选题	2024 年第十四届 APMCM 亚太地区大学生数学建模竞赛（中文赛项）	参赛编号
B		Apmcm2410 1449

基于统计与机器学习的洪灾概率预测模型研究

摘　要

近年来，全球气候变化和人类活动的影响使得洪水灾害的频率和严重程度呈上升趋势，为了更有效地预防和减轻洪水灾害的影响，本文基于题目所给数据信息，建立数学模型进行分析，从而建立出合理的洪灾概率预测模型，提高我们对洪灾的预防能力。

针对问题一，首先计算出所给数据的偏度系数、峰度系数、变异系数等描述统计量。然后根据结果对数据进行缺失值异常值处理便于后续分析。接着引入 ***Spearman* 相关系数**进行初步分析发现每个特征列都与洪灾概率有一定相关度，于是采用 ***lasso* 特征选择**的方法进一步分析，最后我们结合了相关系数分析结果和最优的α值的 *lasso* 特征选择结果进行综合评价，结论为**侵蚀、排水系统、海岸脆弱性这些指标与洪水发生相关性不大，其余指标都与洪水发生相关性较高，最高者为基础设施恶化**。最后我们根据结果分析了原因并给出相关建议。

针对问题二，第一问首先对所有特征列进行 ***Kmeans*++聚类分析**然后采用小提琴图来对不同指标不同类别进行**差异性分析**，结果为在森林砍伐，基础设施恶化，人口得分这三个指标上，三个类别之间的差异性较大，分别是低风险，低风险，高风险较为突出。第二问构建**随机森林分类器**通过特征重要性结果来筛选合适的指标，我们选择特征重要性前五的指标再次训练随机森林模型，通过网格扩展进行参数优化后最优准确率为76%，最终敏感性分析结果为：**基础设施的恶化程度人口密度、人口结构或其他人口统计特征土地使用方式、农药使用或作物种植模式等因素对预测结果有较大的影响。**

针对问题三，首先引入了**互信息**的特征筛选方法，综合问题一以及互信息的结果选择出了合适指标，接着采用基于树的模型 *xgboost*，*catboost*，*LightGBM* 作为基线模型，综合考虑了 *Weighted Ensemble*，*Voting*，*Stacking* 等多种**集成学习**方法对洪水概率进行预测，选取 *MAPE* 作为评价指标，最优模型为 *catboost* 其 *MAPE* 为 0.066，接着使用 ***optuna* 超参数优化**方法进行优化调参，最终结果为 *Final MSE*：0.0024，*Final RMSE*:0.049，然后对模型进行 ***shap* 可解释性分析**包括摘要图、特征间交互作用图以及热力图，最终结论为如果需要在五个指标的情况下完美还原原模型的性能，需要更**详细的分析一些特征列之间的交互关系**，因为即使某些特征单独看起来不重要，它们与其他特征的组合可能对预测有显著影响。

针对问题四，首先使用问题三中的最优模型进行预测，然后从图像方面以及定量的 *KS* 检验方法综合分析了预测结果，最终结论为**预测结果近似正态分布**。

最后本文对所建立的模型进行了讨论和分析，综合评价了模型的优缺点。

关键词：概率预测；*K-means*++；特征选择；集成学习；*shap* 解释性分析；

一、 问题重述

1.1 问题背景

洪水是一种由暴雨、急剧融冰化雪、风暴潮等自然因素引起的自然灾害，其特点是江河湖泊的水量迅速增加或水位迅猛上涨。洪水不仅对人类社会的安全构成威胁，还可能造成巨大的经济损失和生态破坏。历史上，中国和世界其他地区都有多次严重的洪水灾害记载，这些灾害往往与人口增长、土地利用变化和森林砍伐等人为活动密切相关。近年来，全球气候变化和人类活动的影响使得洪水灾害的频率和严重程度呈上升趋势。例如，2023 年 6 月 24 日至 25 日，中国有 15 条河流发生了超警洪水，造成数十亿美元的经济损失。此外，长江上游的乱砍滥伐导致了严重的水土流失问题，淤积的泥沙量巨大，对河流和湖泊的生态环境造成了严重影响。为了更有效地预防和减轻洪水灾害的影响，我们需要通过数学建模和数据分析的方法来预测洪水发生的概率。这需要分析大量的洪水事件数据，包括各种可能影响洪水发生的因素，如季风强度、地形排水、河流管理、森林砍伐等。通过对这些数据的分析，我们可以识别出与洪水发生密切相关的关键指标，并建立预测模型来评估不同地区发生洪水的风险。

1.2 问题要求

问题 1. 请分析附件 train.csv 中的数据，分析并可视化上述 20 个指标中，哪些指标与洪水的发生有着密切的关联，哪些指标与洪水发生的相关性不大并，分析可能的原因，然后针对洪水的提前预防，提出合理的建议和措施。

问题 2. 将附件 train.csv 中洪水发生的概率聚类成不同类别，分析具有高、中、低风险的洪水事件的指标特征。然后，选取合适的指标，计算不同指标的权重，建立发生洪水不同风险的预警评价模型，最后进行模型的灵敏度分析。

问题 3. 基于问题 1 中指标分析的结果，请建立洪水发生概率的预测模型，从 20 个指标中选取合适指标，预测洪水发生的概率，并验证预测模型的准确性。如果仅用 5 个关键指标，如何调整改进洪水发生概率的预测模型。

问题 4. 基于问题 3 中建立的洪水发生概率的预测模型，预测附件 test.csv 中所有事件发生洪水的概率，并将预测结果填入附件 submit.csv 中。然后绘制这 74 多万件发生洪水的概率的直方图和折线图，分析此结果的分布是否服从正态分布。

二、 问题分析

2.1 问题一的分析

针对问题一，首先要对数据进行描述性统计，然后根据统计结果来处理数据的异常值和缺失值，接着使用斯皮尔曼相关系数进行初步的相关性分析，然后可以选择机器学习的方法来拟合数据，通过特征重要性结果来选取合适指标，最后阐述可能的原因以及提出建议。

2.2 问题二的分析

针对问题二，根据题目选择聚类数目为三类，然后使用 *kmeans*++聚类方法，并得出聚类中心，接着先使用全部特征列来构建随机森林分类器进行分类，通过特征重要性来筛选出合适的指标，然后使用超参数调优方法训练出最优模型进行分类预测并可视化结果，最后进行敏感度的分析。

2.3 问题三的分析

针对问题三，首先要综合考虑问题一的评价结果，可以引入别的方法再次进行合适指标的选取，通过综合分析选取方法的结果来划定最终合适指标，本文采用基于树

的模型 *xgboost*，*catboost*，*LightGBM* 作为基线模型，然后综合考虑多种集成学习的方法，通过交叉验证的 *MAPE* 来评判出最优模型，接着使用 *optuna* 超参数优化方法得出最优模型，通过 *shap* 进行可解释性分析，最后选取五个重要的指标进行最小化模型。

2.4 问题四的分析

针对问题四，采用问题三中的最优模型来进行预测，使用定量的检验方法 *KS* 检验以及图像综合分析来判断预测结果是否正态。

下面是我们的总体思路：

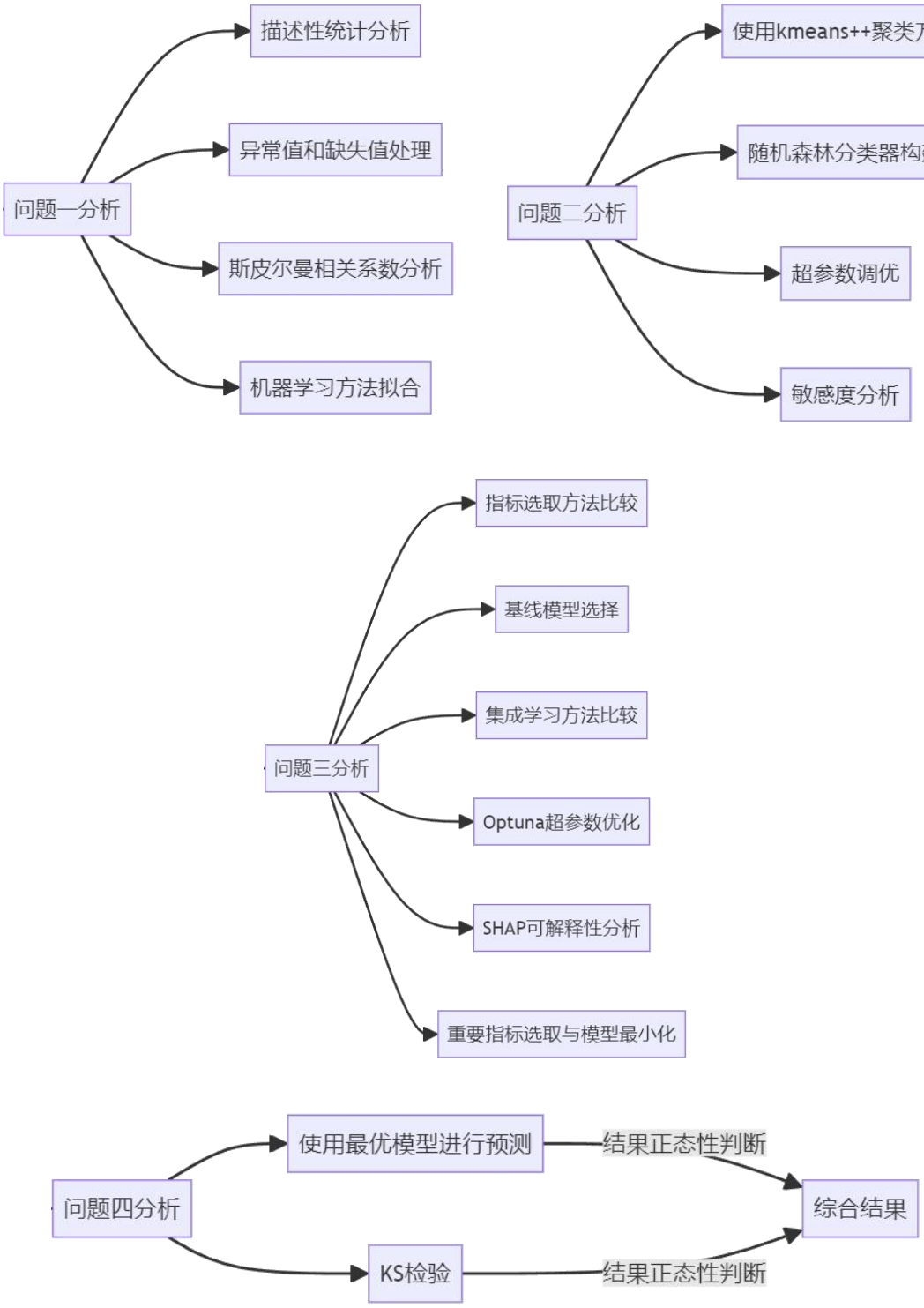

图 1 总体思路图

三、 模型假设

1. 假设通过超参数优化能够找到模型的最佳参数，且这些参数在整个超参数空间中是可探索的。
2. 假设通过 *SHAP* 值分析能够得到模型预测的可解释性，即能够识别出对预测结果影响最大的特征。
3. 假设所选用的评估指标（*MAPE*）能够全面反映模型的性能。

四、 符号说明

符号	意义
S_k	偏度系数
K_u	峰度系数
CV	变异系数
ρ	斯皮尔曼相关系数
β	*Lasso* 回归系数
c_1	初始聚类中心
y_i	决策树预测结果
$p(x,y)$	联合分布函数
$I(X;Y)$	X，Y之间的互信息
$obj(\theta)$	*Xgboost* 目标函数
γ_m	学习率
w	权重向量
$\mathrm{g}(\cdot)$	元模型函数
$\phi_j\big(F(x)\big)$	*Shap* 值

五、 问题一模型的建立与求解

5.1 数据预处理

5.1.1 探索性分析

我们首先对 *train.csv* 中的数据进行探索性分析，本文引入均值、最大值、最小值、中位数、标准差、偏度系数、峰度系数，变异系数来描述统计数据。

(1)偏度系数

偏度系数(*Skewness*)用于衡量数据分布的偏斜程度。$S_k = 0$ 表示数据近似对称分布，$S_k > 0$ 表示数据呈右偏分布，$S_k < 0$ 表示数据呈左偏分布。偏度系数的绝对值越大，数据分布的偏斜程度越明显。其计算公式如下：

$$S_k = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu^3}{\sigma^3}$$

其中S_k为偏度系数，*E(X)*为均值，μ^3为三阶中心距

(2)峰度系数

峰度系数(*Kurtosis*)用于衡量数据分布的峰度程度。$K_u = 0$ 表示数据分布为正态分布，$K_u < 0$ 表示数据分布的峰度较小，数据更分散，$K_u > 0$ 表示数据分布的峰度较大，数据更集中。峰度系数的绝对值越大，数据分布的峰度程度越明显。其计算公式如下：

$$K_u = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mu^4}{\sigma^4}$$

其中K_u为峰度系数，*E(X)*为均值，μ^4为三阶中心距

(3)变异系数

变异系数是一个统计学中的度量指标，用来衡量一组数据的离散程度与平均值的比例关系。它通过将标准差与均值进行比较，提供了一个无单位的度量方式，因此特别适用于比较不同单位或不同尺度数据的离散程度。其计算公式如下：

$$CV = \frac{\sigma}{\mu} \times 100\%$$

其中σ是数据的标准差，μ是数据的均值。

由于变量很多，所以我们这里只展示一部分：

表 1 部分变量统计学指标表

Column	*Mean*	*Std Dev*	*Kurtosis*	*Skewness*	*Coefficient of Variation*
季风强度	4.921994135	2.056183121	0.339480202	0.443775231	0.417754078
地形排水	4.926550795	2.093638159	0.240787398	0.459312377	0.424970379
河流管理	4.954774813	2.071778187	0.225043638	0.426512212	0.418137709
森林砍伐	4.942263548	2.051734804	0.27450746	0.43404529	0.415140711
城市化	4.942944472	2.083140401	0.246514022	0.441572467	0.421437144
气候变化	4.934179243	2.057920436	0.240905012	0.429192474	0.417074519
大坝质量	4.955645519	2.082146398	0.259995166	0.440933992	0.420156444
淤积	4.926949431	2.065527326	0.286055346	0.44842349	0.41923047

农业实践	4.942641204	2.067985801	0.211422856	0.419868546	0.418396909

从统计分析结果来看，我们可以对每个变量进行一些基本的解读：

Mean (均值): 所有变量的均值都接近 5，这表明这些变量的评分或测量值在 5 分的量表上处于中等偏上的水平。

Std Dev (标准差): 所有变量的标准差都在 2 左右，表明数据的分散程度相对一致，但存在一定的变异性。

Kurtosis (峰度): 除了 id 这一列的峰度为-1.2（表明数据分布比正态分布更加平坦），其他变量的峰度值接近 0，这表明这些变量的分布接近正态分布。

Skewness (偏度): 大部分变量的偏度接近 0.4 或-0.4，这表明数据分布有轻微的正偏或负偏。正偏度意味着数据的右尾更长，而负偏度意味着左尾更长。在这种情况下，大部分变量显示出轻微的正偏度。

