

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		apmcm2410195 5

基于机器学习的洪水风险预测模型及其应用

摘要

本研究旨在对洪水灾害的数据进行分析与预测，通过数学建模和数据分析的方法，预测发生洪水灾害的概率。我们分步解决了以下四个关键问题：

对于问题 1，首先对 20 个指标与洪水发生的数据进行分析，初步分析得知该数据为定量类型数据，通过 shapiro-wilks 正态性检验判断知该附件数据不服从正态分布。又因为洪水概率这一因变量为连续型的，所以若只采用斯皮尔逊相关系数进行分析的话会很难分析出哪些指标是强相关的。所以本问我们采用了四个机器学习算法进行了相关性分析。通过四个模型的性能分析后建立了一个洪水概率的 XGBoost-SHAP 模型。引入 SHAP 可解释性模型避免因样本重采样导致结果不同，最后对结果进行特征重要性结果的归一化处理后得出明显结果。结果表明，与洪水发生密切关联的指标有（顺序从强到弱依次列出）侵蚀、大坝质量、地形排水、基础设施恶化、季风强度、城市化。与洪水发生相关性不大的指标有气候变化、人口得分。

对于问题 2，针对洪水发生概率的风险分类问题我们首先对附件 train.csv 中的洪水发生概率数据进行初步处理，然后利用 K-means 聚类算法进行聚类，由肘部法则取 k 为 3，把数据聚为高中低三个风险等级。通过描述性统计和可视化分析，展示了不同风险类别的数据特征分布。随后，使用四种机器学习模型（决策树、随机森林、XGBoost 和 GBDT）计算各特征的重要性评分，并通过灵敏度分析验证特征对预测结果的影响，确保模型的合理性和稳定性。最终，基于标准化后的特征数据和权重，建立了综合风险评分模型，为洪水风险评估和防范提供了科学依据。

对于问题 3，第（1）问基于问题 1 中指标分析的结果，选取了关键特征包括政策因素、气候变化、滑坡、基础设施恶化、城市化、大坝质量、河流管理、地形排水、森林砍伐和人口得分建立洪水发生概率的预测模型。并利用决策树、随机森林、XGBoost 和 GBDT 四种机器学习模型进行了训练和评估。通过对比分析模型的 MSE、 R^2 和 MAPE，其中 XGBoost 模型在各项评估指标上表现最佳。第（2）问我们选取了包括侵蚀、大坝质量、地形排水、基础设施恶化和季风强度在内的前五个关键指标，进行了 TPS 超参数调优，以及问题 2 中的四种机器学习模型进行优化，进一步提升了模型的性能，最终选取的模型在验证集上的 MAPE 为 7.51%。经过 XGBoos-TPE 模型优化之后，MAPE 为 0.0748。

对于问题 4，我们利用问题 2 中的三个洪水发生概率预测模型，对 test.csv 中的事件进行预测，并将结果填入 submit.csv 中。通过绘制直方图和折线图，分析了预测结果的分布。采用 Kolmogorov-Smirnov 和 Shapiro-Wilk 检验验证数据分布是否符合正态分布，结果显示数据接近正态分布但存在偏差。

关键词：shapiro-wilks；XGBoost-SHAP；TPS

一、 问题重述

1.1 问题背景

近年来，洪水灾害频繁发生，给全球造成了巨大的经济损失，洪水灾害的问题始终是一个备受关注的焦点。人为的森林破坏、土地利用的变化、河流和湖泊的淤积等因素加剧了洪水灾害的严重程度，在这种情况下，特别是，人口增长、耕地扩大、森林砍伐和城市化等人类活动进一步增加了洪水的风险。如何根据历史情况作出相应的洪水灾害的数据分析与预测尤为重要。

附件给出 `train.csv`、`test.csv` 以及 `submit.csv`，其中其中 `train` 文件中约 1048573 数据。

本文基于这些上述数据，通过统计分析和优化模型，从不同角度解决问题中提出的各个问题：

问题 1：分析并可视化 20 个指标，并回答哪些与洪水发生有密切关联，哪些相关性较弱，并探讨可能的原因。提出我们合理的建议和措施。

问题 2：将洪水发生概率聚类成高、中、低风险类别，分析它们的指标特征，计算不同指标的权重，建立预警评价模型并进行灵敏度分析。

问题 3：基于问题 1 中指标分析的结果，建立洪水发生概率的预测模型，选取合适指标预测洪水发生的概率，并验证你们预测模型的准确性并调整改进洪水发生概率的预测模型。

问题 4：基于问题 2 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。绘制 74 多万件发生洪水的概率的直方图和折线图并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

需要分析附件 `train.csv` 中的数据，分析并可视化上述 20 个指标中。我们首先需要对数据进行初步分析，初步分析该数据后得知该数据为定量类型的数据，经过 shapiro-wilks 正态性检验后判断出它们是不符合正态分布的。所以本问可以采用斯皮尔曼相关系数来进行相关性分析，但是由于洪水概率这个变量数据是一个连续值，所以如果用皮尔逊相关系数的话会很难分析出它们之间的强相关度。我们需要解决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这两个问题。所以本文将会采用几个机器学习的算法去分析特征重要度，同时需要引入 SHAP 可解释机器学习模型，这样可以避免因样本重采样导致结果不同。最后对模型进行求解之后，对数据进行归一化处理，最后得出结果并对结果进行详细分析。

2.2 问题二的分析

首先对附件 `train.csv` 中的洪水发生概率数据进行标准化处理，以确保每个特征在 K-Means 聚类过程中导致权重相同。随后应用 K-Means 算法把数据聚为高、中、低三个风险等级，先通过描述性统计和可视化分析各类数据的特征分布，然后我们使用四个机器学习模型分别计算不同特征的重要性评分，评估模型的性能，并进行灵敏度分析验证特征对预测结果的影响，确保模型的合理性和稳定性。

2.3 问题三的分析

基于问题 1 中指标分析的结果，选取了包括政策因素、气候变化、滑坡、基础设施恶化、城市化、大坝质量、河流管理、地形排水、森林砍伐、人口得分这几个指标建立洪水发生概率的预测模型。使用问题一中的四个模型重新建立洪水发生概率的预测模型。从 20 个指标中选取 SHAP 模型中的前 5 个关键指标，预测洪水发生的概率，并验证你们预测模型的准确性。然后使用超参数调优和模型堆叠的方法改进我们的洪水发生概率的预测模型。

2.4 问题四的分析

本问题将采用 Kolmogorov-Smirnov 正态性检验，来检验结果的分布是否服从正态分布。

三、 符号说明

符号	说明
q	决策树的结构函数
ω	叶子节点的权重
Normalized_Importance	归一化后的特征重要度
Importance	原始的特征重要度

四、 模型建立与求解

4.1 问题一

4.1.1 相关性分析

在进行相关性分析之前，先对附件 train 中的数据进行检查，发现并未存在缺失值等情况。对于 20 个指标中，先对部分数据进行 shapiro-wilks 正态性检验后发现，这下些指标数据的分布是不服从正态分布的，p 值远小于 0.05，有足够的证据拒绝零假设，数据不符合正态分布。进一步计算相关系数矩阵，发现这些指标数据之间有着一定的相关性。

首先编写 python 语言（代码见附件），分别进行 shap-wilks 正态性检验^[1]（ H_0 表示总体具有正态分布） $\alpha = 0.05$ ，检验结果都表明数据不服从正态分布。绘制出指标之间的相关性热力图，以森林砍伐与气候变化为例，相关系数为 0.17，它们之间存在一定的正相关，例如降水模式的改变，从而影响洪水发生的概率等。由指标间的相关性热力图可以看出，20 个指标之间的相关系数都很接近，范围在[0.170, 0.181]这个范围中间，表明这些指标大多都与洪水的发生有着密切的关联，结果见下图 1，（热力图见附件）由于 spearman 相关系数都十分接近，故很难判断出哪些指标与洪水的发生更为有着密切的关联。所以接下来本文会考虑再次对指标数据进行更为准确的相关性分析。

由于 20 个指标数据不来自正态分布，因此，采用常见的回归分析 20 个指标与洪水概率之间的关系会导致参数估计误差较大，为了解决这一问题，我们将采用 XGBoost-SHAP 模型,并计算和可视化 SHAP 值来刻画这些关系。

