['颜色_浅蓝']+15.346*X2['颜色_深绿']+(-12.637)*X2['颜色_深蓝']
+(-4.097)*X2['颜色_紫']+8.352*X2['颜色_绿']+1.279*X2['颜色_蓝绿']+(-3.348)*X2['颜色_黑']
X1['二氧化硅(SiO2)']=y.to_list()

y=0.818+0.566*X2['纹饰_A']-0.563*X2['纹饰_B']-0.368*X2['纹饰_C']
-0.137*X2['类型_铅钡']+0.137*X2['类型_高钾']-0.254*X2['颜色_浅绿']
+0.575*X2['颜色_浅蓝']+0.566*X2['颜色_深绿']-1.115*X2['颜色_深蓝']
-0.305*X2['颜色_紫']+2.197*X2['颜色_绿']-0.408*X2['颜色_蓝绿']
-1.185*X2['颜色_黑']
X1['氧化钠(Na2O)']=y.to_list()

y=0.858+2.694*X2['纹饰_A']+(-4.903)*X2['纹饰_B']+3.067*X2['纹饰_C']
+(-4.237)*X2['类型_铅钡']+5.095*X2['类型_高钾']+0.088*X2['颜色_浅绿']
+0.352*X2['颜色_浅蓝']+0.419*X2['颜色_深绿']+0.722*X2['颜色_深蓝']
+(-0.039)*X2['颜色_紫']+(-0.151)*X2['颜色_绿']+(-0.9)*X2['颜色_蓝绿']+0.366*X2['颜色_黑']
X1['氧化钾(K2O)']=y.to_list()

y=0.761+1.332*X2['纹饰_A']+(-2.854)*X2['纹饰_B']+2.283*X2['纹饰_C']
+(-1.237)*X2['类型_铅钡']+1.998*X2['类型_高钾']+(-0.11)*X2['颜色_浅绿']
+0.109*X2['颜色_浅蓝']+(-0.898)*X2['颜色_深绿']+1.029*X2['颜色_深蓝']
```

```
+(-0.468)*X2['颜色_紫']+(-0.623)*X2['颜色_绿']+0.15*X2['颜色_蓝绿']+1.571*X2['颜色_黑']

X1['氧化钙(CaO)']=y.to_list()

y=0.217+0.654*X2['纹饰_A']+(-0.63)*X2['纹饰_B']+0.194*X2['纹饰_C']
+(-0.18)*X2['类型_铅钡']+0.398*X2['类型_高钾']+0.617*X2['颜色_浅绿']
+0.145*X2['颜色_浅蓝']+(-0.205)*X2['颜色_深绿']+0.154*X2['颜色_深蓝']
+(-0.238)*X2['颜色_紫']+(-0.201)*X2['颜色_绿']+(-0.036)*X2['颜色_蓝绿']+(-0.019)*X2['颜色_黑']
X1['氧化镁(MgO)']=y.to_list()

y=1.389+3.821*X2['纹饰_A']+(-2.181)*X2['纹饰_B']+(-0.251)*X2['纹饰_C']
+(-0.709)*X2['类型_铅钡']+2.098*X2['类型_高钾']+1.387*X2['颜色_浅绿']
+1.225*X2['颜色_浅蓝']+1.362*X2['颜色_深绿']+(-1.98)*X2['颜色_深蓝']
+0.251*X2['颜色_紫']+0.724*X2['颜色_绿']+(-0.466)*X2['颜色_蓝绿']+(-1.114)*X2['颜色_黑']
X1['氧化铝(Al2O3)']=y.to_list()


y=0.266+0.625*X2['纹饰_A']+(-0.777)*X2['纹饰_B']+0.418*X2['纹饰_C']
+(-0.163)*X2['类型_铅钡']+0.43*X2['类型_高钾']+0.261*X2['颜色_浅绿']+(-0.039)*X2['颜色_浅蓝']
+(-0.482)*X2['颜色_深绿']+0.346*X2['颜色_深蓝']+0.183*X2['颜色_紫']
+(-0.869)*X2['颜色_绿']+0.428*X2['颜色_蓝绿']+0.439*X2['颜色_黑']
X1['氧化铁(Fe2O3)']=y.to_list()