Coefficient of Variation (变异系数): 所有变量的变异系数在 0.4 左右，这表明相对于均值，标准差的大小是相似的。

除了洪水概率和 id，其他列的统计特性相似，均值接近，标准差和变异系数也相似，表明这些变量在量表上的评分分布特性较为一致。

5.1.2 缺失值处理

接着我们统计出了每个变量的缺失值结果如下图：

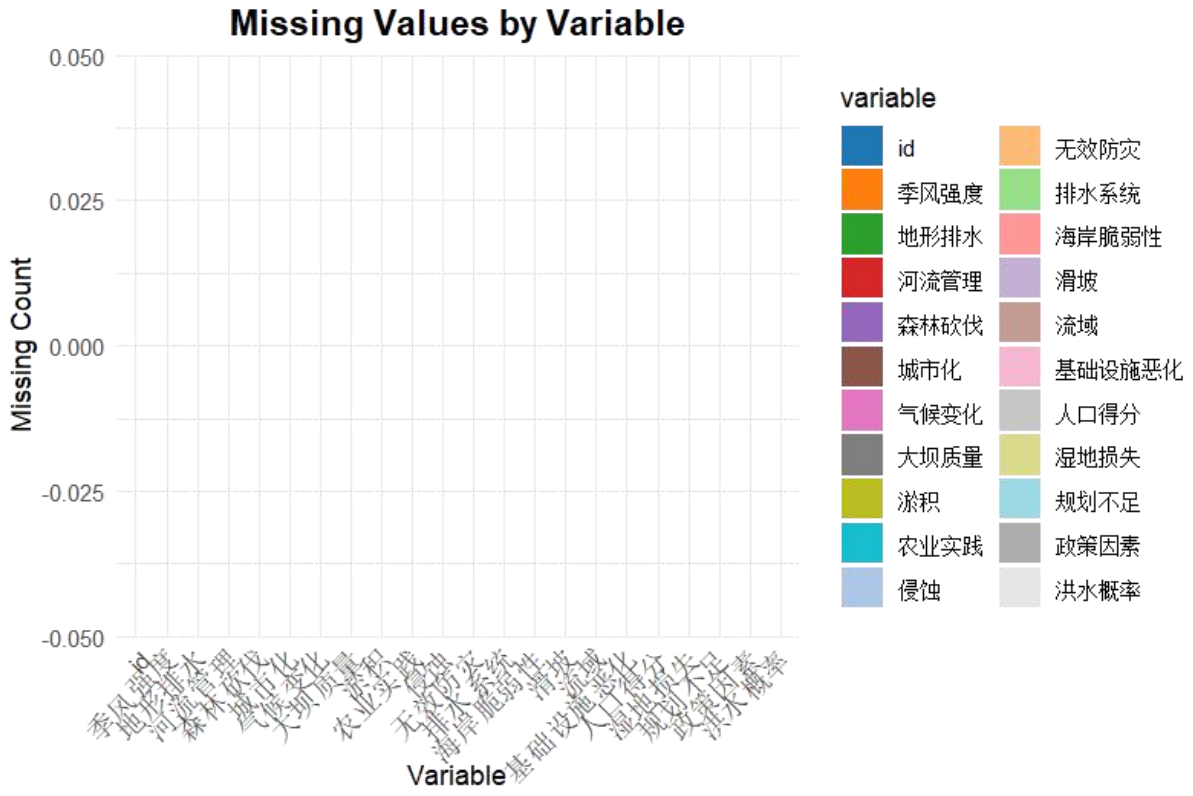

图 2 各变量缺失值统计图

从图 1 中我们可以看出，数据中不存在缺失值，有利于我们后续的数据处理。

5.1.3 异常值处理

由于我们的特征列大都近似正态分布，所以我们采用 3σ原则来进行异常值处理，3σ原则主要用于识别正态分布数据中异常的、处于极端值的数据点。我们可以用数学表达

式来准确地定义 3σ原则，对于一组数据集，假设其平均值为μ, 标准差为σ。则 3σ原则下的数据点范围可以定义如下：

下界：

$$\mu - 3\sigma$$

上界：

$$\mu + 3\sigma$$

这意味着根据 3σ原则，大约 99. 73%的数据点应该落在$\mu - 3\sigma$和$\mu + 3\sigma$这个区间内。任何超出这个范围的数据点(即小于$\mu - 3\sigma$或大于$\mu + 3\sigma$)被认为是异常值。

5. 2 特征选择

5. 2. 1 相关性分析

斯皮尔曼相关系数(*Spearman's rank correlation coefficient*)是一种非参数统计量，用于衡量两个变量之间的等级相关性。它适用于数据的等级表示方式或数据量级的比较，而不是精确的数值比较。斯皮尔曼相关系数可以用来分析两个变量之间的单调关系，即一个变量的增加会伴随另一个变量的增加或减少但不需要这两个变量之间的关系是线性的。 斯皮尔曼相关系数的数学表达式如下：

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

其中d_i是两个变量在排名上差异的绝对值，即$d_i = |r_1(i) - r_2(i)|$, 其中$r_1(i)$和$r_2(i)$分别是两个变量i点的排名，n是数据点的数量

我们计算了特征列与洪水概率之间的相关系，并绘制出了相关性热力图如下：

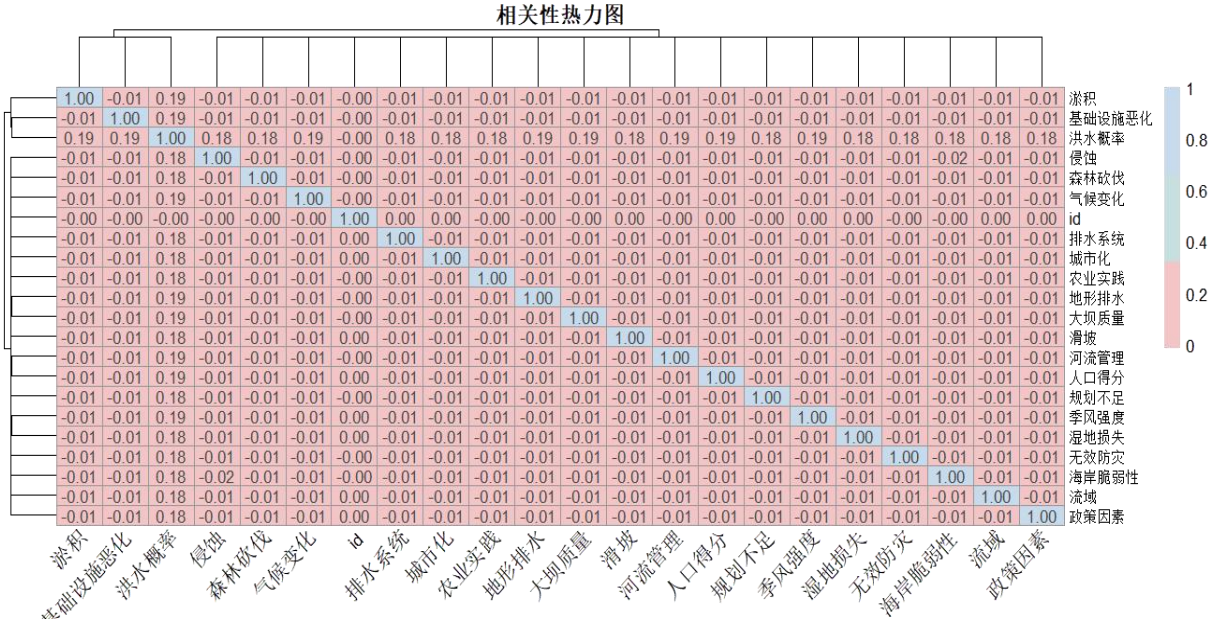

图 3 相关性热力图

结果分析：

基础设施恶化：与洪水概率的相关系数为 0.189065，这是一个正相关，表明基础设施恶化程度越高，洪水概率也越高。

季风强度、地形排水、大坝质量、河流管理、淤积、人口得分、滑坡：这些指标与洪水概率的相关系数都超过了 0.18，表明它们与洪水概率有中等程度的正相关。

气候变化、森林砍伐、湿地损失、农业实践、无效防灾：这些指标的相关系数在 0.18 到 0.15 之间，仍然可以认为是中等程度的正相关。

流域、政策因素、规划不足、城市化、侵蚀、排水系统：这些指标与洪水概率的相关系数在 0.15 到 0.1 之间，可以认为是低度到中等程度的正相关。

海岸脆弱性：与洪水概率的相关系数为 0.175964，略高于 0.15，可以认为是低度正相关。

5.2.2 *lasso* 特征选择

Lasso 回归是一种线性回归模型，它在最小化预测误差的同时，通过添加 *L1* 正则化项来实现特征选择。*Lasso* 回归主要用于处理变量选择问题，其特点在于惩罚项会使得一部分参数的值降为零，从而实现特征的稀疏表示。

Lasso 回归特征选择的原理：

Lasso 回归的目标函数可以表示为：

$$\text{minimize} \quad \frac{1}{2n} \| y - X\beta \|^2 + \lambda \sum_{i=1}^{p} |\beta_i|$$

L1 正则化的惩罚项：

$$\lambda \sum_{i=1}^{p} |\beta_i|$$

其导致回归系数β可能被变为零，这意味着一些特征被从模型中“选择”出来，而其他特征的系数则保持非零值。我们使用均方误差作为评估指标，目标是寻找最优的α值，λ^*:

$$\lambda^* = argmin_{\lambda} \frac{1}{K} \sum_{k=1}^{K} MSE(\lambda)$$

其中：$MSE(\lambda)$是在当前α值下，基于 K 折交叉验证的均方误差的平均值。

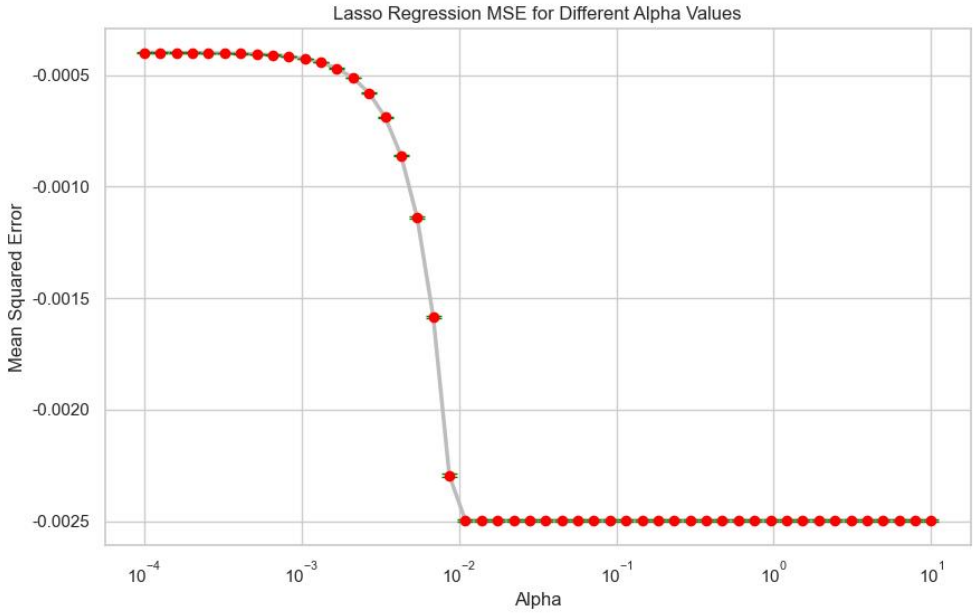

图 4 最优 alpha 选择结果图

最终结果为 ***Optimal alpha value***：0.0001，然后我们将最优值带回到原 *lasso* 模型进行特征选择，最终结果为：

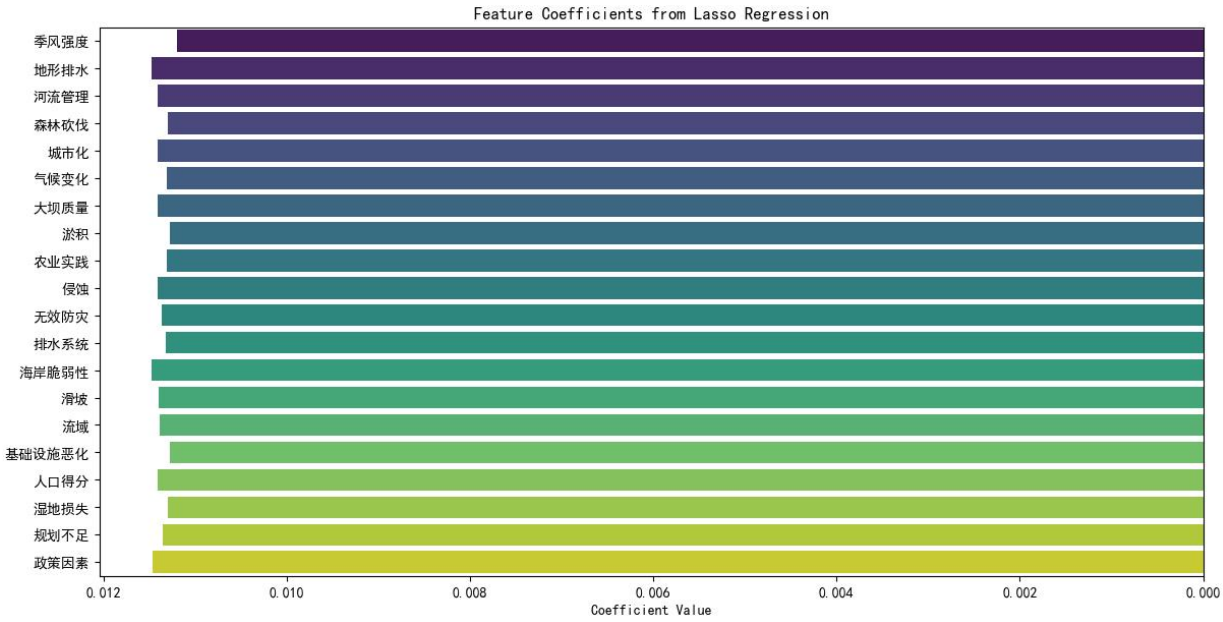

图 5 最终特征选择结果图

5.2.3 可视化拟合分析

最后我们拟合了特征列与洪水概率之间的关系，以季风强度与洪水概率为例，结果如下：

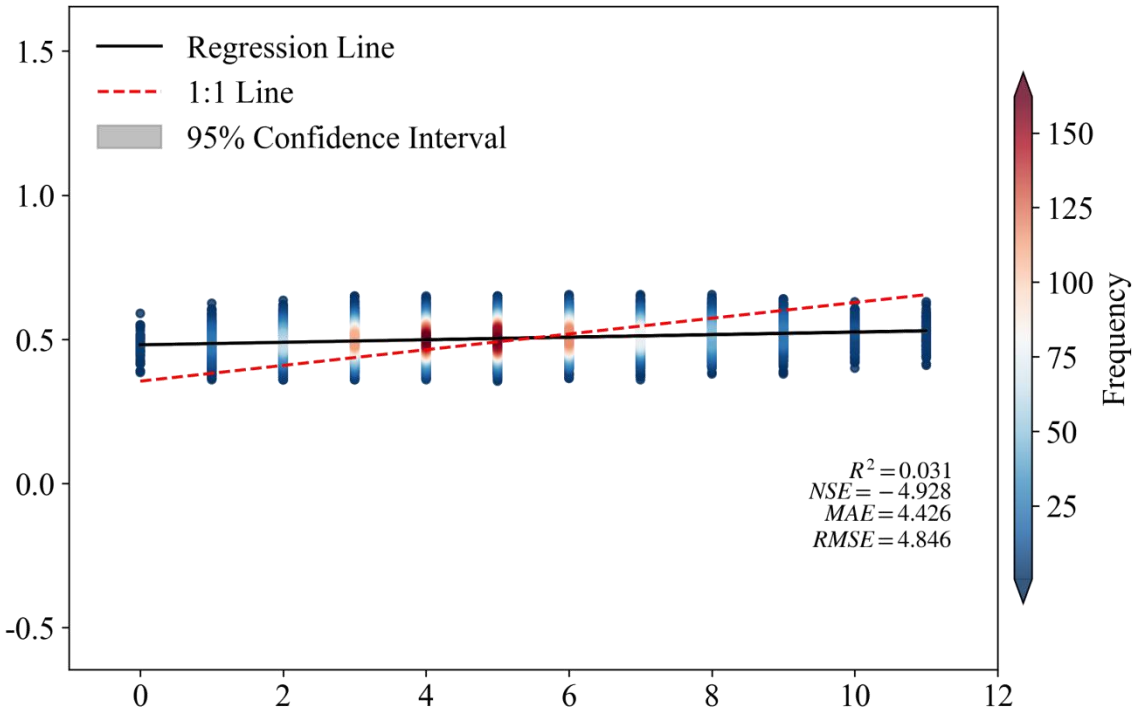

图 6 季风强度与洪水概率拟合结果图

从图中我们可以看出，二者间的相关性并不强，但是有一定的正相关性。

5.2.4 综合分析

根据斯皮尔曼相关系数、拟合分析以及 *lasso* 特征选择的结果，我们可以得出以下结论和建议：

与洪水发生高度相关的指标：

基础设施恶化：具有最高的相关系数，表明基础设施的恶化可能显著增加洪水发生的概率。这可能与排水系统失效、堤坝维护不足等因素有关。

季风强度、地形排水、河流管理、大坝质量、淤积、人口得分、滑坡、气候变化、森林砍伐：这些指标的相关系数都较高，表明它们与洪水发生有较强的联系。

城市化、农业实践、无效防灾、流域、湿地损失、政策因素、规划不足：这些指标虽然相关系数略低，但仍然显示出与洪水发生的正相关性。

与洪水发生相关性不大的指标：

id：相关系数接近 0，表明这个指标与洪水发生没有相关性。

侵蚀、排水系统、海岸脆弱性：虽然在 *Lasso* 回归中保留了这些指标，但它们在相关性分析中表现出的相关性较低。

可能的原因分析：

相关性高的指标可能直接影响洪水的发生，例如，季风强度增加降水量，地形排水条件差可能导致水无法快速排走，河流管理和大坝质量直接影响河流的控制能力。人口增长和城市化可能导致更多的地表覆盖，减少水的渗透，增加洪水的风险。森林砍伐和湿地损失减少了水的吸收和储存能力，可能导致洪水发生更加频繁。

预防措施建议：

1.加强基础设施建设与维护：改善排水系统，确保堤坝和水库的维护，以减少洪水的风险。2.河流管理：合理规划河流流域，减少泥沙淤积，提高河流的排水能力。3.森林和湿地保护：保护和恢复森林及湿地，增加水的吸收和储存能力。4.城市规划：在城市规划中考虑防洪措施，如建设防洪堤、蓄洪区等。5.气候变化适应：研究气候变化对洪水的影响，制定相应的适应策略。6.提高公众意识：通过教育和宣传提高公众对洪水灾害的认识和防范意识。

通过综合考虑相关性分析、拟合分析和 *Lasso* 回归结果，我们可以更准确地识别关键指标，并制定针对性的预防措施来减少洪水灾害的风险。

六、 问题二模型的建立与求解

6.1 *K-means*++聚类分析

根据问题一中的分析我们可以知道每个特征都与洪水概率有着一定的关系，所以我们在进行聚类分析特征选择时采用了所有特征列，我们采用 *K-means*++聚类算法，将洪水概率作为目标列：

K-means++算法是 *K-means* 算法的拓展，在初始化簇中心时，摒弃了 *K-means*++算 法随机初始化的思想，其选择初始化聚类中心的原则是初始化簇中心之间的相互距离 要尽可能地远。

K-means++算法的数学表达式和步骤：

算法步骤：

Step1）在数据集中随机选择一个样本点作为第一个初始聚类中心c_1；

Step2）计算距离：对于每个数据点x_i,计算它到目前为止选择的聚类中心的距离：

$$d(x_i, c_j)$$

Step3）选择后续中心：为每个数据点x_i,计算其到已选定中心的最小距离的平方：

$$d(x_i, c_j)^2$$

然后计算每个点的加权概率：

$$P(x_i) = \frac{d(x_i, c_j)^2}{\sum_j \ d(x_i, c_j)^2}$$

从数据集中以概率$P(x_i)$选择下一个聚类中心。

Step4）重复步骤 3 直到达到 *K* 个中心：重复步骤 3 直到选择 *K* 个中心。

Step5）*K-means* 阶段：使用初始化的 *K* 个中心，运行标准的 *K-means* 算法，直到收敛。

聚类结果如下：

表 2 聚类中心表

聚类种类	中心值_洪水概率
0	0.48177777777777775
1	0.5414285714285715
2	0.38333333333333336

表 3 聚类评价指标表

轮廓系数	DBI	CH
0.568	0.473	201.165

● 轮廓系数：对于一个样本集合，它的轮廓系数是所有样本轮廓系数的平均值。轮廓系数的取值范围是$[-1, 1]$，同类别样本距离越相近不同类别样本距离越远，分数越高，聚类效果越好。

● *DBI*：该指标用来衡量任意两个簇的簇内距离之后与簇间距离之比。该指标越小表示聚类效果越好。

● *CH*：通过计算类内各点与类中心的距离平方和来度量类内的紧密度（分母），通过计算类间中心点与数据集中心点距离平方和来度量数据集的分离度（分子），*CH* 指标由分离度与紧密度的比值得到，*CH* 越大表示聚类效果越好。

从聚类指标结果上看，我们的聚类效果还是较好的。

6.2 不同类别间特征差异分析

为了描述不同类别间的特征差异，我们绘制出了每个指标的针对每个类别的小提琴图来描述数据的分布：

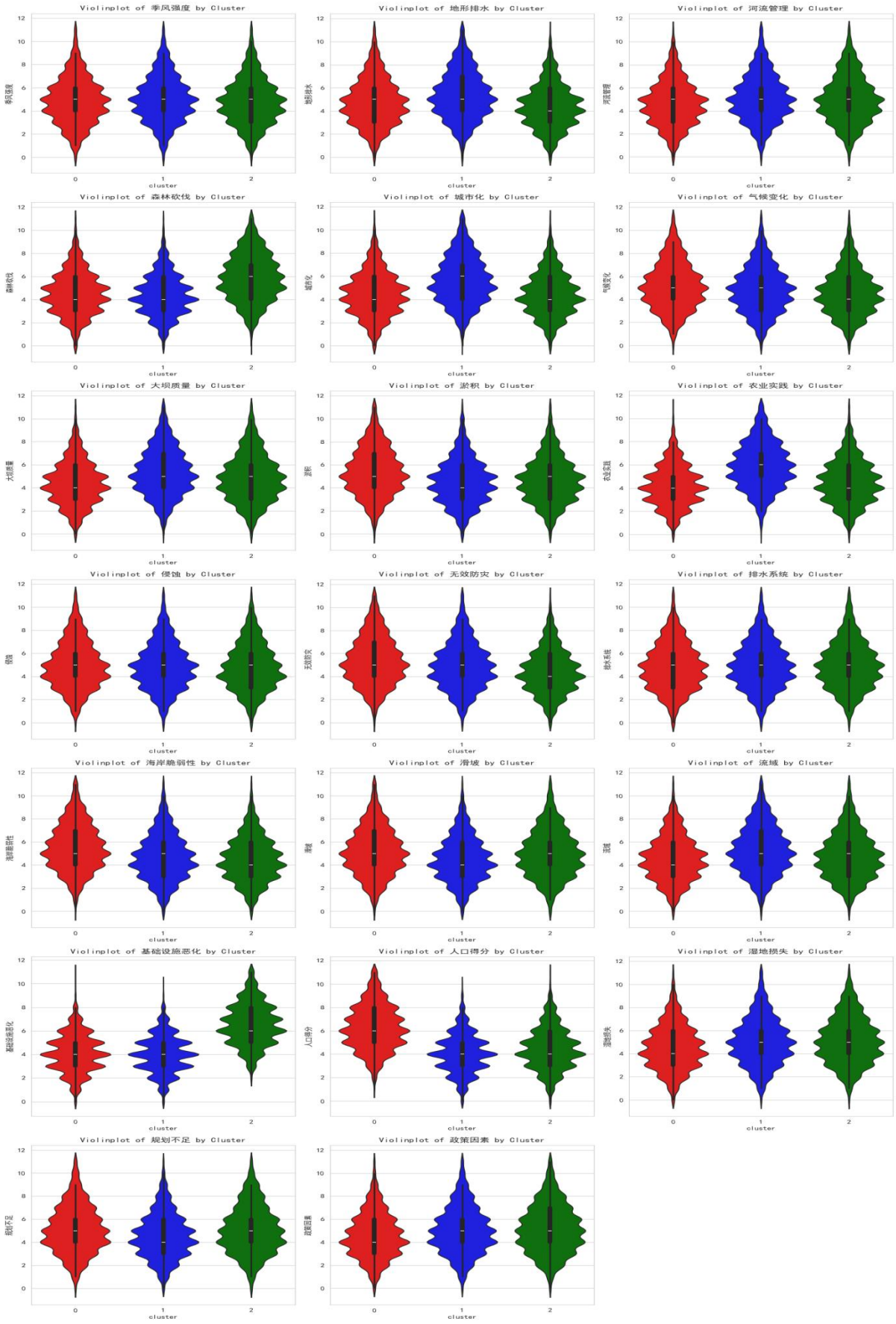

图 7 不同指标不同类别小提琴图图

从图中我们可以看出，在森林砍伐，基础设施恶化，人口得分这三个指标上，三个类别之间的差异性较大，分别是类别 2，类别 2，类别 1 较为突出。

6.3 预警评价模型建立

我们采用随机森林模型通过特征重要性来评估合适的指标及其权重，随机森林 是一种集成学习方法，用于解决分类和回归问题。它由多个决策树组成，每个决策树都 是独立训练的，并通过投票或平均值来进行最终的预测。

其表达式如下：

对于分类问题，随机森林通过多个决策树的投票来确定最终的预测结果。假设有N个决策树，每个决策树的预测结果为y_i，则随机森林的预测结果为：

$$\text{Prediction}=MajorityVote(y_1,y_2,\ldots,y_N)$$

其中，*MajorityVote* 表示投票过程，选择出现次数最多的类别作为最终的预测结果。

由此我们将 *K-means++*聚类的结果作为目标列，其余除洪水概率外作为特征列构建随机森林分类器，特征重要性结果如下：

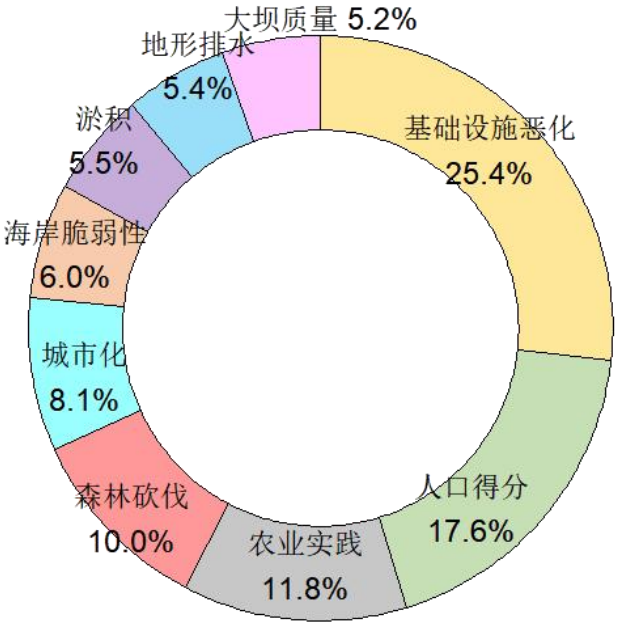

图 8 随机森林分类特征重要性结果图

根据特征重要性分析，我们可以得出以下结论：基础设施恶化是影响预测结果的最关键因素，表明其在模型中占据主导地位。紧随其后的是人口得分，其对结果也具有显著影响。农业实践和森林砍伐的重要性值位列第三和第四，突显了土地使用和森林覆盖变化对结果的较大影响。城市化和海岸脆弱性的重要性占比分别为 8.1%和 6.0%，表明城市发展和沿海地区的脆弱性同样不容忽视。地形排水和大坝质量的重要性占比接近，说明地形条件和水利设施的维护对结果有相似程度的影响。整体来看，特征重要性分布呈现出明显的梯度，其中基础设施恶化和人口得分的重要性明显高于其他因素。

接着我们选择了特征重要性最高的五个指标重新代入到随机森林分类器中训练，结果如下：

表 4 随机森林初步分类结果

	precision	*recall*	*f1-score*
Class_0	0.66	0.66	0.66
Class_1	0.75	0.75	0.75
Class_2	0.77	0.77	0.77
accuracy			0.73
macro avg	0.73	0.73	0.73
weighted avg	0.73	0.73	0.73

为了寻求最优的分类性能，我们使用了网格扩展来进行参数调优：

网格搜索（*Grid Search*）是一种用于超参数调优的方法，常用于机器学习模型的参数选择过程中。在网格搜索中，我们指定了一个参数候选集合，然后通过穷举组合这些参数的方式来进行模型训练和评估，最终选择在给定数据集上表现最好的参数组合。网格搜索的过程可以用以下步骤来描述：