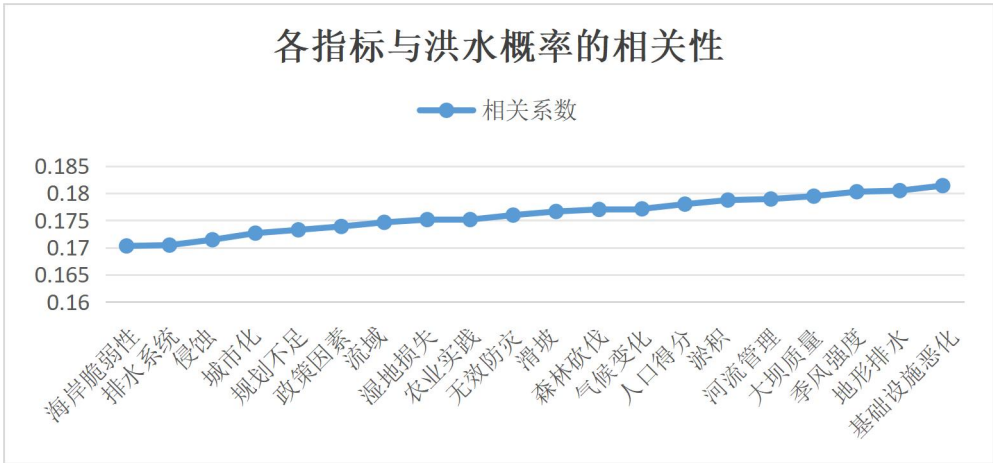


图 1 各指标与洪水发生的斯皮尔曼相关系数

4.1.2 训练四个模型对比选择

在本文中，我们比较了四种机器学习模型在预测洪水发生概率方面的性能：Decision Tree、Random Forest、XGBoost 和 GBDT。我们使用了两个评价指标： R^2 值和 MAPE 值，以衡量每个模型的预测准确性和误差。以下是每个模型的性能分析：

表 1 训练以下四个模型后得到的性能指标

模型类型	R^2 值	MAPE 值
决策树	1.0	8.63609931495659e-19
随机森林	0.9514605561820191	0.018328247837986904
XGBoost	0.8200363600581444	0.03441857278004139
GBDT	0.6158853177978914	0.05262189239966728

通过比较四种模型的性能，可以得出以下结论：

- 1. 决策树：虽然在训练数据上表现完美，但可能存在过拟合问题。
- 2. 随机森林：表现优异，具有很高的预测准确性和较低的相对误差，推荐使用。
- 3. XGBoost：表现良好，能够处理复杂数据，适用范围广泛。
- 4. GBDT：预测性能一般，需要进一步优化参数以提升模型表现。

四个种模型的特征重要度结果如下：

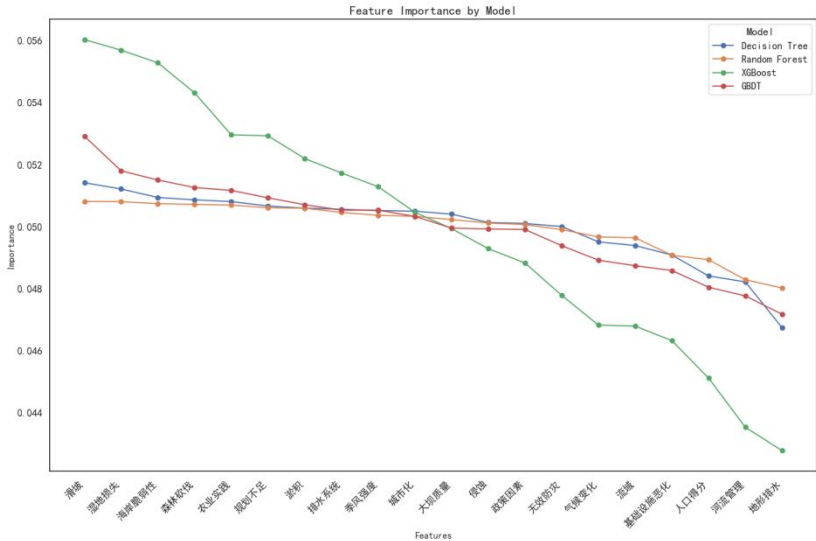


图 2 四个种模型的特征重要度结果

综合考虑，随机森林和 XGBoost 是较为可靠的选择，它们在预测准确性和误差控制方面表现出色，本文中选择了 XGBoost-SHAP 模型。

4.2 洪水概率的 XGBoost-SHAP 模型

4.2.1 数据处理

附件中的数据一共 1048576 行，采用 XGBoost-SHAP 可以缩短模型的计算时间的需求，减少数据对内存的使用提高了并行时的效率。为了完成 XGBoost-SHAP 模型的建立，按照下列方法进行指标数据的处理和说明：

首先将附件 train.csv 文件中的数据全部加载，进行 Shapiro-Wilk 正态性检验。定义 20 个指标数据特征，并且以洪水概率为目标变量，定义函数 1 来执行 Shapiro-Wilk 检验，然后计算每个特征的 Shapiro-Wilk 检验，最后得到的结果显示所有的 p 值显著性都在 0.05 以下，故所有指标均不服从正态分布。

$$W = \frac{\sum_{i=1}^n a_i Z_i}{\sqrt{\sum_{i=1}^n Z_i^2}}$$

其中， a_i 是根据样本大小 n 确定的系数。 Z_i 是标准化后的样本值。
然后对 train.csv 数据进行统计性描述分析。（结果见附录）

4.2.2 模型设定

假设数据集 D 被划分为 n 个样本和 m 个特征，每个指标数据有一个目标变量洪水概率 y_i 和特征变量 x_i 。其中 y_i 代表第 i 个指标的洪水概率， x_{ij} 代表第 i 指标的第 j 个特征。

$$\hat{y}_i = \sum_{k=1}^K g_k(\mathbf{x}_i)$$

其中 g_k 是第 k 个树模型， K 是树的总数。

XGBoost 的目标函数由损失函数 L ，和正则化项 Ω 组成：

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

其中，损失函数 l 衡量预测值 $\hat{y}_i$ 与真实值 y_i 之间的误差，常用平方误差：

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

正则化项 Ω 用于控制模型的复杂度：

$$\Omega(g_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

其中， T 是树的叶子节点数， w_j 是第 j 个叶子的权重， γ 和 λ 是正则化参数

4.2.3 模型检验

我们定义了 XGBoost 模型，并使用网格搜索进行参数调优找到性能最优的参数组合，然后最优参数在训练集上训练模型，并在测试集上进行评估，使用 SHAP 库计算特征的 SHAP 值，并绘制 SHAP Summary Plot 和 Dependence Plot（见附录），以解释每个特征对洪水发生概率的贡献。

我们使用决定系数 R^2 衡量模型解释数据变异的能力，定义为：

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

MAPE 衡量预测值与实际值之间的平均百分比误差，定义为：

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

计算得到以下结果：

XGBoost R^2 : 0.8200363600581444

XGBoost MAPE: 0.03441857278004139

4.2.4 模型表达式

把参数的后验估计带入模型表达式得到最终的 XGBoost 模型

$$\hat{y}_i = \sum_{k=1}^K g_k(\mathbf{x}_i)$$

其中，每棵树 g_k 的输出可以表示为：

$$g_k(\mathbf{x}) = w_{q(\mathbf{x})}$$

4.2.5 模型求解

XGBoost 通过梯度提升算法进行求解。假设已经构建了 t 棵树，当前模型的预测为：

$$\hat{y}_i^{(t)} = \sum_{k=1}^t g_k(\mathbf{x}_i)$$

需要最小化以下目标函数来构建第 $t+1$ 棵树：

$$\mathcal{L}^{(t+1)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)} + g_{t+1}(\mathbf{x}_i)) + \Omega(g_{t+1})$$

使用二阶泰勒展开近似损失函数：

$$\mathcal{L}^{(t+1)} \approx \sum_{i=1}^n \left[l(y_i, \hat{y}_i^{(t)}) + g_{t+1}(\mathbf{x}_i) \cdot \frac{\partial l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)}} + \frac{1}{2} g_{t+1}(\mathbf{x}_i)^2 \cdot \frac{\partial^2 l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)2}} \right] + \Omega(g_{t+1})$$

记一阶导数和二阶导数为：

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)}}, \quad h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)2}}$$