y=0.572+0.539*X2['纹饰_A']+(-0.67)*X2['纹饰_B']+0.704*X2['纹饰_C']
+(-0.008)*X2['类型_铅钡']+0.58*X2['类型_高钾']+(-1.378)*X2['颜色_浅绿']+0.093*X2['颜色_浅蓝']
+(-0.541)*X2['颜色_深绿']+(-1.515)*X2['颜色_深蓝']+4.526*X2['颜色_紫']
+(-0.625)*X2['颜色_绿']+0.724*X2['颜色_蓝绿']+(-0.713)*X2['颜色_黑']
X1['氧化铜(CuO)']=y.to_list()


y=5.161+(-1.254)*X2['纹饰_A']+(-8.675)*X2['纹饰_B']+15.089*X2['纹饰_C']
+15.59*X2['类型_铅钡']+(-10.429)*X2['类型_高钾']+(-7.648)*X2['颜色_浅绿']+(-1.085)*X2['颜色_浅蓝']
+(-2.256)*X2['颜色_深绿']+18.635*X2['颜色_深蓝']+(-12.837)*X2['颜色_紫']
+0.886*X2['颜色_绿']+4.977*X2['颜色_蓝绿']+4.489*X2['颜色_黑']
X1['氧化铅(PbO)']=y.to_list()


y=2.192+1.825*X2['纹饰_A']+(-1.251)*X2['纹饰_B']+1.618*X2['纹饰_C']
+5.599*X2['类型_铅钡']+(-3.407)*X2['类型_高钾']+(-3.556)*X2['颜色_浅绿']+(-3.152)*X2['颜色_浅蓝']
+(-1.904)*X2['颜色_深绿']+(-0.302)*X2['颜色_深蓝']+15.852*X2['颜色_紫']
+(-2.028)*X2['颜色_绿']+0.843*X2['颜色_蓝绿']+(-3.561)*X2['颜色_黑']
X1['氧化钡(BaO)']=y.to_list()


y=0.547+0.223*X2['纹饰_A']+(-2.059)*X2['纹饰_B']+2.383*X2['纹饰_C']+0.213*X2['类型_铅钡']
```

```
+0.334*X2['类型_高钾']+(-0.364)*X2['颜色_浅绿']+0.555*X2['颜色_浅蓝']+(-3.049)*X2['颜色_深绿']
+0.733*X2['颜色_深蓝']+(-0.106)*X2['颜色_紫']+(-1.959)*X2['颜色_绿']
+0.076*X2['颜色_蓝绿']+4.662*X2['颜色_黑']
X1['五氧化二磷(P2O5)']=y.to_list()



y=0.063+(-0.006)*X2['纹饰_A']+0.003*X2['纹饰_B']+0.067*X2['纹饰_C']+0.157*X2['类型_铅钡']
+(-0.094)*X2['类型_高钾']+(-0.159)*X2['颜色_浅绿']+0.064*X2['颜色_浅蓝']
+(-0.031)*X2['颜色_深绿']+0.147*X2['颜色_深蓝']+0.208*X2['颜色_紫']+(-0.31)*X2['颜色_绿']
+(-0.014)*X2['颜色_蓝绿']+0.158*X2['颜色_黑']
X1['氧化锶(SrO)']=y.to_list()

y=0.082+0.166*X2['纹饰_A']+(-0.122)*X2['纹饰_B']+0.039*X2['纹饰_C']
+(-0.124)*X2['类型_铅钡']+0.206*X2['类型_高钾']+(-0.001)*X2['颜色_浅绿']+(-0.121)*X2['颜色_浅蓝']
+(-0.102)*X2['颜色_深绿']+0.904*X2['颜色_深蓝']+(-0.076)*X2['颜色_紫']
+0.074*X2['颜色_绿']+(-0.319)*X2['颜色_蓝绿']+(-0.278)*X2['颜色_黑']
X1['氧化锡(SnO2)']=y.to_list()

y=0.341+0.704*X2['纹饰_A']+(-0.775)*X2['纹饰_B']+0.412*X2['纹饰_C']+
(-0.368)*X2['类型_铅钡']+0.709*X2['类型_高钾']+(-0.326)*X2['颜色_浅绿']+
(-1.137)*X2['颜色_浅蓝']+(-0.283)*X2['颜色_深绿']+(-0.522)*X2['颜色_深蓝']
+5.138*X2['颜色_紫']+0.13*X2['颜色_绿']+(-1.129)*X2['颜色_蓝绿']+
(-1.531)*X2['颜色_黑']
X1['二氧化硫(SO2)']=y.to_list()