Step1）定义参数候选集合：根据模型的需求和先验知识，选择一组待调优的参数，并为每个参数定义一个候选值集合。

Step2）穷举组合：将参数候选集合中的所有参数进行排列组合，生成一个参数组合的网格。

Step3）训练和评估：对于每个参数组合，使用训练数据训练模型，并使用验证数据或交叉验证方法进行评估，得到一个模型性能指标（如准确率、*F1-score* 等）作为评估结果。

Step4）选择最佳参数组合：根据评估结果，选择具有最佳性能的参数组合作为最终的模型参数。

最优参数结果如下：

表 5 网格扩展优化结果

max_depth	*max_features*	*min_samples_leaf*	*min_samples_split*	*n_estimators*
10	*sqrt*	2	5	100

表 6 最优分类结果

	precision	*recall*	*f1-score*
Class_0	0.71	0.69	0.70
Class_1	0.78	0.79	0.79
Class_2	0.79	0.80	0.80
accuracy			0.76
macro avg	0.76	0.76	0.76
weighted avg	0.76	0.76	0.76

综合评价：从这些指标来看，模型整体表现良好，准确率达到 76%，且在各个类别上的性能相对均衡。然而，*Class_0* 的精确度和召回率略低于 *Class_1* 和 *Class_2*，这表明模型在 *Class_0* 上的性能有待提高。

6.4 敏感性分析

我们接着提取出了模型的特征重要性：**基础设施恶化：0.3341，人口得分：0.2421，农业实践：0.1653，森林砍伐：0.1412，城市化：0.1173**

综合分析：

基础设施恶化是最重要的特征，具有最高的特征重要性值 0.3341。这意味着在模型的决策过程中，基础设施的恶化程度对预测结果有最大的影响。

人口得分排在第二位，重要性值为 0.2421。人口得分可能反映了人口密度、人口结构或其他人口统计特征，这些特征对模型的预测也有一定程度的影响。

农业实践的特征重要性值为 0.1653，位列第三。这可能包括土地使用方式、农药使用或作物种植模式等因素，这些都可能影响模型的预测。

森林砍伐的特征重要性值为 0.1412，位列第四。森林砍伐可能与土地覆盖变化、生物多样性损失或碳排放有关，这些因素在模型预测中也起到一定作用。

城市化的特征重要性最低，为 0.1173。尽管它的重要性相对较低，但仍然对模型的预测结果有一定的贡献。

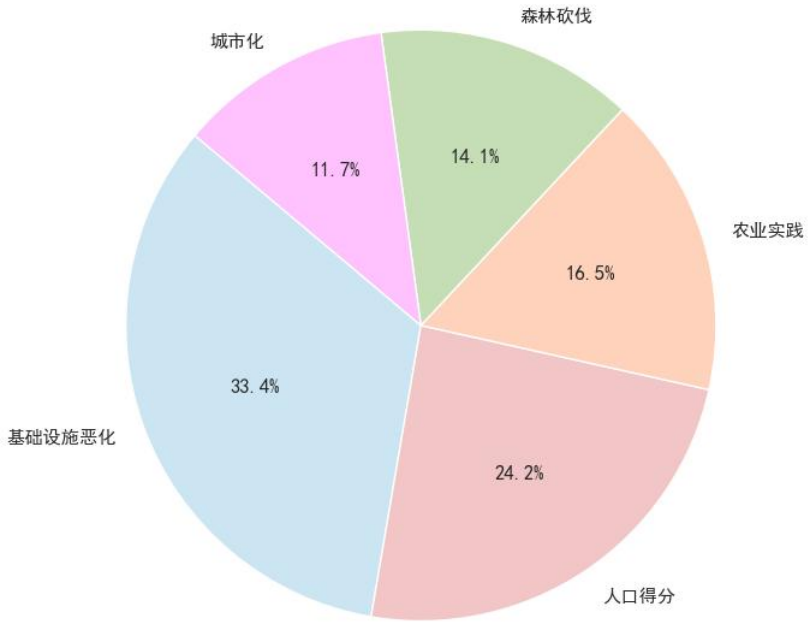

图 9 最优模型重要性可视化图

七、 问题三模型的建立与求解

7.1 数据预处理

我们选择直接使用问题一处理过后的数据。

7.2 特征工程

由于第一问的指标评价结果不足以选择出合适的指标，所以我们又引入了互信息的方法来进行特征选择：

互信息 (*Mutual Information*) 是一种度量两个随机变量之间相互依赖程度的统计量。

它可以用于衡量一个变量中的 信息对于另一个变量的预测能力，或者说用于度量两个变量之间的共享信息量。其表达式如下：

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p\,(x,y) log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

其中：$I(X;Y)$表示变量X和Y之间的互信息，$p(x,y)$是联合概率分布函数，表示变量X和Y同时取值x和y的概率；$p(x)$和$p(y)$分别是变量X和Y的边缘概率分布函数，表示变量X和Y分别取值x和y的概率。

互信息的值越大，表示两个变量之间的相互依赖程度越高；值越小或接近于零，表示两个变量之间的相互依赖程度越低或不存在相互依赖关系，结果如下：

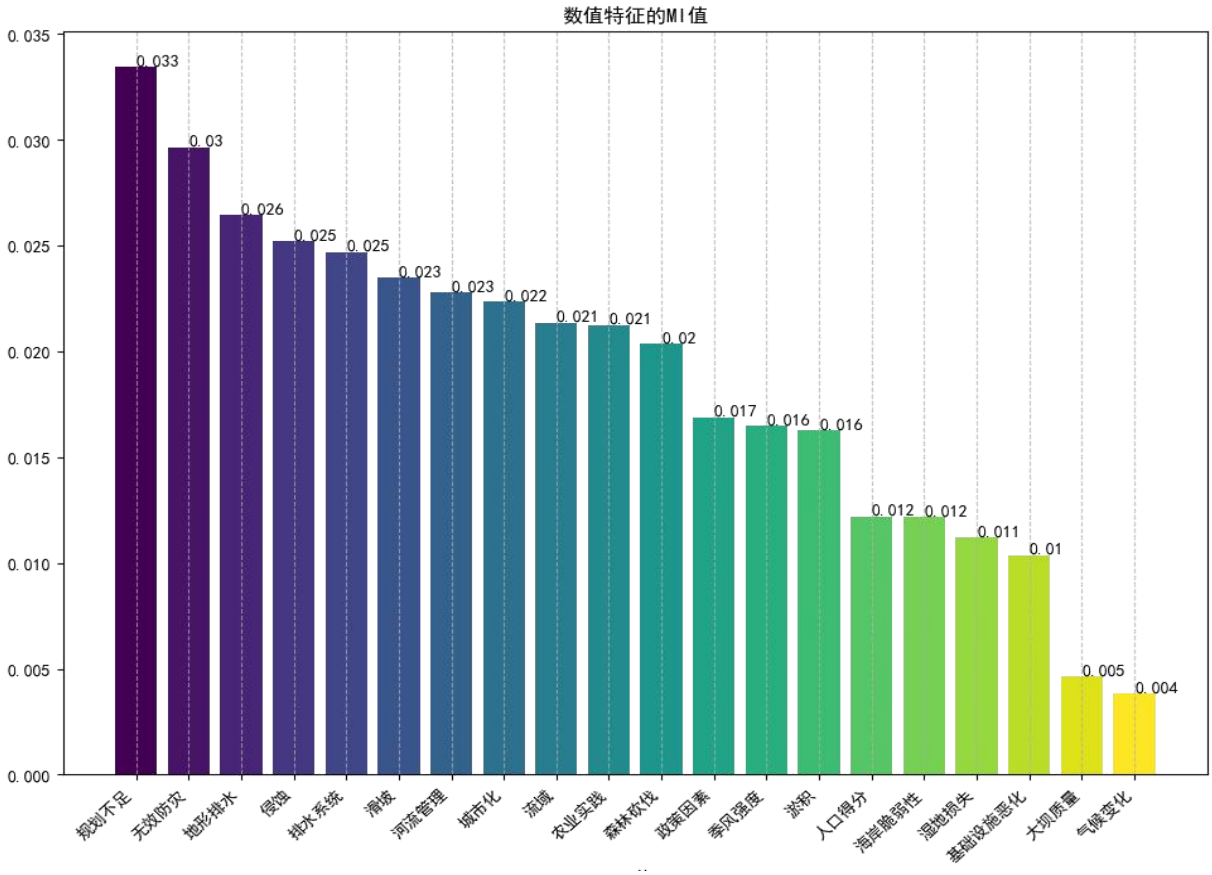

图 10 互信息结果图

基于互信息的方法，我们可以发现特征列之间有着一定的区分度了，于是我们综合了问题一以及互信息的结果，最终选择出了与洪水概率相关最高的十个指标：

基础设施恶化、规划不足、无效防灾、季风强度、地形排水、侵蚀、河流管理、淤积、人口得分，滑坡。

7.3 概率预测模型的建立

7.3.1 *Baseline*

首先我们采用 *XGBoost*、*Catboost*、*LightGBM* 作为基线模型：

XGBoost 是一种梯度提升树算法的优化实现，它在机器学习和数据科学领域广泛应

用。*XGBoost* 通过优化目标函数并利用梯度提升技术，能够有效地处理各种类型的回归和分类问题。其表达式如下：
对于回归问题：

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

其中，$\hat{y}_i$是对样本x_i的预测值，K是(决策树)的数量，f_k是第k个决策树的预测函数。*XGBoost* 的目标函数由两部分组成：损失函数和正则化项。损失函数衡量预测值与真实值之间的差异，而正则化项用于控制模型的复杂度。
目标函数的数学表达式如下：

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

其中，l是损失函数，衡量预测值与真实值之间的差异；Ω是正则化项，用于控制模型的复杂度；θ是模型参数。

Catboost 是一种梯度提升树算法，*Catboost* 在处理分类和回归问题时表现出色，并且具有处理高维类别特征的能力。*Catboost* 的数学表达式与传统的梯度提升树算法类似，它通过迭代地训练一系列决策树模型，并将它们组合成一个强 大的集成模型。其数学表达式可以表示为：

$$\hat{f}(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$$

其中：γ_m是第m轮迭代的学习率(步长)，$h_m(x)$是第m轮迭代的决策树模型的预测结果。

LightGBM 是一种高效梯度提升决策树框架。它利用了诸如叶节点优化、分层学习率以及基于 *Histogram* 的渐进学习等技术，以实现更快的训练速度和更高的模型性能。*LightGBM* 遵循梯度提升树的基本原理，即通过迭代地构建决策树来提升模型，每次迭代构建一棵树来减少当前模型的残差（即预测与实际值之间的差值）。在每次迭代中，树的叶节点用于输出预测值，这些叶节点的值会根据梯度下降来调整，以最小化损失函数。

我们使用所选的十个特征列来对洪灾概率进行拟合，基线模型结果如下

表 7 基线模型交叉验证结果表

模型名称	*Fold 1 - MAPE*	*Fold 2 - MAPE*	*Fold 3 - MAPE*	*Fold 4 - MAPE*	*Fold 5 - MAPE*	*MAPE* ± 标准差
XGBoost	0.070519	0.070462	0.071263	0.071162	0.068331	0.070347 ± 0.001059
Catboost	0.066070	0.067376	0.067968	0.067544	0.065113	0.066814 ± 0.001061
LightGBM	0.078245	0.078958	0.078833	0.080079	0.077577	0.078738 ± 0.000830

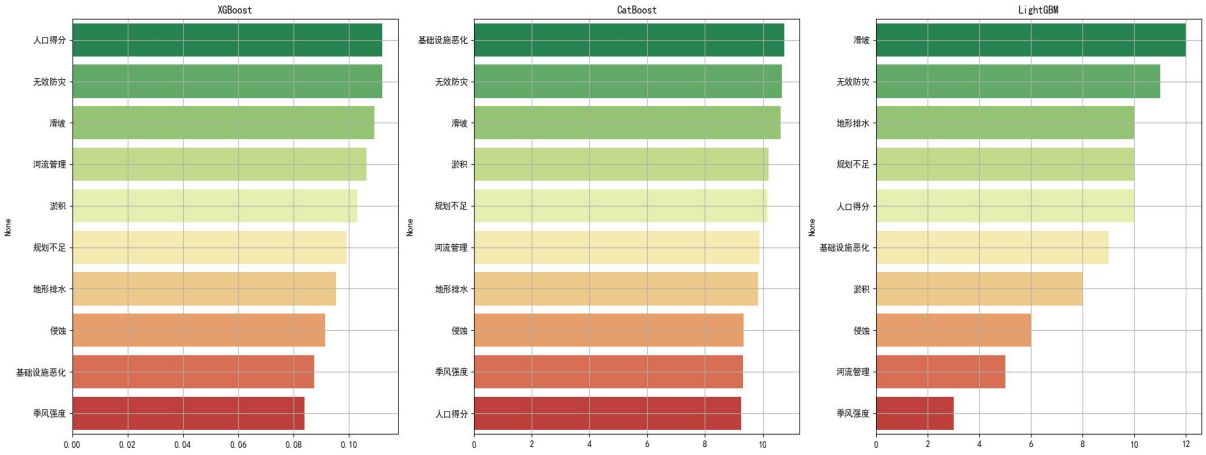

图 11 基线模型特征重要性图

综合评价：

Catboost 在所有模型中表现出了最低的平均 *MAPE*，这意味着它在预测精度上表现最佳。*XGBoost* 的平均 *MAPE* 略高于 *Catboost*，但仍然表现良好，且稳定性较高。*LightGBM* 的平均 *MAPE* 最高，且在不同折中的波动较大，这可能表明模型对数据的敏感度较高或存在过拟合的风险。

7. 3. 2 *Weighted Ensemble*

Weighted Ensemble 是一种集成学习方法，它融合了多个机器学习模型（或基模型）的预测结果，以提高预测的准确性和稳定性。这种方法的核心思想是在训练过程中赋予每个基模型不同的权重，使得预测结果的最终输出是这些模型预测结果的加权平均。其表达式为：

假设我们有M个基模型 $(M = 1,2,\ldots,m)$,每个模型$f_m(x)$都在给定的输入 x 上给出预测。*Weighted Ensemble* 的目标是通过计算每个模型的预测结果的加权平均来获得最终的预测结果。 设权重向量为：

$$w = [w_1, w_2, \ldots, w_M]^T$$

其中w_m是模型$f_m(x)$的权重。

权重通常需要满足

$$\sum_{m=1}^{M} w_m = 1$$

以确保权重之和为 1，从而形成一个有效的加权平均。

最终的预测结果$E(x)$可以表示为：

$$E(x) = \sum_{m=1}^{M} w_m \cdot f_m(x)$$

为了找到最优的参数组合，我们采用了 *optuna* 超参数优化的方法，*optuna* 使用了高斯过程（*Gaussian Process*）来建模超参数空间中的目标函数。目标函数可以是机器学习模型的性能指标，例如准确率、均方误差等。*optuna* 的目标是最小化或最大化目标函数的值，以找到最优的超参数配置。最终优化结果如下：