则目标函数可以近似为：

$$\mathcal{L}^{(t+1)} \approx \sum_{i=1}^n \left[g_{t+1}(\mathbf{x}_i) \cdot g_i + \frac{1}{2} g_{t+1}(\mathbf{x}_i)^2 \cdot h_i \right] + \Omega(g_{t+1})$$

计算每个特征的 SHAP 值 ϕ_{ij} ：

$$\phi_{ij} = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|! (|N| - |S| - 1)!}{|N|!} [f(S \cup \{j\}) - f(S)]$$

其中，N 是所有特征的集合，S 是特征子集，f(S) 是只考虑特征子集 S 时的模型预测。

通过 XGBoost-SHAP 进行可解释性分析，代码经过约 40 分钟得到以下结果：

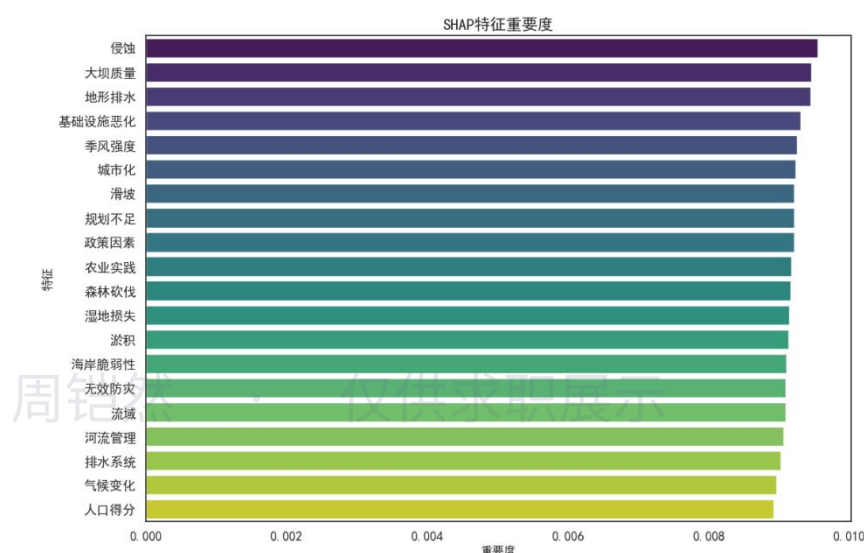


图 3 SHAP 特征重要度

在之前绘制的特征重要度图中，虽然展示了各模型对特征的重要性评分，但由于评分值的差异较小，各特征之间的重要性并不明显。因此，我们通过归一化处理将特征重要度标准化到同一尺度，使得不同特征的重要性对比更加直观和明显。首先，我们将特征名称和 SHAP 特征重要度存储在一个数据框中，以结构化的方式存储数据，方便后续的归一化处理和绘图。使用归一化公式将每个特征的重要度值标准化到 0 到 1 的范围。归一化公式如下：

$$\text{Normalized_Importance} = \frac{\text{Importance} - \min(\text{Importance})}{\max(\text{Importance}) - \min(\text{Importance})}$$

(1) 结果分析：归一化后的结果如下

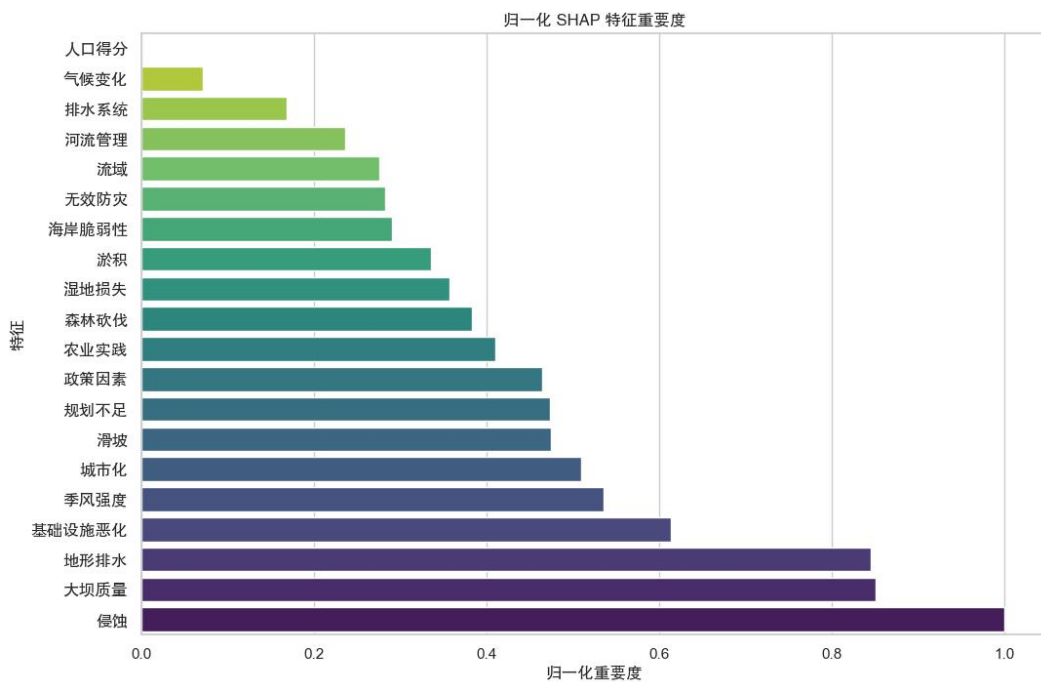


图 4 归一化 SHAP 特征重要度

本文将重要度大于 0.5 的指标视为与洪水发生有着密切关联，而小于 0.5 的指标视为与洪水发生相关性不大的指标

结果表明：

与洪水发生密切关联的指标有（顺序从强到弱依次列出）：侵蚀、大坝质量、地形排水、基础设施恶化、季风强度、城市化

与洪水发生相关性不大的指标：气候变化、人口得分

(2) 原因分析

1. 侵蚀通常会导致土壤稳定性降低，增加地表径流，减少雨水渗透，从而增加洪水发生的风险。^[2]

2. 大坝质量直接影响洪水的控制和管理。质量差的大坝容易在洪水期间破裂，导致灾难性洪水。^[3]

3. 地形排水系统的有效性决定了雨水能否迅速排走。如果排水不畅，容易形成积水 and 洪水。^[2]

4. 季风强度直接影响降雨量，强季风会带来大量降水，增加洪水发生的概率。

5. 城市化导致的不透水地面增加，雨水难以渗透，增加地表径流，导致洪水发生。

而对于与洪水发生相关性不大的两个指标来说，尽管气候变化对长期气候模式有影响，但在短期内其对洪水发生的直接影响较小。人口密度和结构对洪水发生的物理过程没有直接影响，但会影响洪水的社会经济影响。

(3) 预防建议和措施

针对以上原因的研究，本文提出了以下预防建议和措施：

① 第一点建议是需要加强土壤和河岸管理。

措施：实施植被恢复和土壤保护措施、减少侵蚀、土壤流失并且增强地表的雨水渗

透能力。这有助于减少地表径流和洪水风险。

② 第二点建议是要定期检查和维修大坝
措施：定期对大坝进行检查和维护，确保大坝结构的安全性，提高大坝应对洪水的能力。确保大坝的设计和建造符合最高安全标准。

③ 第三点建议是需要改进城市排水系统
措施：优化城市排水系统设计，增加渗透性地面和绿地，减少地表径流，降低城市洪水风险。推广绿色基础设施，如绿色屋顶和雨水花园。

④ 第四点建议是提升基础设施建设
措施：更新和改进老化的基础设施，确保排水系统和其他关键设施在极端天气事件中的有效性。采用现代化技术来提升基础设施的耐久性和效率。

⑤ 第五点建议是加强气象监测和预警
措施：增强季风和极端天气的监测能力，建立及时有效的预警系统，提前采取防洪措施。利用先进的气象预报技术和大数据分析，提高预警的准确性。

⑥ 第六点建议是推进生态城市建设
措施：推广生态城市建设，减少不透水地面的比例，增加绿地和水体，增强城市的水文循环和雨水管理能力。利用自然基础设施（如湿地和森林）来管理水资源和减少洪水风险。