X1['sum']=0
for i in ['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)']:
X1['sum']+=X1[i]

for i in ['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)']:
X1[i]=X1[i].clip(lower=0)

del X1['sum']
X1.to_excel('未风化文物风化前的化学成分含量.xlsx',index=None)#进行岭回归预测并且得到结果
```

问题二代码：

```
    #运用了层次聚类的算法，在excel表单中先对数据进行处理
```

```
import numpy
import pandas
from spsspro.algorithm import statistical_model_analysis
#生成案例数据
data_x1 = pandas.DataFrame({
    "A": numpy.random.random(size=100),
    "B": numpy.random.random(size=100)
})
data_y = pandas.Series(data=numpy.random.random(size=100), name="Y")
result = statistical_model_analysis.ridge_regression(y=data_y, x1=data_x1)
print(result)#利用spss求得岭回归的若干参数和岭回归的系数

#聚类
import numpy
import pandas
from spsspro.algorithm import statistical_model_analysis
#生成案例数据
data = pandas.DataFrame({
    "A": numpy.random.random(size=20),
    "B": numpy.random.random(size=20)
})
result = statistical_model_analysis.cluster_analysis(data, cluster_num=3)
print(result)

#将高钾和铅钡分成两组，输出相应的层次聚类树状图

del X1['二氧化硅（SiO2']
import numpy
import pandas
from spsspro.algorithm import statistical_model_analysis
#生成案例数据
data = pandas.DataFrame({
    "A": numpy.random.random(size=20),
    "B": numpy.random.random(size=20)
})
result = statistical_model_analysis.cluster_analysis(data, cluster_num=3)
print(result)

#进行敏感性检验
```

问题三代码：

```
#第三问
#使用了Adaboost，Random
    Forest，Xgboost，LightGBM进行分类预测，在第二问中得到了模型的准确率，召回率，精确率，算得综合系数F1，
```

```
结论：Xgboost，LightGBM预测效果较好，完全一致，其他两种方法有少许偏差，但也不大

#Adaboost
class DecisionTreeClassifierWithWeight:
def __init__(self):
self.best_err = 1 # 最小的加权错误率
self.best_fea_id = 0 # 最优特征id
self.best_thres = 0 # 选定特征的最优阈值
self.best_op = 1 # 阈值符号，其中 1: >, 0: <

def fit(self, X, y, sample_weight=None):
if sample_weight is None:
sample_weight = np.ones(len(X)) / len(X)
n = X.shape[1]
for i in range(n):
feature = X[:, i] # 选定特征列
fea_unique = np.sort(np.unique(feature)) # 将所有特征值从小到大排序
for j in range(len(fea_unique)-1):
thres = (fea_unique[j] + fea_unique[j+1]) / 2 # 逐一设定可能阈值
for op in (0, 1):
y_ = 2*(feature >= thres)-1 if op==1 else 2*(feature < thres)-1 # 判断何种符号为最优
err = np.sum((y_ != y)*sample_weight)
if err < self.best_err: # 当前参数组合可以获得更低错误率，更新最优参数
self.best_err = err
self.best_op = op
self.best_fea_id = i
self.best_thres = thres
return self

def predict(self, X):
feature = X[:, self.best_fea_id]
return 2*(feature >= self.best_thres)-1 if self.best_op==1 else 2*(feature <
    self.best_thres)-1

def score(self, X, y, sample_weight=None):
y_pre = self.predict(X)
if sample_weight is not None:
return np.sum((y_pre == y)*sample_weight)
return np.mean(y_pre == y)

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
X, y = load_breast_cancer(return_X_y=True)
y = 2*y-1 # 将0/1取值映射为-1/1取值
X_train, X_test, y_train, y_test = train_test_split(X, y)
DecisionTreeClassifierWithWeight().fit(X_train, y_train).score(X_test, y_test)
```