最优参数组合：[1.00745929e-04 9.42122588e-01 5.77766657e-02]

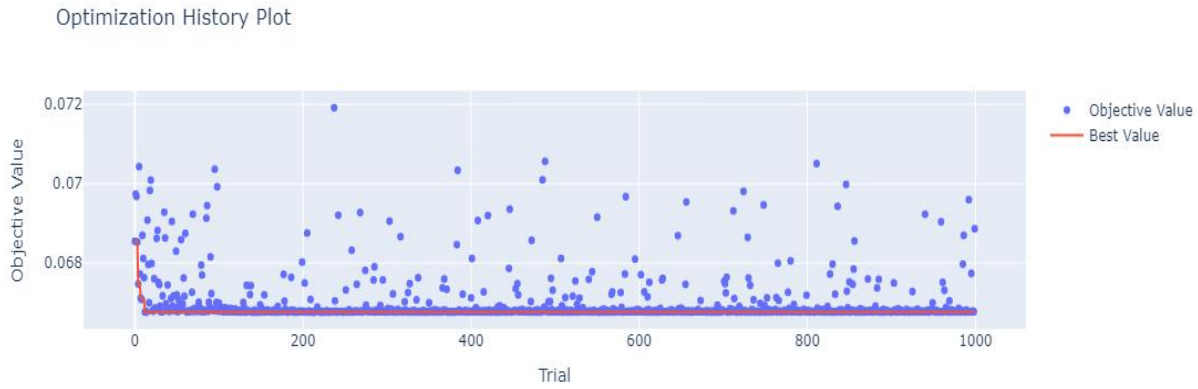

图 12 加权模型超参数调优结果图

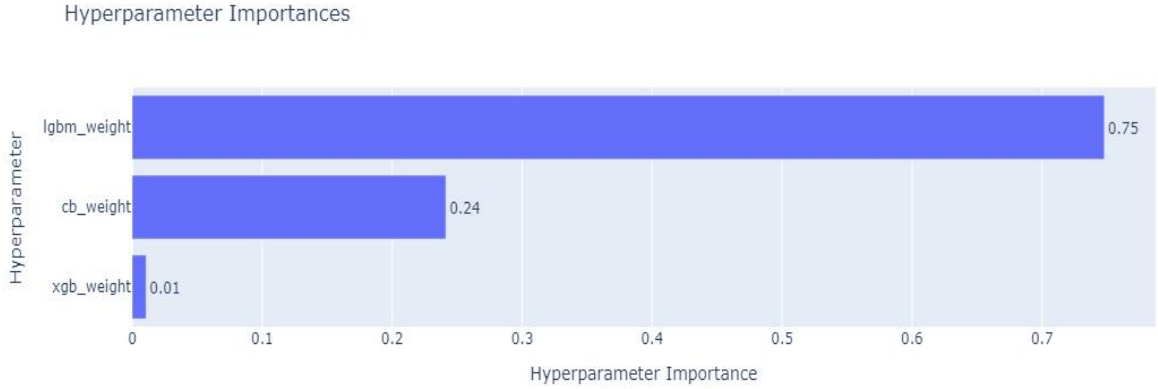

图 13 加权模型调优特征重要性图

该方法的最终 *MAPE* 为 0.068124。

7.3.3 *Voting*

Voting 是集成学习方法中的一种，主要用于分类任务。其核心思想是通过多个基础模型（或称为基模型）的预测结果进行整合，以提高预测的准确度和稳定性。*Voting* 可以分为两大类：硬投票（*Hard Voting*)和软投票（*Soft Voting*）。

数学表达式总结：

硬投票：

$$Y_i = argmax \sum_{m=1}^{M} 1\{predictions_m[i] = k\}$$

软投票：

$$Y_i = argmax_k \sum_{m=1}^{M} \omega_m . probabilities_m[i][k]$$

通过 *Voting* 方法，可以有效地结合多个模型的优势，提高预测的准确性和鲁棒性，该方法的最终 *MAPE*：0.068488。然后我们又使用了优化后的最优权重进行了 *weighted voting*，*MAPE* 为 0.066766。

7.3.4 *Stacking*

Stacking（叠加）是一种集成学习方法，用于将多个基础模型的预测结果结合起来，以提高整体预测性能。在 *Stacking* 中，我们不仅仅使用基础模型的预测结果作为输入特

征，还将它们的预测结果作为训练集来训练一个元模型（也称为次级学习器）。
对于训练集：
假设有K个基础模型和一个元模型，第 k 个基础模型的预测结果为：

$$f_k(X)$$

其中X是训练集的输入特征。
元模型的输入特征为：

$$F(X) = [f_1(X), f_2(X), \ldots, f_K(X)]$$

其中$F(X)$是一个$N \times K$的矩阵，N是训练集样本数量
元模型的输出为

$$Y$$

即目标变量的预测结果。
元模型可以表示为

$$Y = g\big(F(X)\big)$$

其中 g(·)是元模型的函数。
对于测试集：
假设测试集的输入特征为

$$X_{test}$$

用基础模型对测试集进行预测，得到预测结果矩阵

$$F_{test} = [f_1(X_{test}), f_2(X_{test}), \ldots, f_K(X_{test})]$$

将F_{test}作为输入特征，通过元模型得到最终的预测结果$Y_{test} = g(F_{test})$。
通过 *Stacking*，我们可以利用不同基础模型之间的互补性，以及元模型的整体学习能力，提高预测的准确性和泛化能力，其最终 *MAPE* 为 0.066883。

7.3.5 综合比较分析

最后我们可视化了这几个模型的 *mean MAPE*，以及 *FOLD MAPE* 结果：

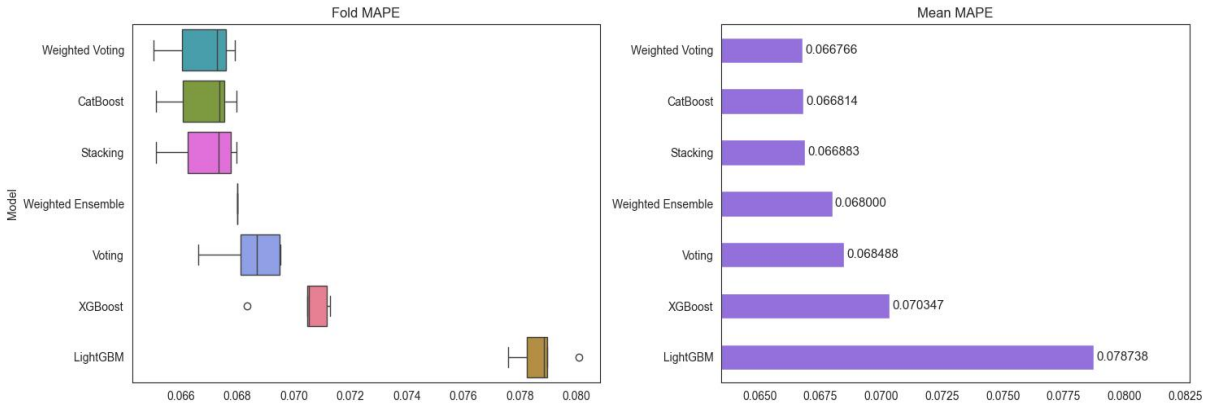

图 14 模型的 *MAPE* 结果图

从图中我们可以分析出 *catboost* 是最优模型，于是我们最终选择 *catboost* 来进行概率预测。

7.4 *catboost*

通过上面的分析我们已经选择出了最优化模型为 *catboost*，接下来我们对其进行超参数调优来获得模型的最佳性能：

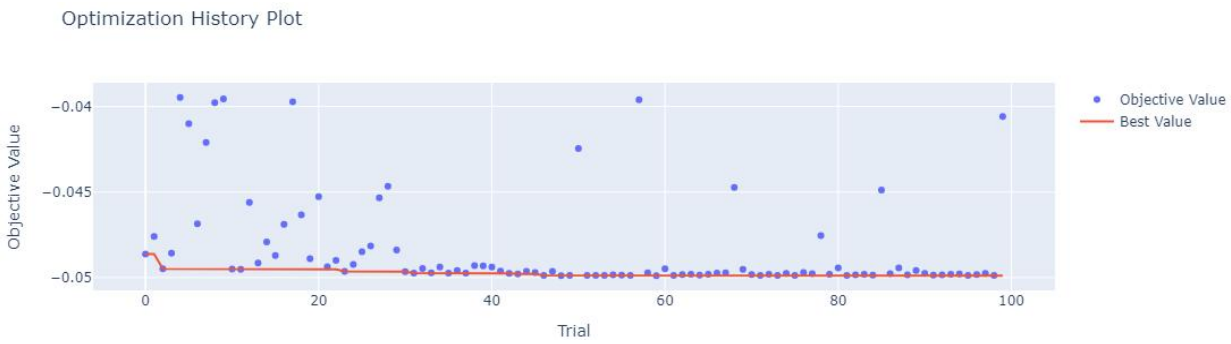

图 15 *catboost* 超参数调优结果图

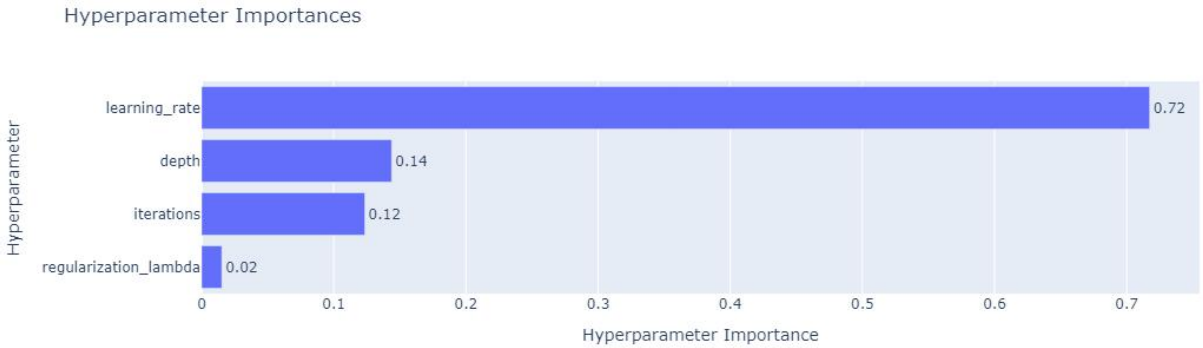

图 16 *catboost* 调优特征重要性图

最优参数为：

iterations: 120

learning_rate: 0.00011405800769687978

depth: 3

regularization_lambda: 0.09553182571585939

于是我们将最优参数代回模型中进行训练，最终结果为：

***Final MSE*: 0.0024902002943744737, *Final RMSE*: 0.04990190672083056**

7.5 *shap* 可解释性分析

SHAP (*SHapley Additive exPlanations*) 是一种基于 *Shapley* 值的解释方法，用于深度学习和其他复杂模型的特征重要性分析及预测结果的解释。*SHAP* 通过将预测结果分解为每个特征的贡献，帮助理解模型的决策过程，提供了一种全局和局部的可解释性分析。其表达式如下：

考虑一个目标函数 F,为 n 个特征$x_1, x_2, ..., x_n$计算预测结果。假设 F 是由 m 个基础模型$f_1, f_2, ..., f_m$的加权和构成：

$$F(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$$

其中$x = (x_1, x_2, ..., x_n)$是输入特征向量，α_i表示模型f_i对总预测的加权贡献。

SHAP 值定义为对于每个特征 j 的平均期望值贡献：

$$\phi_j(F(x)) = \frac{1}{n!} \sum_{S \subset \{1,2,\ldots,n\} \setminus \{j\}} \Delta_{S \cup \{j\}}$$