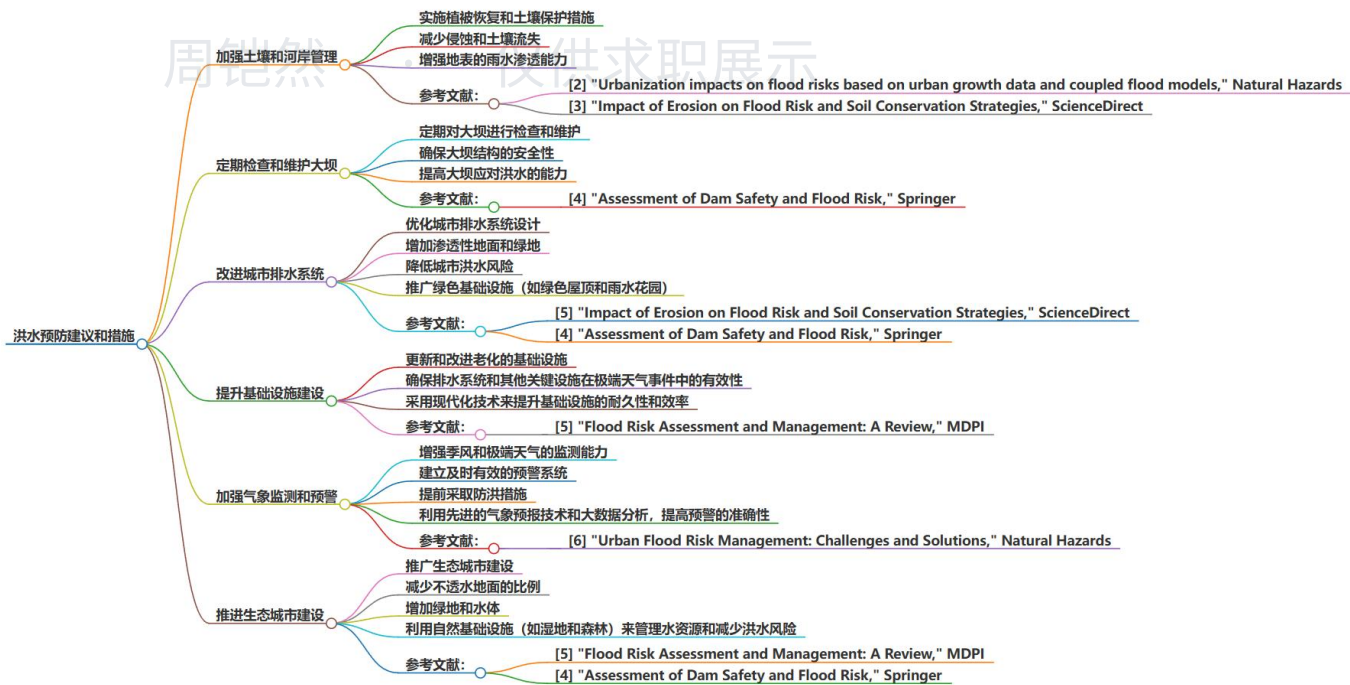


图 5 六点合理的建议和措施脑图

4.3 问题二

4.3.1 数据准备

首先导入数据，然后绘制频率分布直方图和概率密度图。因此我们可以了解洪水概

率数据的分布特征，如是否服从正态分布、是否存在偏态等。这对后续的建模和分析非常重要，因为许多机器学习算法假设输入数据服从某种分布。了解数据分布有助于我们在选择模型参数时做出更明智的决策。例如，知道数据是否有长尾分布可以帮助我们调整 K-means 聚类算法中的参数，如簇中心的初始位置等。

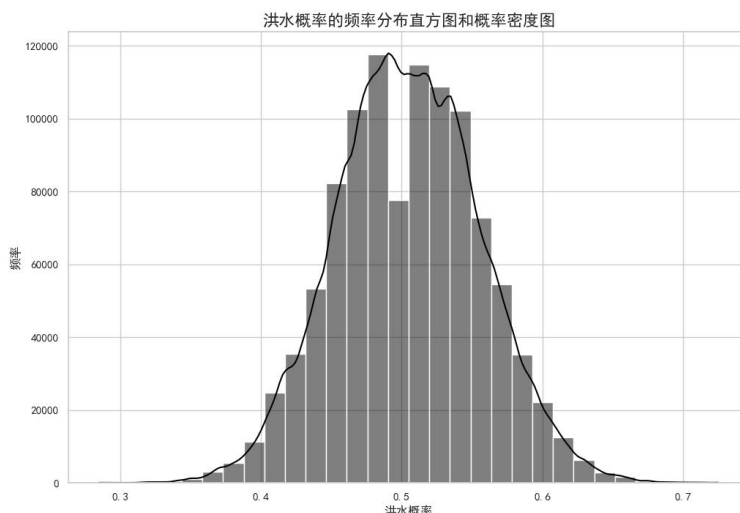


图 6 洪水概率符合正态分布

根据图形展示的结果可以看出，该洪水发生的概率数据是一个正态分布。可以进一步进行聚类分析，将洪水概率分为高、中、低风险类别，并结合特征指标分析各类别的特征，以制定针对性的洪水防范措施。

4.3.2 K-means 聚类分析洪水概率

K-means 模型，是一种基于划分方法聚类的，将样本与质点的距离作为数据对象间相似度的衡量标准，即数据对象间的距离越小，则它们的相似度越高，将指定样本聚成同一个类簇。train.csv 数据中包含洪水事件的特征数据和洪水发生概率，我们将使用这些数据进行 K-means 聚类分析。

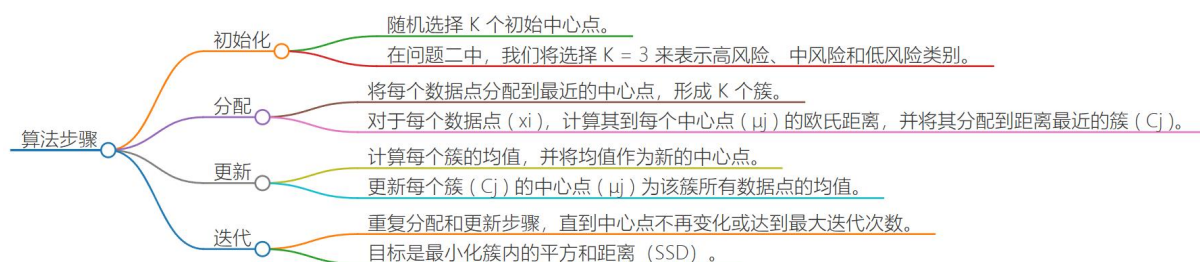


图 7 K-means 算法步骤

K-means 是一种非监督学习算法，常用于聚类分析。其目的是将数据集分成 K 个簇 (clusters)，使得簇内样本的相似度最大，簇间样本的相似度最小。其目标函数为：