```
48
49      class AdaBoostClassifier_:
50      def __init__(self, n_estimators=50):
51      self.n_estimators = n_estimators
52      self.estimators = []
53      self.alphas = []
54
55      def fit(self, X, y):
56      sample_weight = np.ones(len(X)) / len(X) # 初始化样本权重为 1/N
57      for _ in range(self.n_estimators):
58      dtc = DecisionTreeClassifierWithWeight().fit(X, y, sample_weight) # 训练弱学习器
59      alpha = 1/2 * np.log((1-dtc.best_err)/dtc.best_err) # 权重系数
60      y_pred = dtc.predict(X)
61      sample_weight *= np.exp(-alpha*y_pred*y) # 更新迭代样本权重
62      sample_weight /= np.sum(sample_weight) # 样本权重归一化
63      self.estimators.append(dtc)
64      self.alphas.append(alpha)
65      return self
66
67      def predict(self, X):
68      y_pred = np.empty((len(X), self.n_estimators)) #
            预测结果二维数组，其中每一列代表一个弱学习器的预测结果
69      for i in range(self.n_estimators):
70      y_pred[:, i] = self.estimators[i].predict(X)
71      y_pred = y_pred * np.array(self.alphas) # 将预测结果与训练权重乘积作为集成预测结果
72      return 2*(np.sum(y_pred, axis=1)>0)-1 # 以0为阈值，判断并映射为-1和1
73
74      def score(self, X, y):
75      y_pred = self.predict(X)
76      return np.mean(y_pred==y)
77
78      from sklearn.ensemble import AdaBoostClassifier
79      AdaBoostClassifier_().fit(X_train, y_train).score(X_test, y_test)
80      AdaBoostClassifier().fit(X_train, y_train).score(X_test, y_test)
81
82      #随机森林 (Random Forest)
83      #构造特征集和标签集
84      x = rawdata.drop('isrun',axis = 1)
85      y = rawdata['isrun']
86
87      #划分出训练集和测试集
88      from sklearn.model_selection import train_test_split
89      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state=0)
90      x_train.shape#查看训练集数据量
91      x_test.shape#查看测试集数据量
92
```

```
#使用随机森林进行分类
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()#使用默认参数将随机森林分类器实例化
rfc.fit(x_train,y_train)#模型拟合

#模型效果评价
#评分方法1:
score1 = rfc.score(x_test,y_test)#查看拟合出的分类器在测试集上的效果
print(score1)

#评分方法2:
from sklearn.metrics import roc_auc_score
proba = rfc.predict_proba(x_test)#使用分类器预测测试集中每个样本属于0和1的概率
score2 = roc_auc_score(y_test,proba[:,1])
print(score2)

#评分方法3:
from sklearn.model_selection import cross_val_score
score3 = cross_val_score(rfc,x_train,y_train,scoring='accuracy',cv = 3)
print(score3)
print(score3.mean())#整体平均得分

#预测
rfc.predict(x_test)#使用分类器预测测试集的类别

#特征重要性
importance = rfc.feature_importances_#查看各个特征列的重要性
col = rawdata.columns#查看数据框的全部字段名（包括isrun）,返回格式为Index
import numpy as np
re = pd.DataFrame({'特征名':np.array(col)[:-1],'特征重要性':importance}).sort_values(by =
    '特征重要性',axis = 0,ascending = False)
print(re)

#参数调优
#先调n_estimators，即随机森林中树的棵数
from sklearn.model_selection import GridSearchCV
num_estimator =
    {'n_estimators':range(50,300,50)}#随机森林中树的棵数，以50为起点，50为步长，最多为300棵树
gs1 = GridSearchCV(estimator = rfc,param_grid = num_estimator,scoring = 'roc_auc',cv = 3)
gs1.fit(x_train,y_train)
print(gs1.best_estimator_)#查看最佳分类器对应的得分
print(gs1.best_score_)#查看最佳分类器对应的得分

#将n_estimators固定为最优值（200），然后再调树的最大深度max_depth
maxdepth = {'max_depth':range(3,10,1)}
gs2 = GridSearchCV(estimator = RandomForestClassifier(n_estimators = 200),param_grid =
```