SHAP 提供了一种系统化的方法来解释复杂模型的决策过程，通过 *SHAP* 值计算每个特征对预测结果的贡献，从而增强了模型的可解释性。

我们绘制出了 *shap* 摘要图、特征间交互作用图以及热力图来进行综合分析：

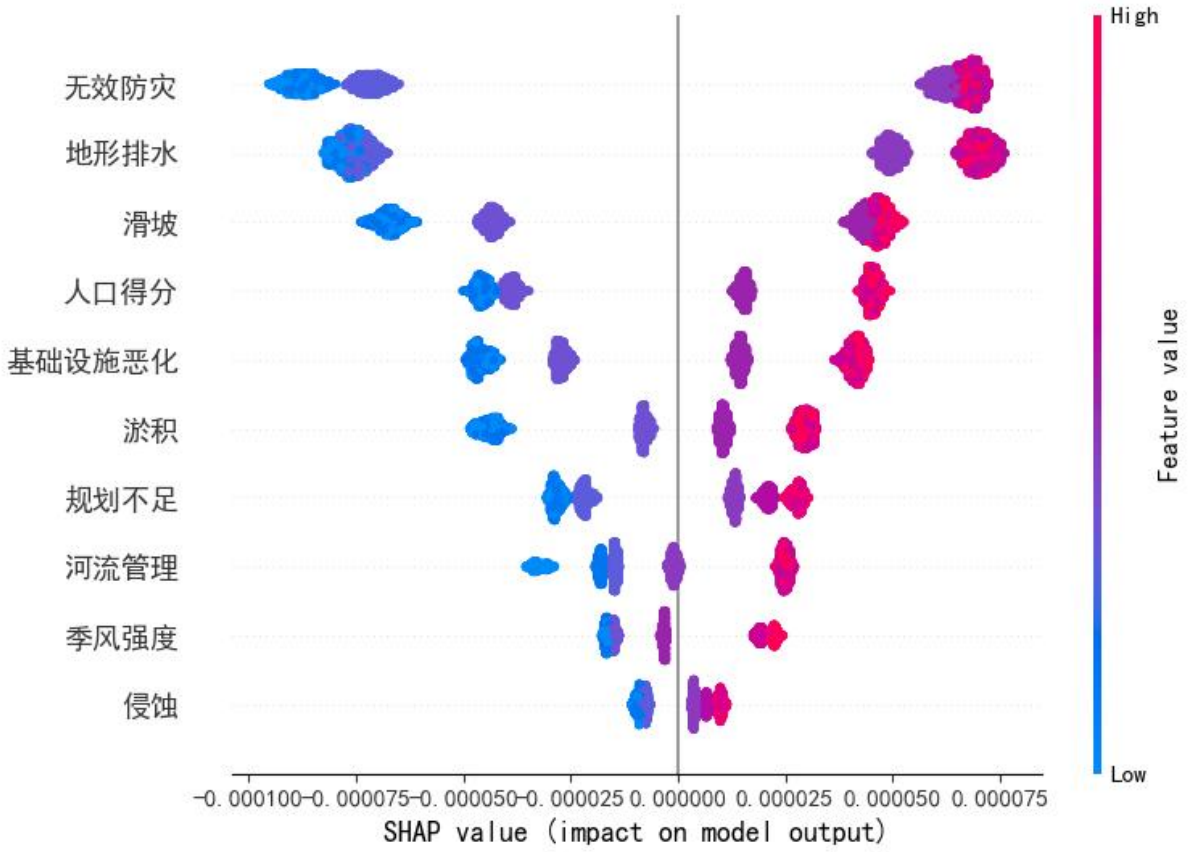

图 17 *shap* 摘要图

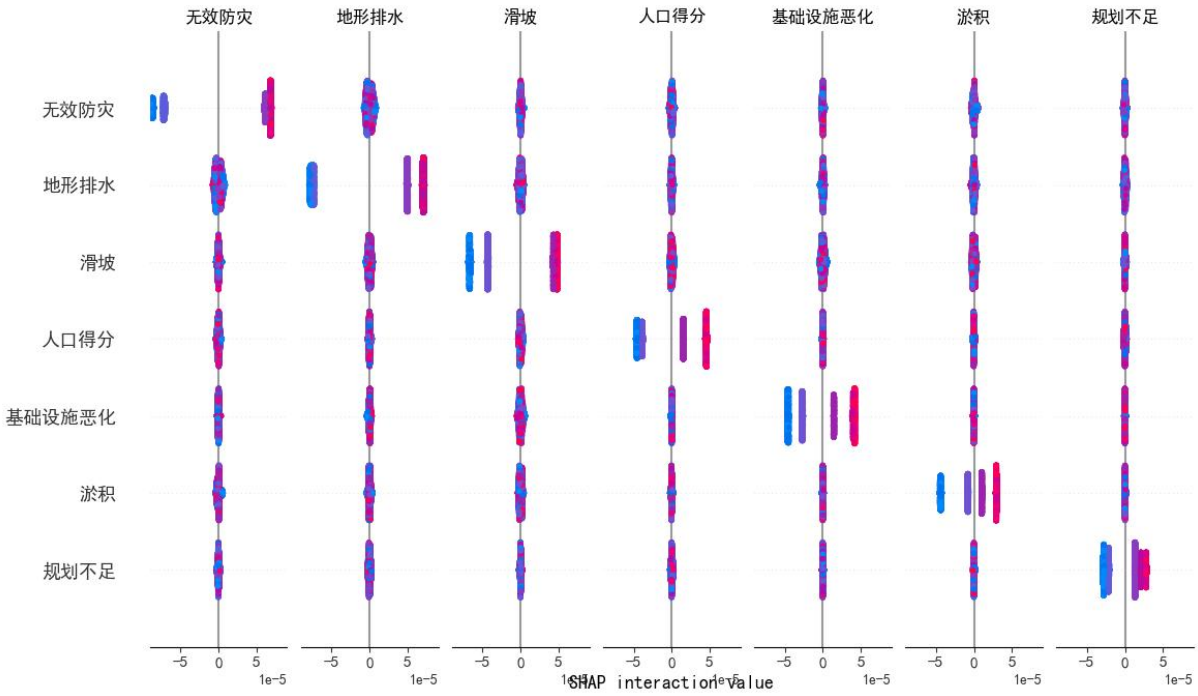

图 18 特征交互作用图

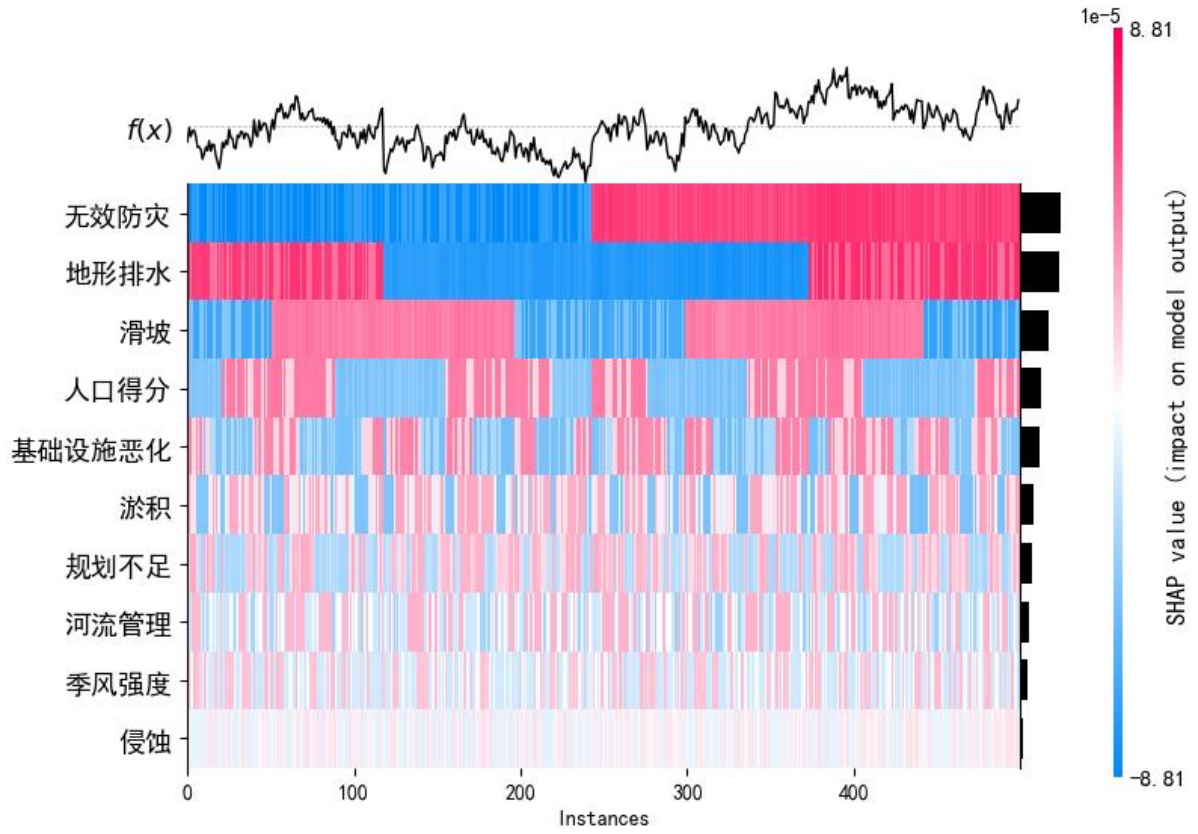

图 19 *shap* 热力图

摘要图分析：

在图中，蓝色表示高特征值（*High*），紫色表示低特征值（*Low*）。每个点代表一个特征值，其位置反映了该特征值对模型输出的贡献程度。

从图中可以看出，某些特征值对模型输出具有显著的影响，而另一些则影响不大。例如，“无效防灾”和“地形排水”这两个特征值对模型输出的影响相对较大，因为它们的点位于较高的区域。相反，“滑坡”、“人口得分”、“基础设施恶化”等特征值对模型输出的影响较小，因为它们的点位于较低的区域。

交互作用图分析：

该图展示了不同因素（无效防灾、地形排水、滑坡、人口得分、基础设施恶化、淤积和规划不足）对某一变量的影响。通过分析这些因素的交互作用，可以更好地理解它们在特定情境下的综合影响。

1. 无效防灾与地形排水：这两个因素之间存在显著的负向交互作用。这意味着，当这两个因素同时存在时，它们对目标变量的负面影响会增强。这可能是由于地形排水不良导致无效防灾措施难以发挥作用，或者两者共同导致了某种负面结果。

2. 滑坡与人口得分：这两个因素之间存在正向交互作用。这表明，当滑坡发生时，人口得分可能会受到影响并增加。这可能是因为滑坡事件会导致人员伤亡或财产损失，从而影响到人口得分。

3. 基础设施恶化与淤积：这两个因素之间存在负向交互作用。这意味着，当基础设施恶化严重时，淤积问题可能更加突出，反之亦然。这可能是由于基础设施恶化导致的水流不畅，进而加剧了淤积问题。

4. 规划不足与地形排水：这两个因素之间存在负向交互作用。这表明，当规划不足时，地形排水问题可能更加严重。这可能是由于缺乏有效的规划导致地形排水系统无法正常工作。

5. 淤积与人口得分：这两个因素之间存在负向交互作用。这意味着，当淤积问题严重时，人口得分可能会受到影响并降低。这可能是因为淤积导致水质下降或生态环境恶化，进而影响到人口的健康和生活质量。

热力图分析：

这张热力图展示了不同因素对某一变量 *f(x)*的影响程度。通过颜色编码，我们可以直观地看到每个因素的正负影响以及其相对重要性。

1. 无效防灾：蓝色表示负向影响，红色表示正向影响。从图中可以看出，在大部分情况下，无效防灾对变量 *f(x* 有正向影响，尤其是在 100 到 200 之间。然而，在某些特定点上，如 300 和 400 处，无效防灾对变量 *f(x)*有负向影响。

2. 地形排水：同样采用蓝色和红色来表示正负影响。地形排水在大多数情况下对变量 *f(x)*有正向影响，尤其在 100 到 200 之间。但在 300 和 400 处，地形排水对变量 *f(x)* 有负向影响。

3. 滑坡：使用粉色表示正负影响。滑坡在大部分情况下对变量 *f(x)*有负向影响，尤其是在 350 到 450 之间。

4. 人口得分：采用浅蓝色表示正负影响。人口得分在大多数情况下对变量 *f(x)*没有显著影响或影响较小。

5. 基础设施恶化：使用深蓝色表示正负影响。基础设施恶化在大部分情况下对变量 *f(x)*有负向影响，尤其是在 100 到 250 之间。

6. 淤积：采用紫色表示正负影响。淤积在大部分情况下对变量 *f(x)*有负向影响，尤其是在 350 到 450 之间。

7. 规划不足：使用绿色表示正负影响。规划不足在大部分情况下对变量 *f(x)*有正向影响，尤其是在 150 到 250 之间。

8. 河流管理：采用橙色表示正负影响。河流管理在大部分情况下对变量 *f(x)*有正向影响，尤其是在 150 到 250 之间。

9. 季风强度：使用黄色表示正负影响。季风强度在大部分情况下对变量 *f(x)*有正向影响，尤其是在 150 到 250 之间。

10. 侵蚀：采用棕色表示正负影响。侵蚀在大部分情况下对变量 *f(x)*有正向影响，尤其是在 150 到 250 之间。

7.6 最小化模型

通过 *shap* 热力图我们可以筛选出对模型输出影响最大的五个特征分别为：无效防灾，地形排水，滑坡，人口得分，基础设施恶化，于是我们采用这五个特征列再次对模型进行训练，最终结果为：

***Final MSE*: 0.00245802829083526, *Final RMSE*: 0.051880139294949104**

从误差角度来看，我们的 *catboost* 模型在选用对模型输出影响程度最大的五个指标的情况下可以很好的还原原模型的性能，如果需要在五个指标的情况下完美还原原模型的性能，可能需要更详细的分析一些特征列之间的交互关系，因为即使某些特征单独看起来不重要，它们与其他特征的组合可能对预测有显著影响。

八、 问题四模型的建立与求解

我们采用第三问的最优模型来进行预测，接着我们对预测结果进行正态性检验：

①图像方面：

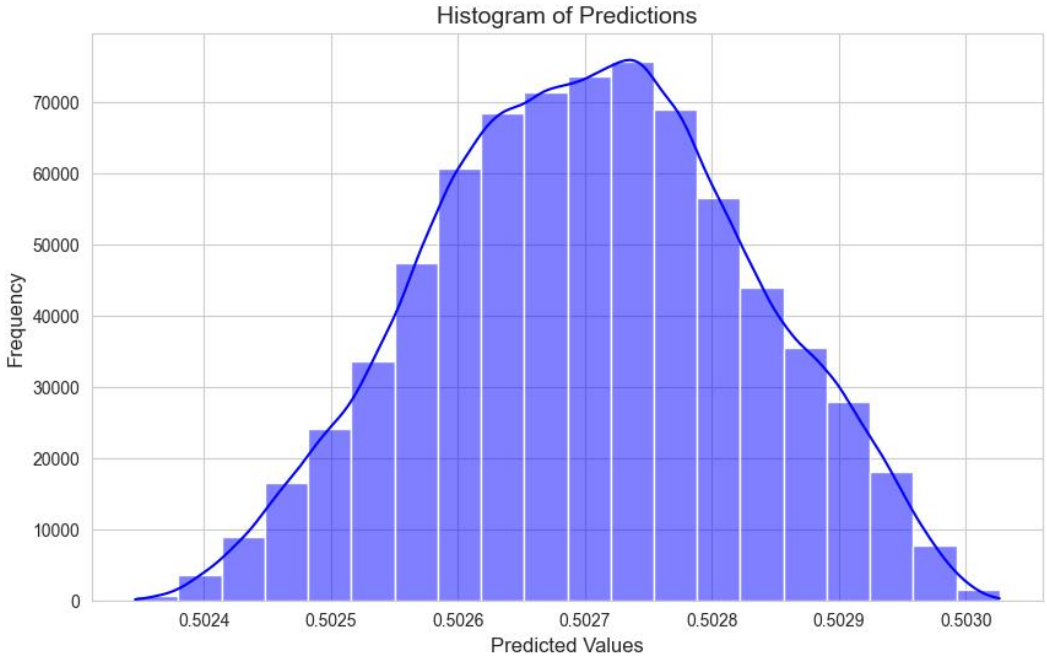

图 20 预测结果直方图

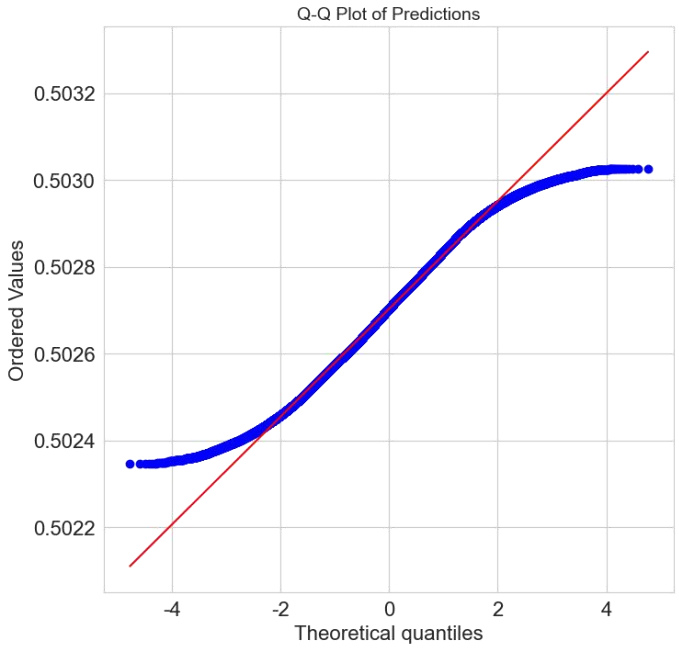

图 21 预测结果 Q-Q 图

从结果图中我们可以看出结果的分布是近似服从正态分布的。

②定量分析：

Kolmogorov-Smirnov 是一种用于判断数据是否符合某种特定分布（如正态分布）的统计检验方法。它通过比较样本分布函数与假设分布函数之间的最大偏离程度来判断数据是否符合给定的分布。在判断正态性时，通常假设的分布就是正态分布。