$$J = \sum_{i=1}^k \sum_{j=1}^{n_i} \|x_j^{(i)} - \mu_i\|^2$$

其中：

k 是簇的数量。

n_i 是第 i 个簇中的数据点数量。

$x_j^{(i)}$ 是第 i 个簇中的第 j 个数据点。
 μ_i 是第 i 个簇的中心点（均值）

距离计算欧氏距离：

$$d(x,y)=\sqrt{\sum_{i=1}^n(x_i-y_i)^2}$$

簇内方差，度量簇内数据点的紧密度：

$$J_i=\sum_{j=1}^{n_i}\|x_j^{(i)}-\mu_i\|^2$$

总目标函数为所有簇内方差之和：

$$J=\sum_{i=1}^k J_i$$

使用 K-Means 算法对洪水概率进行聚类。根据肘部法则所显示，取 $K=3$ ，将数据分为高、中、低三类风险。

基于标准化后的特征数据和权重，计算综合风险评分：

$$\text{风险评分}=\sum_{i=1}^n w_i \cdot x_i'$$

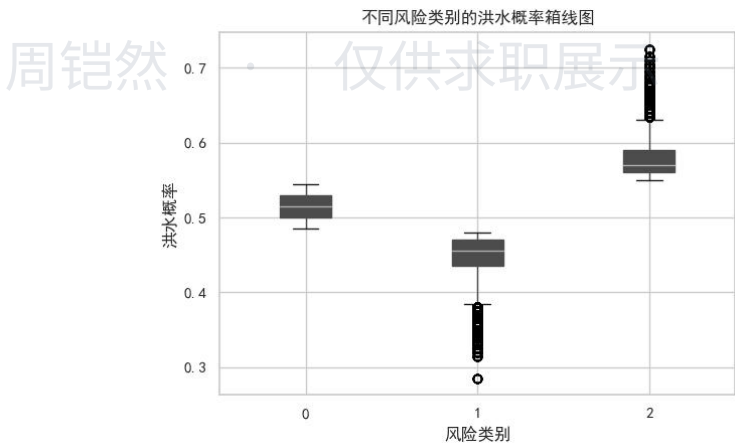


图 8 不同风险类别的洪水概率箱线图

该箱线图展示了三类风险（0，1，2）中洪水概率的分布情况。高风险类别（0）的洪水概率较高，中位数约为 0.5，并存在一些高于 0.7 的异常值；中等风险类别（1）的洪水概率也集中在 0.45 到 0.55 之间，中位数约为 0.5，但有一些低于 0.4 的异常值；低风险类别（2）的洪水概率主要集中在 0.4 到 0.5 之间，中位数约为 0.45，几乎没有异常值。这表明不同风险类别的洪水概率分布特征明显，识别这些特征有助于制定相应的防洪措施，高风险区域需要更严格的防洪措施，而中低风险区域则需适当维护和改善现有措施。

然后我们对各个风险类别的进行了描述性统计，得到以下结果：

表 2 描述性统计

风险类别	count	mean	std	min	25%	50%	75%	max
0	310465	0.564417	0.026394	0.535	0.545	0.56	0.58	0.725
1	282202	0.442312	0.024768	0.285	0.43	0.45	0.46	0.47
2	455908	0.502121	0.017062	0.475	0.485	0.5	0.515	0.53

通过分析这些统计数据，可以更好地理解不同风险类别的洪水概率特征，制定有针对性的防洪策略，提高防洪效果。高风险类别的洪水概率最高且波动性最大，需要重点关注和防范。中等风险类别的洪水概率最低且波动性适中，需要适当的防范措施，确保洪水风险在可控范围内。低风险类别的洪水概率居中且波动性最小，可以保持现有的防洪措施，并进行定期检查和维护。

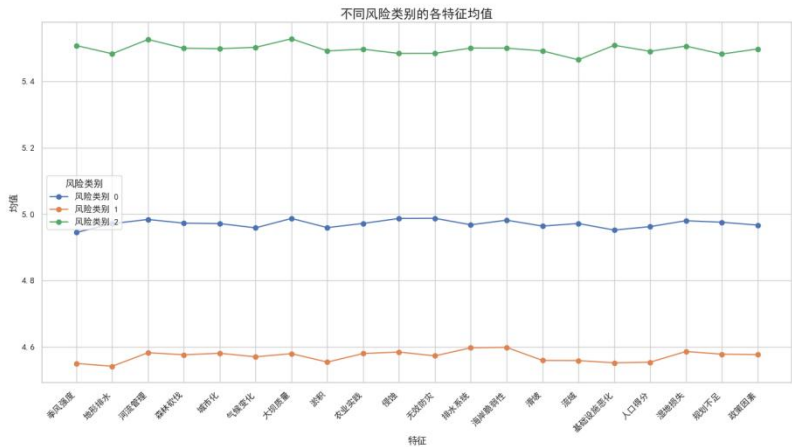


图 9 不同类别的各特征均值

4.3.3 决策树回归的预警评价模型

通过决策树回归模型，可以预测不同区域的洪水发生概率，从而制定相应的预防和应对措施。通过特征重要性分析，识别对洪水风险影响最大的因素，以便采取更有效的防洪措施。

我们把 train.csv 数据全部加载读取，对数据进行了训练。我们的目标是根据这些特征预测洪水发生的概率。洪水概率是我们的目标变量。

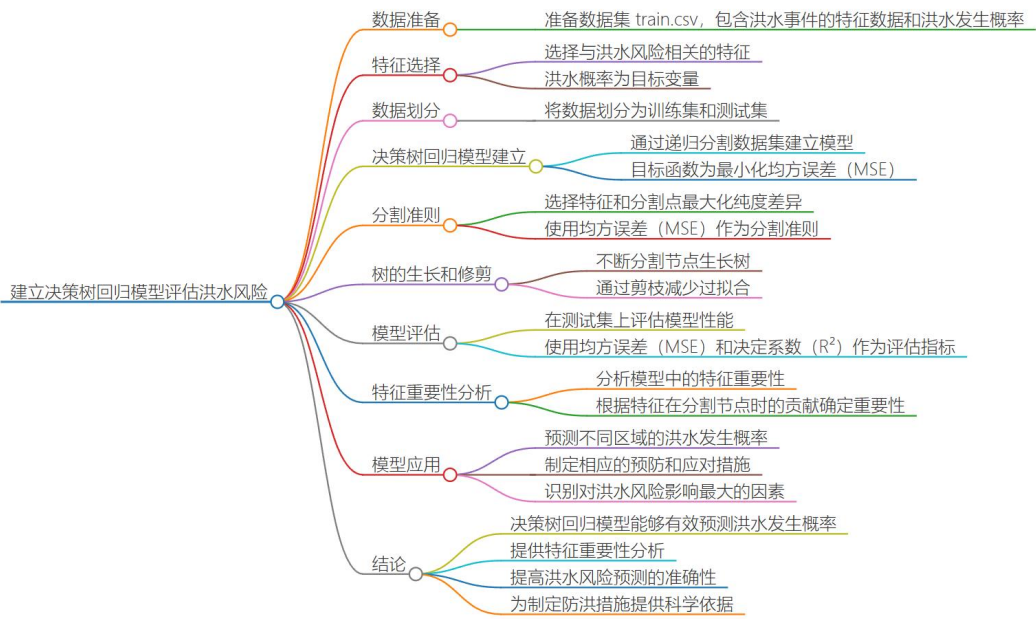


图 10 建立决策树回归模型思维过程

1. 数据准备

我们有数据集 $\mathcal{D}$ ，其中包含特征矩阵 X 和目标变量 y 。

$$\mathcal{D} = \{(X_i, y_i)\}$$

其中, X_i 是第 i 个数据点的特征向量, y_i 是其对应的洪水概率。
将数据集划分为训练集和测试集:

$$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}} = \text{train_test_split}(\mathcal{D}, \text{test_size} = 0.2)$$

决策树回归模型通过递归地将数据集分割成更小的子集来建模。在每个节点, 我们选择一个特征 f 和一个分割点 s 来最大化分割前后数据集的纯度差异。

2. 模型建立

设目标函数为最小化均方误差 (MSE) :

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

其中, N 是数据点的数量, y_i 是实际值, $\hat{y}_i$ 是预测值。

3. 分割准则

在每个节点, 我们选择特征 f 和分割点 s 来最大化分割前后数据集的纯度差异, 通常使用均方误差 (MSE) 作为分割准则:

$$\text{MSE}_{\text{split}} = \frac{1}{N_{\text{left}}} \sum_{i \in \text{left}} (y_i - \hat{y}_{\text{left}})^2 + \frac{1}{N_{\text{right}}} \sum_{i \in \text{right}} (y_i - \hat{y}_{\text{right}})^2$$

其中, N_{left} 和 N_{right} 分别是左子集和右子集的数据点数量, $\hat{y}_{\text{left}}$ 和 $\hat{y}_{\text{right}}$ 分别是左子集和右子集的平均值。

4. 树的生长和修剪

模型通过不断分割节点来生长, 直到达到停止条件, 如最大深度或最小分割点数量。然后, 通过剪枝来减少过拟合, 剪枝通过删除不显著的分支来简化树结构。

5. 模型评价

在测试集上评估模型性能, 使用均方误差 (MSE) 和决定系数 (R^2) 作为评估指标:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}$$

其中, $\bar{y}$ 是实际值的均值。

表 3 分类报告

	precision	recall	f1-score	support
0	0.93	0.93	0.93	92783
1	0.91	0.91	0.91	84486
2	0.92	0.92	0.92	137304
accuracy	0.92	0.92	0.92	314573
macro avg	0.92	0.92	0.92	314573
weighted avg	0.92	0.92	0.92	314573