```
    maxdepth,scoring = 'roc_auc',cv = 3)
gs2.fit(x_train,y_train)
print(gs2.best_estimator_)#查看最佳分类器
print(gs2.best_score_)#查看最佳分类器对应的得分

#max_depth=8, n_estimators=200固定不变, 继续调min_samples_split
minsamples = {'min_samples_split':range(2,50,2)}
gs3 = GridSearchCV(estimator = RandomForestClassifier(max_depth=8,
    n_estimators=200),param_grid = minsamples,scoring = 'roc_auc',cv = 3)
gs3.fit(x_train,y_train)
print(gs3.best_estimator_)#查看最佳分类器
print(gs3.best_score_)#查看最佳分类器对应的得分

#基于最优的参数进行预测
best_rfc = RandomForestClassifier(max_depth=8, min_samples_split=16,
    n_estimators=200)#使用最优参数对随机森林进行类的实例化
best_rfc.fit(x_train,y_train)#模型拟合
print(best_rfc.score(x_test,y_test))#查看best_rfc在测试集上的得分
best_rfc.predict(x_test)#对测试集样本进行分类

#LightGBM
from LightGBM import LGBMRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.externals import joblib

# 加载数据
iris = load_iris()
data = iris.data
target = iris.target

# 划分训练数据和测试数据
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2)

# 模型训练
gbm = LGBMRegressor(objective='regression', num_leaves=31, learning_rate=0.05,
    n_estimators=20)
gbm.fit(X_train, y_train, eval_set=[(X_test, y_test)], eval_metric='l1',
    early_stopping_rounds=5)

# 模型存储
joblib.dump(gbm, 'loan_model.pkl')
# 模型加载
gbm = joblib.load('loan_model.pkl')
```

```
178
179     # 模型预测
180     y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration_)
181
182     # 模型评估
183     print('The rmse of prediction is:', mean_squared_error(y_test, y_pred) ** 0.5)
184
185     # 特征重要度
186     print('Feature importances:', list(gbm.feature_importances_))
187
188     # 网格搜索，参数优化
189     estimator = LGBMRegressor(num_leaves=31)
190     param_grid = {
191         'learning_rate': [0.01, 0.1, 1],
192         'n_estimators': [20, 40]
193     }
194     gbm = GridSearchCV(estimator, param_grid)
195     gbm.fit(X_train, y_train)
196     print('Best parameters found by grid search are:', gbm.best_params_)
197
198     #xgboost
199     # 切分训练集和测试集
200     train_x, test_x, train_y, test_y =
            train_test_split(data,target,test_size=0.2,random_state=7)
201
202     # xgboost模型初始化设置
203     dtrain=xgb.DMatrix(train_x,label=train_y)
204     dtest=xgb.DMatrix(test_x)
205     watchlist = [(dtrain,'train')]
206
207     # booster:
208     params={'booster':'gbtree',
209         'objective': 'binary:logistic',
210         'eval_metric': 'auc',
211         'max_depth':5,
212         'lambda':10,
213         'subsample':0.75,
214         'colsample_bytree':0.75,
215         'min_child_weight':2,
216         'eta': 0.025,
217         'seed':0,
218         'nthread':8,
219         'gamma':0.15,
220         'learning_rate' : 0.01}
221
222     # 建模与预测: 50棵树
```

```
bst=xgb.train(params,dtrain,num_boost_round=50,evals=watchlist)
ypred=bst.predict(dtest)

# 设置阈值、评价指标
y_pred = (ypred >= 0.5)*1
print ('Precesion: %.4f' %metrics.precision_score(test_y,y_pred))
print ('Recall: %.4f' % metrics.recall_score(test_y,y_pred))
print ('F1-score: %.4f' %metrics.f1_score(test_y,y_pred))
print ('Accuracy: %.4f' % metrics.accuracy_score(test_y,y_pred))
print ('AUC: %.4f' % metrics.roc_auc_score(test_y,ypred))

ypred = bst.predict(dtest)
print("测试集每个样本的得分\n",ypred)
ypred_leaf = bst.predict(dtest, pred_leaf=True)
print("测试集每棵树所属的节点数\n",ypred_leaf)
ypred_contribs = bst.predict(dtest, pred_contribs=True)
print("特征的重要性\n",ypred_contribs )

xgb.plot_importance(bst,height=0.8,title='重要特征', ylabel='特征')
plt.rc('font', family='Arial Unicode MS', size=14)
plt.show()