我们使用该检验方法对结果进行检验：*KS* 检验统计量: 0.018，*KS* 检验 p 值: 3.42e-229 在 0.05 的显著性水平下，拒绝正态性的零假设。

最中我们将二者的检验结果综合考虑得出结论，预测结果是近似正态分布的。

九、 模型的评价与推广

9.1 预测模型的优点

多模型基线: 使用 *xgboost*、*catboost* 和 *LightGBM* 作为基线模型是一个很好的开始，因为这些基于树的模型在许多回归问题上表现良好。

集成学习: 考虑多种集成学习方法来提高模型性能，这通常可以提高模型的泛化能力和准确性。

性能评估: 使用 *MAPE*（平均绝对百分比误差）作为评价指标是合适的，因为它能够衡量预测值与实际值之间的相对误差，是评估回归模型性能的常用指标。

超参数优化: 使用 *optuna* 进行超参数优化，因其是一个强大的库，可以帮助找到最佳的超参数组合。

模型解释性: 通过 *SHAP*（*SHapley Additive exPlanations*）进行模型可解释性分析，有助于理解模型预测背后的逻辑，这对于模型的信赖度和进一步的改进至关重要。

9.2 预测模型的缺点

特征选择方法: 还可以更全面的进行特征选择，例如询问相关方面的专家等方法以确保选取的特征是最优的。

模型评估: 仅使用 MAPE 可能不足以全面评估模型性能。考虑使用其他指标，如 MAE（平均绝对误差）、RMSE（均方根误差）等，以获得更全面的评估。

模型泛化能力: 没有训练模型在不同数据集上的泛化能力。

9.2 预测模型的推广

跨学科应用：

模型的方法论和技术可以应用于其他类型的自然灾害风险评估，如地震、干旱等。

风险评估与管理：

模型可以应用于不同地区的风险评估，帮助政府和相关部门制定防洪措施和应急计划。

城市规划与建设：

模型结果可以指导城市规划者在易受洪水影响的地区进行更合理的土地利用规划和基础设施建设。

十、参考文献

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). DOI: 10.1145/2939672.2939785

[2]Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. In Advances in Neural Information Processing Systems (pp. 6638-6648).

[3]Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (pp. 3146-3154).

附录

附录 1：支撑材料文件列表
附录 2：代码

附录 1：支撑材料文件列表

文件列表名	
问题一	数据处理以及热力图代码
问题二	Kmeans 聚类以及随机森林分类代码 描述性统计以及 Lasso 特征选择代码
问题三	最优模型选取代码 Catboost+shap 分析代码
问题四	预测结果 submit.csv 正态性检验代码

附录2：代码

流程图 *mermaid* 代码

```
graph LR    A -->|问题一分析| D D -->|斯皮尔曼相关系数分析| D1 D -->|机器学习方法拟合| D2 D -->|特征重要性分析| D3 D4 A -->|问题二分析| E E -->|确定聚类数目为三类| E1 E -->|使用 kmeans++聚类方法| E2 E -->|聚类中心确定| E3 E -->|随机森林分类器构建| E4 E4 -->|特征重要性筛选| E5 E -->|超参数调优| E6 E -->|分类预测与结果可视化| E7 E -->|敏感度分析| E8 A -->|问题三分析| F F -->|综合问题一评价结果| F1 F -->|指标选取方法比较| F2 F -->|基线模型选择| F3 F3 -->|xgboost| F31 F3 -->|catboost| F32 F3 -->|LightGBM| F33 F -->|集成学习方法比较| F4 F -->|交叉验证 MAPE 评估| F5 F -->|Optuna 超参数优化| F6 F -->|SHAP 可解释性分析| F7 F -->|重要指标选取与模型最小化| F8 A -->|问题四分析| G G -->|使用最优模型进行预测| G1 G -->|KS 检验| G2 G -->|结果正态性判断| G3
```

问题一数据处理以及热力图代码

```
file_path <- "C:/Users/30766/Desktop/train.xlsx"

# 读取 xlsx 文件
data <- read_excel(file_path)
str(data)

library(ggplot2)
library(RColorBrewer)

# 计算每个变量的缺失值数量
na_counts <- sapply(data, function(x) sum(is.na(x)))
```

```
# 将缺失值计数转换为数据框
na_df <- data.frame(
  variable = names(na_counts),
  missing_count = na_counts
)

# 确保变量列是因子类型
na_df$variable <- factor(na_df$variable, levels = names(na_counts))

# 选择颜色方案，例如使用 "Paired" 方案
color_scheme <- brewer.pal(22, "Paired")  # 选择 11 种颜色
custom_colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b",
"#e377c2", "#7f7f7f", "#bcbd22", "#17becf", "#aec7e8", "#ffbb78", "#98df8a", "#ff9896",
"#c5b0d5", "#c49c94", "#f7b6d2", "#c7c7c7", "#dbdb8d", "#9edae5", "#adadad",
"#e7e7e7")

# 绘制条形图，并使用 RColorBrewer 提供的颜色方案
ggplot(na_df, aes(x = variable, y = missing_count, fill = variable)) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = custom_colors) +  # 使用 RColorBrewer 提供的颜色
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # 旋转 x 轴标签
    plot.title = element_text(face = "bold", size = 14, hjust = 0.5)  # 调整标题样式和位
置
  ) +
  labs(x = "Variable", y = "Missing Count", title = "Missing Values by Variable")



file_path <- "C:/Users/30766/Desktop/1.xlsx"

# 读取 xlsx 文件
data <- read_excel(file_path)
str(data)
library(tidyverse)
data %>%
  as_tibble() -> df

# 计算相关系数矩阵
corr_matrix <- Hmisc::rcorr(as.matrix(df), type = "pearson")

# 提取相关系数矩阵
correlation_matrix <- corr_matrix$r
```

```
p_matrix <- corr_matrix$P

# 确定显著性阈值
significant_p <- 0.05

# 创建显著性标记矩阵
significant_matrix <- apply(p_matrix, c(1, 2), function(row) {
  ifelse(row < significant_p, "*", "")
})
colormap <- colorRampPalette(rev(brewer.pal(n = 7, name = "Pastel1")))(100)
pheatmap(correlation_matrix,
         clustering_distance_rows = "euclidean", clustering_distance_cols = "euclidean",
         clustering_method = "complete",
         cluster_rows = T,cluster_cols = T,
         treeheight_row = 30, treeheight_col = 30,
         border_color = "grey60",
         display_numbers = T,
         fontsize_number = 10,
         number_format = "%.2f",
         number_color = "grey30",
         show_rownames = T, show_colnames = T,
         main = "相关性热力图",
         color = my_color_scheme,
         legend_breaks = NA,
         legend_labels = NA,
         angle_col = "45")
```

描述性统计以及 Lasso 特征选择

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc, accuracy_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import matplotlib
import warnings
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
# 忽略警告信息
warnings.filterwarnings("ignore")
# 读取 CSV 文件
```

```
df = pd.read_excel("C:/Users/30766/Desktop/train.xlsx")  # 训练集
import pandas as pd
import numpy as np

# 计算峰度和偏度
def calculate_moments(df):
    moments = {}
    for column in df.columns:
        moments[column] = {
            'mean': df[column].mean(),
            'std': df[column].std(),
            'kurtosis': df[column].kurtosis(),
            'skewness': df[column].skew(),
            'cv': df[column].std() / df[column].mean() if df[column].mean() != 0 else
None  # 避免除以 0
        }
    return moments

# 计算所有数值型列的峰度、偏度和变异系数
moments = calculate_moments(df.select_dtypes(include=[np.number]))

# 打印结果
for column, values in moments.items():
    print(f"Column: {column}")
    print(f"  Mean: {values['mean']}")
    print(f"  Std Dev: {values['std']}")
    print(f"  Kurtosis: {values['kurtosis']}")
    print(f"  Skewness: {values['skewness']}")
    print(f"  Coefficient of Variation: {values['cv']}")
print("-" * 40)
import pandas as pd
import numpy as np


# 计算每个特征的均值和 3 倍标准差
mean = df.mean()
std = df.std()
thresholds = mean - 3 * std, mean + 3 * std

# 应用 3σ规则来识别异常值
for column in df.columns:
    df = df[(df[column] >= thresholds[0][column]) & (df[column] <=
thresholds[1][column])]
```

```
# 显示清洗后的数据
print(df.head())

# 计算洪水概率与各个指标之间的相关系数
correlation = df.corr()['洪水概率'].sort_values(ascending=False)

# 打印相关系数
print(correlation)

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler



# 将数据分为特征和目标变量
X = df.drop(['id', '洪水概率'], axis=1)  # 假设'id'和'洪水概率'是列名
y = df['洪水概率']

# 数据标准化
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LassoCV

# 使用 LassoCV 进行交叉验证
lasso_cv = LassoCV(alphas=np.logspace(-4, 1, 50), cv=5, max_iter=10000)
lasso_cv.fit(X_train, y_train)

# 打印最佳 alpha 值
print(f"Optimal alpha value: {lasso_cv.alpha_}")

# 使用 seaborn 和 matplotlib 绘制 MSE 与 alpha 的关系图
sns.set(style="whitegrid")  # 设置图表风格
plt.figure(figsize=(10, 6))  # 设置图表大小
```

```
sns.lineplot(x=lasso_cv.alphas_, y=-lasso_cv.mse_path_.mean(axis=1),
                    marker='o', markersize=6, linewidth=2.5, err_style="bars",
                    ci='sd')  # 绘制线图，并添加误差条表示标准差

plt.xscale('log')  # 设置 x 轴为对数尺度
plt.xlabel('Alpha')  # x 轴标签
plt.ylabel('Mean Squared Error')  # y 轴标签
plt.title('Lasso Regression MSE for Different Alpha Values')  # 图表标题
plt.show()

lasso = Lasso(alpha=0.0001)
# 训练模型
lasso.fit(X_train, y_train)

# 预测测试集
y_pred = lasso.predict(X_test)

# 计算均方误差
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# 检查哪些特征的系数被压缩至零
selected_features = pd.DataFrame(lasso.coef_, index=X.columns, columns=['Coefficient'])
print(selected_features[selected_features['Coefficient'] != 0])
```

问题二 Kmeans 聚类以及随机森林分类

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc, accuracy_score,mean_squared_error
import matplotlib.pyplot as plt
import matplotlib
import warnings
import numpy as np
import math
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
# 忽略警告信息
warnings.filterwarnings("ignore")
# 读取 CSV 文件
df = pd.read_excel("C:/Users/30766/Desktop/5.xlsx")  #
import pandas as pd
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
features = df.drop(['id', '洪水概率'], axis=1)

# 选择洪水概率列作为聚类的目标
flood_probabilities = df['洪水概率']

# 标准化特征数据
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# 应用 K-means 聚类
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features_scaled)

# 将聚类标签添加到原始数据中
df['cluster'] = kmeans.labels_

# 可视化聚类结果（如果特征是二维或三维的）
if features.shape[1] <= 3:
    plt.scatter(features_scaled[:, 0], features_scaled[:, 1], c=kmeans.labels_,
cmap='viridis')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    if features.shape[1] == 3:
        plt.zlabel('Feature 3')
    plt.title('K-means Clustering')
    plt.show()

# 打印聚类结果的统计信息
print(df.groupby('cluster').mean())

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc, accuracy_score,mean_squared_error
import matplotlib.pyplot as plt
import matplotlib
import warnings
import numpy as np
import math
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
```

```
# 忽略警告信息
warnings.filterwarnings("ignore")
# 读取 CSV 文件
df = pd.read_excel("C:/Users/30766/Desktop/第二问.xlsx")

 = df.drop(['cluster'], axis=1)
y = df['cluster']
from sklearn.model_selection import train_test_split, cross_val_score

# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
# 使用测试集评估模型
accuracy = rf_classifier.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")
# 预测测试集的结果
predictions = rf_classifier.predict(X_test)
import matplotlib.pyplot as plt
# 获取特征重要性
feature_importances = rf_classifier.feature_importances_

# 将特征重要性与特征名称对应起来
feature_names = X_train.columns
importances = sorted(zip(feature_names, feature_importances), key=lambda x: x[1],
reverse=True)

# 打印特征重要性
print("Feature importances:")
for feature, importance in importances:
    print(f"{feature}: {importance:.4f}")
    scores = cross_val_score(rf_classifier, X, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", scores)
print("Mean accuracy:", scores.mean())
print("Standard deviation:", scores.std())
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 0.5, 0.7]
}
```

```
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X, y)
print("Best parameters:", grid_search.best_params_)
print("Best cross-validation score:", grid_search.best_score_)

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# 使用 GridSearchCV 找到的最优参数
best_params = grid_search.best_params_

# 使用最优参数重新训练随机森林模型
rf_classifier_best = RandomForestClassifier(**best_params, random_state=42)
rf_classifier_best.fit(X_train, y_train)

# 计算特征重要性
feature_importances = rf_classifier_best.feature_importances_
feature_names = X_train.columns
importances = sorted(zip(feature_names, feature_importances), key=lambda x: x[1], reverse=True)

# 打印特征重要性
print("特征重要性:")
for feature, importance in importances:
    print(f"{feature}: {importance:.4f}")

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# 设置 Seaborn 的绘图风格为 whitegrid
sns.set(style="whitegrid")
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
# 构建数据框结构
data = pd.DataFrame({
    'group': ['基础设施恶化', '人口得分', '农业实践', '森林砍伐', '城市化'],
    'value': [0.3341, 0.2421, 0.1653,0.1412, 0.1173]
})

# 定义颜色
colors = ['#CCE5F3', '#F2C5C6', '#FFD2BB', '#c6deb5', '#ffc2ff']
```

```
# 创建一个图表，包含两个子图
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7))  # 1 行 2 列


# 创建饼图，显示在左边
data['value_normalized'] = data['value'] / data['value'].sum()
ax2.pie(data['value_normalized'], colors=colors, labels=data['group'], autopct='%1.1f%%',
startangle=140)
ax2.set_title("Pie Chart")  # 设置标题
ax2.set_aspect('equal')  # 使饼图的 aspect ratio 相等

plt.tight_layout()  # 调整子图布局以避免重叠
plt.show()
```

问题三最优模型选取

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc, accuracy_score,mean_squared_error
import matplotlib.pyplot as plt
import matplotlib
import warnings
import numpy as np
import math
from sklearn.feature_selection import mutual_info_regression
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.ensemble import VotingRegressor, StackingRegressor
from sklearn.model_selection import TimeSeriesSplit
from optuna.samplers import TPESampler
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
import matplotlib.pyplot as plt
from sklearn.base import clone
import seaborn as sns
import warnings
import optuna
import shutil
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
from sklearn.base import clone
```