从分类报告中可以看出，模型在所有类别上的表现都非常好，具有高精度、高召回率和高 F1 分数。总体准确率为 0.92，说明模型在绝大多数情况下都能正确分类。模型的宏平均和加权平均也表明其在各个类别上的表现较为均衡，没有明显的偏差。

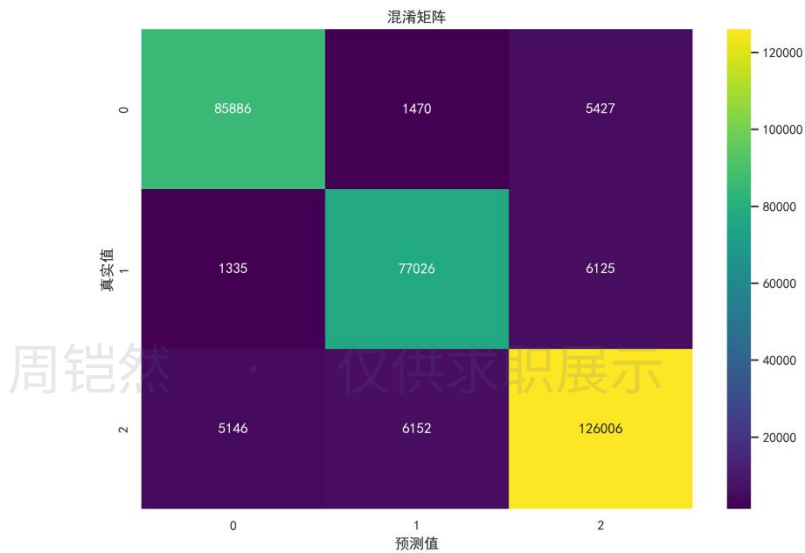


图 11 混淆矩阵

6. 特征重要性分析

决策树模型提供特征重要性，即特征在分割节点时的贡献。特征重要性可以用特征在分割过程中降低的均方误差来表示：

$$\text{Importance}(f) = \sum_{j \in \text{nodes_with_feature_f}} \Delta \text{MSE}_j$$

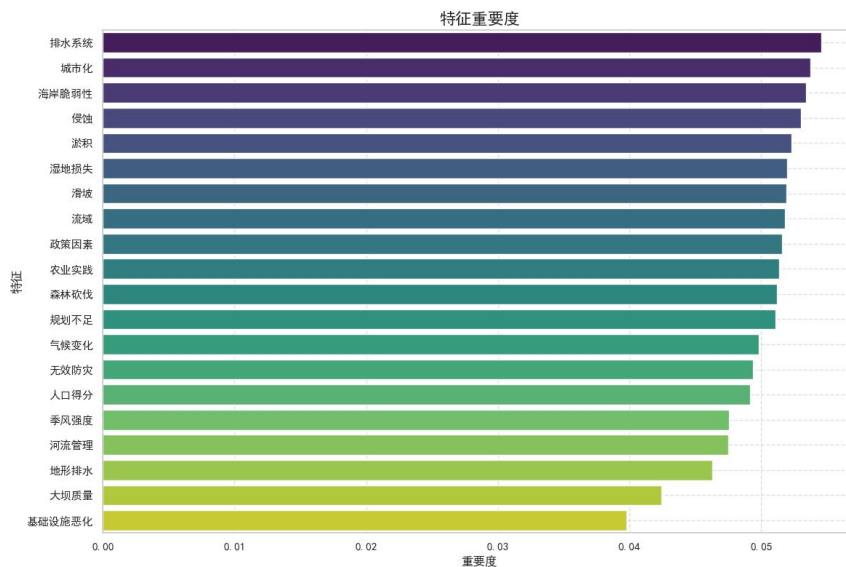


图 12 特征重要度（权重）

结果分析：我们把输出的特征重要度作为不同指标的权重，特征重要性被按照从高到低的顺序排列，每个特征的重要性值表示了它在模型中的贡献度。

不同特征对洪水风险的影响程度不一。排水系统、城市化、海岸脆弱性和侵蚀是最重要的特征，需要重点关注和管理。政策措施和规划的有效性也对洪水风险有显著影响。综合考虑这些特征，构建一个有效的洪水预警评价模型可以帮助制定科学的防洪措施。



图 13 洪水不同风险的建议措施

4.3.4 灵敏度分析

为了评估每个特征对洪水风险预测的影响，我们对 20 个特征进行了灵敏度分析，通过改变单个特征的值并记录模型的平均预测结果，我们能够直观地了解每个特征在不同取值下对模型预测的影响。结果显示在图表中，帮助我们识别出哪些特征在预测模型中起关键作用，从而优化模型并制定更有效的防洪措施。

(1) 单个特征的灵敏度分析

对于单个特征的灵敏度分析，我们设定一个特征值范围，并计算该特征值变化对模型预测的影响。假设我们有一个模型 f 和一个数据集 X ，对于某个特征 `feature`，其值范围为 `feature_range`，我们进行以下计算：

数据集为 X ，对于每个特征值 v 在 `feature_range` 进行以下操作，：

$$X_{\text{temp}} = X$$

将特征 `feature` 的值设为 v ，使用模型进行预测：

$$X_{\text{temp}}[\text{feature}] = v$$

$$\hat{y} = f(X_{\text{temp}})$$

计算平均预测值：

$$\bar{y}(v) = \frac{1}{n} \sum_{i=1}^n \hat{y}_i$$

最后将结果存入列表：`mean_predictions.append( $\bar{y}(v)$ )` 我们循环调用单个特征的灵敏度分析函数进行了全特征的灵敏度分析，结果见附录。

4.4 问题 3 (1)

基于问题 1 中指标分析的结果，我们选取了包括政策因素、气候变化、滑坡、基础设施恶化、城市化、大坝质量、河流管理、地形排水、森林砍伐、人口得分这几个指标建立洪水发生概率的预测模型。

我们建立了一个重新建立了问题一中的四个模型进行预测，建立的过程这里不再过多赘述。

首先，我们将数据进行了将数据集按照 70% 的训练集和 30% 的测试集进行划分并通过设置 `random_state=0` 确保数据拆分的随机性是可重复的。

四个不同模型的评估结果对比如下：

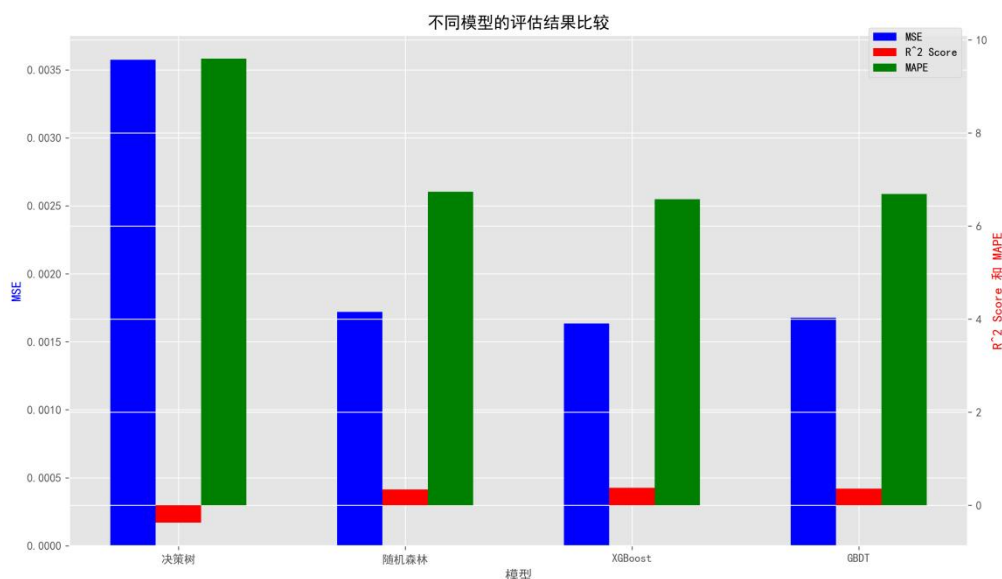


图 14 评估结果比较

基于以上分析，我们优先选择 XGBoost 模型进行洪水发生概率的预测，因为它在所有评估指标上均表现最佳。GBDT 作为次优选择，其次是随机森林，而决策树模型在此场景下并不适合。

XGBoost 模型的 MSE 最低，约为 0.00163，表示模型的预测误差最小。R² Score，最高，约为 0.37，模型具有最强的解释力。MAPE 约为 6.58%。