#然后上下浮动二氧化硅的含量5%，进行灵敏度分析
```

问题四代码：

```
#对高钾和铅钡进行单因素方差分析和相关性检验
#系统
import os

#可视化包
import seaborn as sns
from IPython.display import Image
import pydotplus
import matplotlib.pyplot as plt
sns.set()
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
plt.rcParams['font.sans-serif'] = ['simhei']
plt.rcParams['font.serif'] = ['simhei']
sns.set_style("darkgrid",{"font.sans-serif":['simhei','Droid Sans Fallback']})

#数学包
from scipy import stats
import pandas as pd
import numpy as np
```

```
20
21      #统计检验
22      from statsmodels.formula.api import ols
23      from statsmodels.stats.anova import anova_lm
24      from statsmodels.stats.multicomp import MultiComparison
25
26      #演示数据集
27      from sklearn import datasets
28
29      #读取
30      iris=datasets.load_iris()
31      df_0 = pd.DataFrame(iris.data,
            columns=['SpealLength','Spealwidth','PetalLength','Petalwidth'])
32      df_0['target']=iris.target
33
34      # 封装 双变量-单因素方差分析
35      def my_oneWayAnova(df,cata_name,num_name,alpha_anova=0.05,alpha_tukey=0.05):
36
37      df[cata_name]=df[cata_name].astype('str')
38
39      s1=df[cata_name]
40      s2=df[num_name]
41
42      fml=num_name+'~C('+cata_name+')'
43
44      model = ols(fml,data=df).fit()
45      anova_table_1 = anova_lm(model, typ = 2).reset_index()
46      p1=anova_table_1.loc[0,'PR(>F)']
47
48      #输出 ： 是否相等【不等式序列】
49      if p1>alpha_anova:
50      print('组间【无】显著差异')
51      else:
52      print('组间【有】显著差异')
53      #输出不等式
54
55
56      # 输出： 统计结果表（均值，分位数，差异组）
57      df_p1=df.groupby([cata_name])[num_name].describe()
58
59      # 输出： Tudey 多重比较
60      mc = MultiComparison(df[num_name],df[cata_name])
61      df_smry = mc.tukeyhsd(alpha=alpha_tukey).summary()
62      m = np.array(df_smry.data)
63      df_p2 =pd.DataFrame(m[1:],columns=m[0])
64
```

```
    #输出 ： 分类直接的大小差异显著性
    df_p1_sub=df_p1[['mean']].copy()
    df_p1_sub.sort_values(by='mean',inplace=True)

    output_list=[]

    for x in range(1,len(df_p1_sub.index)):
    if (df_p2.loc[((df_p2.group1==df_p1_sub.index[x-1])&(df_p2.group2==df_p1_sub.index[x]))|
    ((df_p2.group1==df_p1_sub.index[x])&(df_p2.group2==df_p1_sub.index[x-1])),
    'reject'].iloc[0])=="True":
    smb='<'
    else:
    smb='<='
    if x==1:
    output_list.append(df_p1_sub.index[x-1])
    output_list.append(smb)
    output_list.append(df_p1_sub.index[x])
    else:
    output_list.append(smb)
    output_list.append(df_p1_sub.index[x])
    out_sentence=' '.join(output_list)
    print(out_sentence)


    # 输出： 箱线图
    #分布可视化boxplot
    plt.figure(figsize=(12, 8))
    sns.boxplot(x=cata_name, y=num_name,data=df)#,order=df_p1_sub.index

    return df_p1,df_p2


    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import scipy.stats as stats

    # 导入数据
    data = pd.DataFrame(np.random.randn(200,4)*100, columns = ['A','B','C','D'])
    # 相关性计算
    print(data.corr())
    # 绘图
    fig = pd.plotting.scatter_matrix(data,figsize=(6,6),c ='blue',marker =
        'o',diagonal='',alpha = 0.8,range_padding=0.2) # diagonal只能为'hist'/'kde'
    plt.show()
```