```
import numpy as np
import pandas as pd
warnings.filterwarnings('ignore')
N_FOLDS = 10
N_REPEATS = 3
SEED = 27
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']   # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
# 忽略警告信息
warnings.filterwarnings("ignore")
# 读取 CSV 文件
df = pd.read_excel("C:/Users/30766/Desktop/问题四.xlsx")

from sklearn.model_selection import train_test_split
X = df.drop(['洪水概率'], axis=1)
y = df['洪水概率']

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.model_selection import StratifiedKFold, KFold

class Trainer:
    def __init__(self, model, n_folds=5, n_repeats=1):
        self.model = model
        self.n_folds = n_folds
        self.n_repeats = n_repeats

    def fit(self, X, y):
        print(f'Training {self.model.__class__.__name__}')
        start_idx = None
        if self.model.__class__.__name__ in ['LogisticRegression', 'SVC']:
            kf = StratifiedKFold(n_splits=self.n_folds)
        else:
            kf = KFold(n_splits=self.n_folds)

        scores = []
        oof_preds = np.zeros(len(X), dtype=float)
        for fold_idx, (train_idx, val_idx) in enumerate(kf.split(X, y)):
            X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
            y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]
            if start_idx is None:
                start_idx = val_idx[0]
            y_pred = np.zeros_like(y_val, dtype=float)
```

```
                for i in range(self.n_repeats):
                        m = clone(self.model)
                        if self.n_repeats > 1:
                                try:
                                        m.set_params(random_state=i)
                                except:
                                        pass
                        m.fit(X_train, y_train)
                        y_pred += m.predict(X_val)

                y_pred /= self.n_repeats

                score = mean_absolute_percentage_error(y_val, y_pred)
                scores.append(score)

                print(f'--- Fold {fold_idx + 1} - MAPE: {score:.6f}')

                oof_preds[val_idx] = y_pred

        print(f'\n------ MAPE: {np.mean(scores):.6f} ± {np.std(scores):.6f}\n\n')

        return m, scores, oof_preds,start_idx

    def fit_predict(self, X, y, X_test):
        y_pred = np.zeros(len(X_test), dtype=float)
        for i in range(self.n_repeats):
            m = clone(self.model)
            if self.n_repeats > 1:
                try:
                    m.set_params(random_state=i)
                except:
                    pass
            m.fit(X, y)
            y_pred += m.predict(X_test)
        y_pred /= self.n_repeats
        return y_pred
cb_params = {
    "bootstrap_type": "MVS",
    "border_count": 236,
    "colsample_bylevel": 0.9895796854882428,
    "depth": 6,
    "grow_policy": "SymmetricTree",
    "iterations": 4727,
    "l2_leaf_reg": 8.243537067932515,
```

```
    "loss_function": "Quantile",
    "min_data_in_leaf": 41,
    "random_state": SEED,
    "random_strength": 0.8499226947970046,
    "subsample": 0.7467840216120686,
    "thread_count": -1,
    "verbose": False
}

lgbm_params = {
    "boosting_type": "gbdt",
    "colsample_bytree": 0.9547441579422116,
    "learning_rate": 0.04406871820205758,
    "min_child_samples": 8,
    "min_child_weight": 0.5163384032966228,
    "min_split_gain": 0.18807271127987724,
    "n_estimators": 663,
    "n_jobs": -1,
    "num_leaves": 48,
    "random_state": SEED,
    "reg_alpha": 5.4660768249665646,
    "reg_lambda": 5.974261145462608,
    "subsample": 0.039552743069246055,
    "verbose": -1
}
xgb_model = xgb.XGBRegressor()
xgb_trainer = Trainer(xgb_model)
xgb_model, xgb_scores, xgb_oof_preds,idx = xgb_trainer.fit(X_train, y_train)
xgb_test_preds = xgb_trainer.fit_predict(X_train, y_train, X_test)


cb_model = CatBoostRegressor(**cb_params)
cb_trainer = Trainer(cb_model)
cb_model, cb_scores, cb_oof_preds,idx    = cb_trainer.fit(X_train, y_train)
cb_test_preds = cb_trainer.fit_predict(X_train, y_train, X_test)


lgbm_model = LGBMRegressor(**lgbm_params)
lgbm_trainer = Trainer(lgbm_model)
lgbm_model, lgbm_scores, lgbm_oof_preds,idx    = lgbm_trainer.fit(X_train, y_train)
lgbm_test_preds = lgbm_trainer.fit_predict(X_train, y_train, X_test)

fig, axs = plt.subplots(1, 3, figsize=(20, 7))
```

```
palette = sns.color_palette("RdYlGn_r", len(X_train.columns))

xgb_importances = pd.Series(xgb_model.feature_importances_, index=X_train.columns).sort_values(ascending=False)
sns.barplot(y=xgb_importances.index, x=xgb_importances.values, orient='h', ax=axs[0], palette=palette)
axs[0].set_title('XGBoost')
axs[0].grid(True)

cb_importances = pd.Series(cb_model.feature_importances_, index=X_train.columns).sort_values(ascending=False)
sns.barplot(y=cb_importances.index, x=cb_importances.values, orient='h', ax=axs[1], palette=palette)
axs[1].set_title('CatBoost')
axs[1].grid(True)

lgbm_importances = pd.Series(lgbm_model.feature_importances_, index=X_train.columns).sort_values(ascending=False)
sns.barplot(y=lgbm_importances.index, x=lgbm_importances.values, orient='h', ax=axs[2], palette=palette)
axs[2].set_title('LightGBM')
axs[2].grid(True)

plt.tight_layout()
plt.show()

oof_preds = {
    "XGBoost": xgb_oof_preds[idx:],
    "CatBoost": cb_oof_preds[idx:],
    "LightGBM": lgbm_oof_preds[idx:],
}

def objective(trial):
    xgb_weight = trial.suggest_float('xgb_weight', 0.0, 1.0)
    cb_weight = trial.suggest_float('cb_weight', 0.0, 1.0)
    lgbm_weight = trial.suggest_float('lgbm_weight', 0.0, 1.0)

    weights = [
        xgb_weight,
        cb_weight,
        lgbm_weight,
    ]
    weights /= np.sum(weights)
```

```
        preds = np.zeros((lgbm_oof_preds[idx:].shape[0]))
        for model, weight in zip(oof_preds.keys(), weights):
            preds += oof_preds[model] * weight

        return mean_absolute_percentage_error(y_train[idx:], preds)


sampler = TPESampler(seed=SEED)
study = optuna.create_study(direction='minimize', sampler=sampler)
study.optimize(objective, n_trials=1000)

best_weights = study.best_params
best_weights = [best_weights[f'{model}_weight'] for model in ['xgb', 'cb', 'lgbm']]
best_weights /= np.sum(best_weights)

import optuna.visualization as ov

# 绘制优化历史，观察 MAPE 如何随试验次数改进
ov.plot_optimization_history(study)
from sklearn.metrics import mean_absolute_percentage_error
ensemble_preds = np.zeros_like(y_train[idx:])
for model, weight in zip(oof_preds.keys(), best_weights):
    ensemble_preds += oof_preds[model] * weight
# 计算 MAPE
oof_mape = mean_absolute_percentage_error(y_train[idx:], ensemble_preds)

# 输出 MAPE
print(f"The MAPE of the ensemble model with optimal weights is: {oof_mape:.2%}")

voting_model = VotingRegressor(
    estimators=[
        ('xgb', xgb_model),
        ('cb', cb_model),
        ('lgbm', lgbm_model),
    ],
    n_jobs=-1
)
voting_trainer = Trainer(voting_model, n_repeats=1)
voting_model, voting_scores, voting_oof_preds, idx = voting_trainer.fit(X_train, y_train)
voting_test_preds = voting_trainer.fit_predict(X_train, y_train, X_test)

weighted_voting_model = VotingRegressor(
    estimators=[
        ('xgb', xgb_model),
```

```
            ('cb', cb_model),
            ('lgbm', lgbm_model),
        ],
        weights=best_weights,
        n_jobs=-1
)
weighted_voting_trainer = Trainer(weighted_voting_model, n_repeats=1)
weighted_voting_model, weighted_voting_scores, weighted_voting_oof_preds, idx =
weighted_voting_trainer.fit(X_train, y_train)
weighted_voting_test_preds = weighted_voting_trainer.fit_predict(X_train, y_train, X_test)

stacking_model = StackingRegressor(
        estimators=[
            ('xgb', xgb_model),
            ('cb', cb_model),
            ('lgbm', lgbm_model),
        ],
        n_jobs=-1
)
stacking_trainer = Trainer(stacking_model, n_repeats=1)
stacking_model, stacking_scores, stacking_oof_preds, idx = stacking_trainer.fit(X_train,
y_train)
stacking_test_preds = stacking_trainer.fit_predict(X_train, y_train, X_test)

scores = pd.DataFrame({
        "XGBoost": xgb_scores,
        "LightGBM": lgbm_scores,
        "CatBoost": cb_scores,
        "Weighted Ensemble": 0.068,
        "Weighted Voting": weighted_voting_scores,
        "Voting": voting_scores,
        "Stacking": stacking_scores
})

mean_scores = scores.mean().sort_values(ascending=False)
x_min = mean_scores.min() - 0.05 * mean_scores.min()
x_max = mean_scores.max() + 0.05 * mean_scores.max()

sns.set_style('white')
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

sns.boxplot(data=scores, orient='h', palette='husl', ax=axes[0],
order=scores.mean().sort_values().index.tolist())
axes[0].set_title('Fold MAPE')
```

```
axes[0].set_xlabel('')
axes[0].set_ylabel('Model')

mean_scores = scores.mean().sort_values(ascending=False)
mean_scores.plot(kind='barh', ax=axes[1], color='#9370DB')
axes[1].set_title('Mean MAPE')
axes[1].set_xlabel('')
axes[1].set_ylabel('')
axes[1].set_xlim([x_min, x_max])

for i, v in enumerate(mean_scores):
    axes[1].text(v + 1e-4, i, f'{v:.6f}', va='center', fontsize=11)

plt.tight_layout()
plt.show()
```

问题三 catboost 预测

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc, accuracy_score,mean_squared_error
import matplotlib.pyplot as plt
import matplotlib
import warnings
import numpy as np
import math
from sklearn.feature_selection import mutual_info_regression
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.ensemble import VotingRegressor, StackingRegressor
from sklearn.model_selection import TimeSeriesSplit
from optuna.samplers import TPESampler
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
import matplotlib.pyplot as plt
from sklearn.base import clone
import seaborn as sns
import warnings
import optuna
import shutil
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.datasets import make_regression
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.base import clone
import numpy as np
import pandas as pd
warnings.filterwarnings('ignore')
N_FOLDS = 10
N_REPEATS = 3
SEED = 27
matplotlib.rcParams['font.family'] = 'sans-serif'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']   # 使用思源黑体
matplotlib.rcParams['axes.unicode_minus'] = False
# 忽略警告信息
warnings.filterwarnings("ignore")
# 读取 CSV 文件
df = pd.read_excel("C:/Users/30766/Desktop/问题四.xlsx")

from sklearn.model_selection import train_test_split
X = df.drop(['洪水概率'], axis=1)
y = df['洪水概率']

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

def objective(trial):
    # 定义超参数搜索空间
    iterations = trial.suggest_int('iterations', 100, 1000)
    learning_rate = trial.suggest_loguniform('learning_rate', 1e-4, 1e-1)
    depth = trial.suggest_int('depth', 3, 10)
    regularization_lambda = trial.suggest_loguniform('regularization_lambda', 1e-4, 1e2)

    # 创建 CatBoost 模型实例
    model = CatBoostRegressor(
        iterations=iterations,
        learning_rate=learning_rate,
        depth=depth,
        loss_function='RMSE',
        random_seed=42,
        l2_leaf_reg=regularization_lambda   # CatBoost 中的 L2 正则化参数
    )

    # 训练模型
    model.fit(X_train, y_train)

    # 预测测试集
```

```
    predictions = model.predict(X_test)

    # 计算负的 RMSE 作为目标值，Optuna 默认是寻找最小值
return -mean_squared_error(y_test, predictions) ** 0.5

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)

import shap

explainer = shap.TreeExplainer(best_model)
shap_values = explainer.shap_values(X_train)

shap.summary_plot(shap_values, X_train)

shap_interaction_values = explainer.shap_interaction_values(X_train)
shap.summary_plot(shap_interaction_values, X_train)

import optuna.visualization as ov

# 绘制优化历史，观察 MAPE 如何随试验次数改进
ov.plot_optimization_history(study)
ov.plot_param_importances(study)

# 创建 shap.Explanation 对象
shap_explanation = shap.Explanation(values=shap_values[0:500,:],
                                    base_values=explainer.expected_value,
                                    data=X_test.iloc[0:500,:],
feature_names=X_test.columns)
# 绘制热图
shap.plots.heatmap(shap_explanation)
```

问题四预测

```
d1 = pd.read_excel("C:/Users/30766/Desktop/6.xlsx")

import pandas as pd

# 使用模型进行预测
predictions = best_model.predict(d1)

# 将预测结果添加到新的 DataFrame 中
d1['Predicted_Values'] = predictions
d1.to_excel("C:/Users/30766/Desktop/predictions.xlsx", index=False)

mport seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import probplot

# 设置 seaborn 的全局风格和颜色方案
sns.set_style("whitegrid")  # 网格风格
sns.set_palette("deep")  # 颜色方案

# 绘制预测结果的直方图
plt.figure(figsize=(10, 6))
sns.histplot(predictions, bins=20, color='blue', kde=True)  # 添加密度曲线
plt.title('Histogram of Predictions', fontsize=14)  # 增大标题字体大小
plt.xlabel('Predicted Values', fontsize=12)  # 增大 x 轴标签字体大小
plt.ylabel('Frequency', fontsize=12)  # 增大 y 轴标签字体大小
plt.xticks(fontsize=10)  # 增大 x 轴刻度字体大小
plt.yticks(fontsize=10)  # 增大 y 轴刻度字体大小
plt.show()

# 绘制 Q-Q 图
plt.figure(figsize=(8, 8))
probplot(predictions, dist="norm", plot=plt)
plt.title('Q-Q Plot of Predictions', fontsize=14)
plt.show()

import scipy.stats as stats
import pandas as pd

# 执行 Kolmogorov-Smirnov 检验
ks_statistic, p_value = stats.kstest(predictions, 'norm', args=(predictions.mean(),
predictions.std()))

# 打印 KS 检验结果
print(f'KS 检验统计量: {ks_statistic}')
```

```
print(f"KS 检验 p 值: {p_value}")

# 解释 KS 检验结果
alpha = 0.05  # 常用的显著性水平
if p_value < alpha:
    print("在{:.2f}的显著性水平下，拒绝正态性的零假设。".format(alpha))
else:
    print("在{:.2f}的显著性水平下，不能拒绝正态性的零假设。".format(alpha))
```