4.5 问题 3 (2)

我们选取了问题一的 SHAP 模型中的前 5 个关键指标，有侵蚀、大坝质量、地形排水、基础设施恶化、季风强度。然后重新构建模型进行训练，训练结果如下：

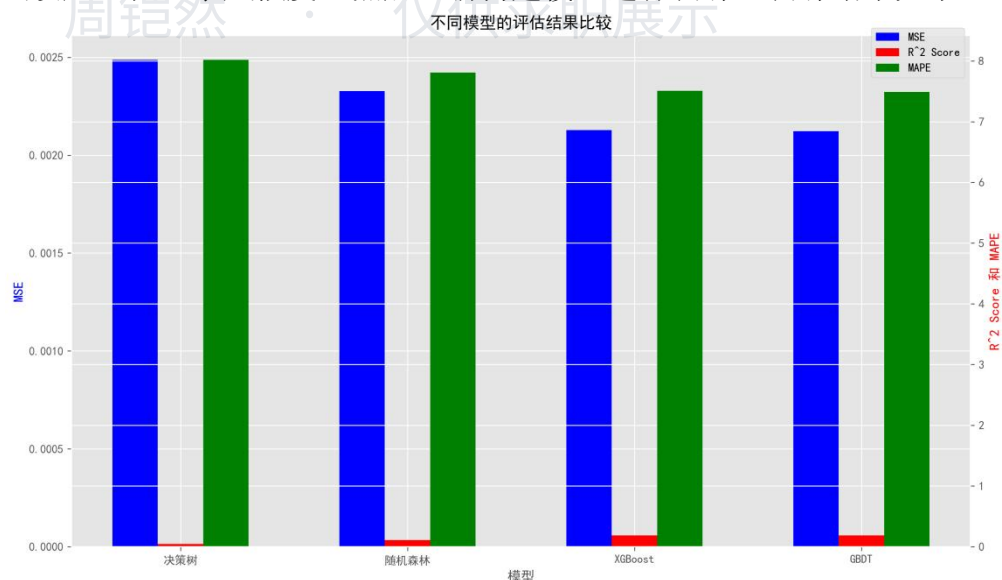


图 15 选取五个关键指标后的对比

基于以上结果，我们优先选择了 XGBoost 模型进行超参数优化。

4.5.1 超参数调优 TPETree-structured Parzen Estimator 模型

使用 hyperopt 库对 XGBoost 模型进行超参数优化，以最小化均绝对百分比误差 (MAPE)。将数据集拆分为训练集和测试集，例如 80% 的数据用于训练，20% 的数据用于测试。

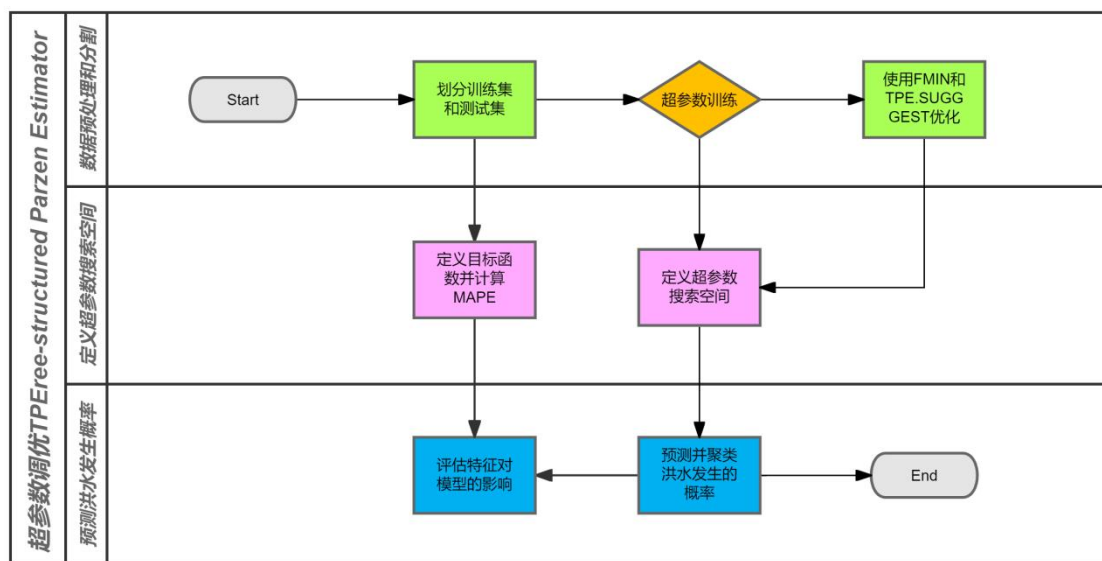


图 16 超参数调优过程

每棵树随机采样的特征比例。68.8% 的特征用于训练，学习速率 0.094，树的最大深度为 3，有 426 棵树，82.9% 的样本用于训练。模型在调优后的最佳均绝对百分比误差 MAPE 为 7.49%。经过 TPE 超参数优化后，XGBoost 模型在准确性和泛化能力上得到了一定的提升。

接着，我们还对上述 decision_tree、gbdt 和 xgboost 模型进行了模型堆叠，然后再重新对洪水发生概率进行预测，结果显示与超参数优化结果接近。

4.6 问题 4

问题四我们将采用 Kolmogorov-Smirnov 正态性检验对结果进行检验，分析此结果的分布是否服从正态分布。

首先我们基于问题 2 中建立的 XGBoost 洪水发生概率预测模型预测附件 text.csv 中所有事件发生洪水的概率。

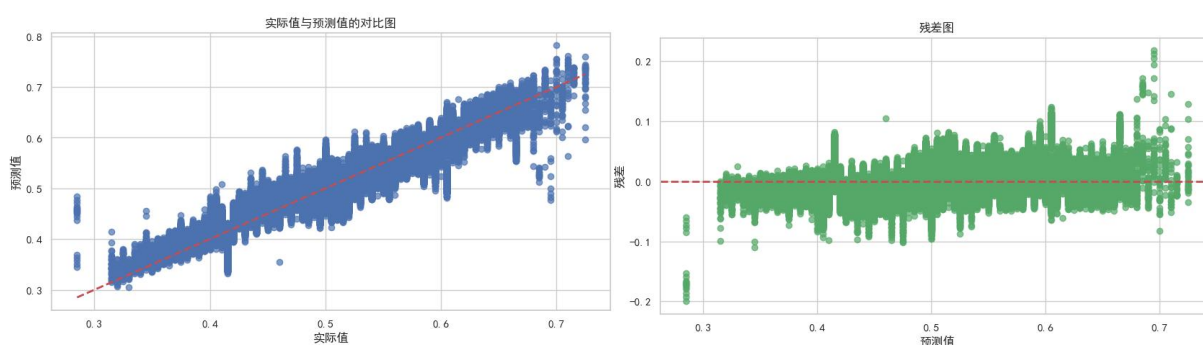


图 17 XGBoost 预测的对比图与残差图

然后我们先预测结果填入附件 submit.csv 中，并绘制这 74 多万件发生洪水的概率的直方图和折线图如下：

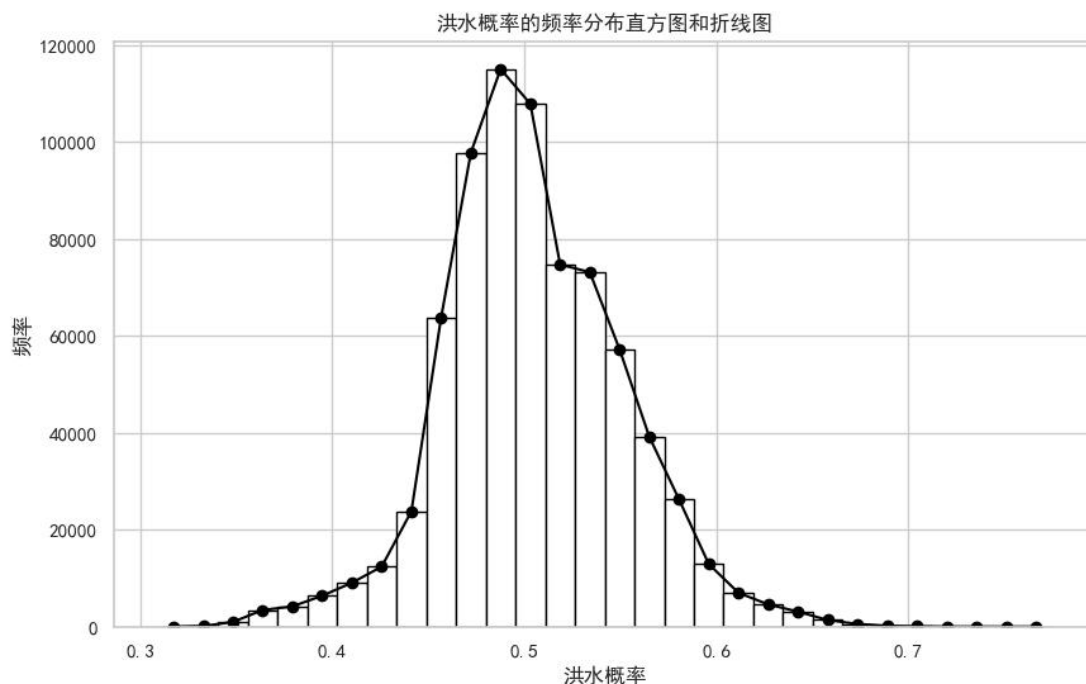


图 18 洪水的频率分布直方图和折线图

数据呈现出一个接近钟形的分布，乍看之下可能被误认为是正态分布。然而，外观上接近正态分布并不等同于统计意义上的正态性。统计检验能够更准确地评估数据是否符合正态分布。

最后我们对预测结果进行 Kolmogorov-Smirnov 检验，检验统计量为 0.04324342686295046，p 值为 0.0。Shapiro-Wilk 正态性检验结果，检验统计量为 0.990226686000824，p 值为 0.0。由于 p 值远小于 0.05，拒绝数据符合正态分布的假设。

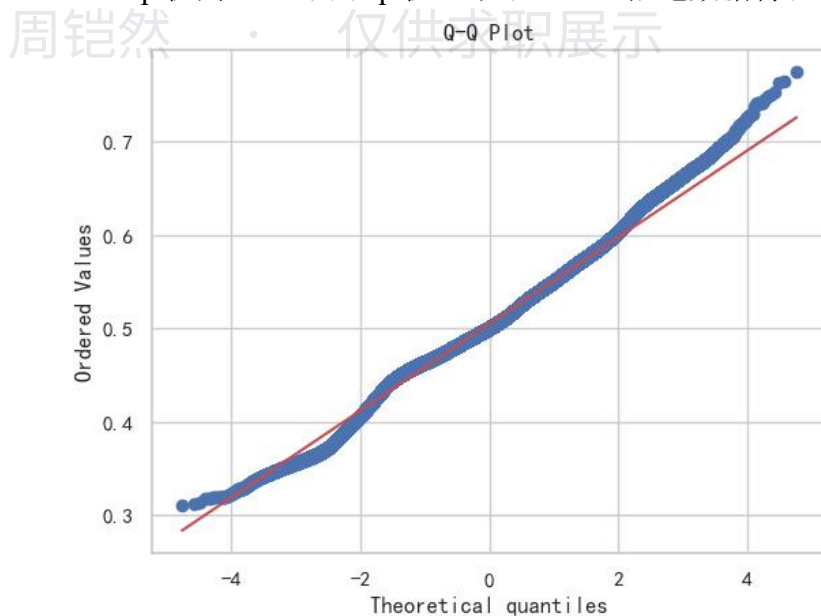


图 18 Q-Q 图

从 Q-Q 图来看，数据点总体上沿着对角线分布，但在极端值处有一些偏离。这进一步表明数据接近正态分布，但存在一些偏差。以下是对这一现象的总结：

Q-Q 图显示洪水概率数据大致沿对角线分布，但在极端值（尾部）存在一些偏离。这表明数据总体上接近正态分布，但存在一定的偏差。尽管从直观上看数据接近正态分

布，Kolmogorov-Smirnov 检验和 Shapiro-Wilk 检验结果均显示数据不符合正态分布，可能是因为这些检验对细微偏差较为敏感，尤其在样本量较大时更容易检测到偏离。

综上，结果数据总体上服从正态分布。

五、 模型的评价及推广

5.1 模型的优点

- 1) 模型统一，通用性强
- 2) 优化合理，结果可靠
- 3) 模型简单易懂，方法灵活，具有较强的推广性。

5.2 模型的推广

本模型可以帮助城市规划者识别高风险区域，制定针对性的防洪措施，优化城市排水系统，提高城市的抗洪能力。

使用本模型，可以评估大坝、排水系统等基础设施的安全性和有效性，指导基础设施的设计、建设和维护。

通过模型分析，可以识别森林砍伐、土地侵蚀等环境问题对洪水风险的影响，指导生态修复和环境保护工作，减少人为因素对洪水发生的影响。

周铠然 · 仅供求职展示

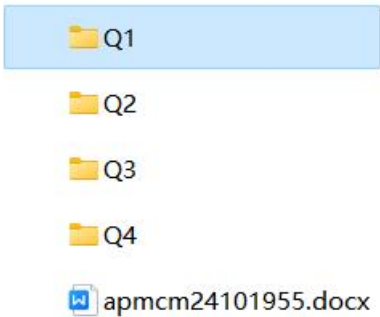
六、 参考文献

- [1] MuhammadAslam,"AnalysingGrayCastIronDataUsingaNewShapiro-WilksTestforNormalityunderIndeterminacy,"InternationalJournalofCastMetalsResearch,34(1):1 – 5,2020.
- [2] "Urbanization impacts on flood risks based on urban growth data and coupled flood models,"NaturalHazards, accessed on July 7, 2024.
- [3] "Impact of Erosion on Flood Risk and Soil Conservation Strategies," ScienceDirect, accessed on July 7, 2024.
- [4] "Assessment of Dam Safety and Flood Risk," Springer, accessed on July 7, 2024.
- [5] "Flood Risk Assessment and Management: A Review," MDPI, accessed on July 7, 2024.
- [6] "Urban Flood Risk Management: Challenges and Solutions," Natural Hazards, accessed on July 7, 2024.

附录

附录 1

介绍：所有文件



Q1:

名称	修改日期	类型	大小
.ipynb_checkpoints	2024-07-08 15:04	文件夹	
correlation_heatmap.png	2024-07-07 15:21	PNG 文件	837 KB
feature_importance_by_model.png	2024-07-07 21:18	PNG 文件	146 KB
Q1 (修改) .ipynb	2024-07-07 23:21	Jupyter 源文件	837 KB
shap_feature_importance.png	2024-07-07 21:32	PNG 文件	66 KB
shap_feature_importance_new_style....	2024-07-07 21:52	PNG 文件	61 KB
train.csv	2024-07-04 23:54	XLS 工作表	55,115 KB
问题一合理的建议和措施脑图.html	2024-07-07 23:02	Microsoft Edge ...	6 KB

Q2:

名称	修改日期	类型	大小
.ipynb_checkpoints	2024-07-08 15:04	文件夹	
confusion_matrix.png	2024-07-08 13:52	PNG 文件	102 KB
data.csv	2024-07-08 15:03	XLS 工作表	57,163 KB
flood_risk_boxplot.png	2024-07-08 12:50	PNG 文件	21 KB
K-means算法步骤.html	2024-07-08 3:38	Microsoft Edge ...	3 KB
Q2 (修改) .ipynb	2024-07-08 15:00	Jupyter 源文件	766 KB
train.csv	2024-07-04 23:54	XLS 工作表	55,115 KB
不同风险类别的各特征均值.png	2024-07-08 3:12	PNG 文件	105 KB
洪水概率的频率分布直方图和概率密度图...	2024-07-08 3:43	PNG 文件	55 KB
决策树回归模型思维脑图.html	2024-07-08 13:09	Microsoft Edge ...	4 KB
决策树结构图.pdf	2024-07-05 0:38	WPS PDF 文档	225,129 KB
灵敏度分析.html	2024-07-08 14:36	Microsoft Edge ...	10 KB
敏感性分析.png	2024-07-08 14:28	PNG 文件	212 KB
特征重要性.png	2024-07-08 14:09	PNG 文件	68 KB
预测评价模型.html	2024-07-08 14:22	Microsoft Edge ...	8 KB

Q3:

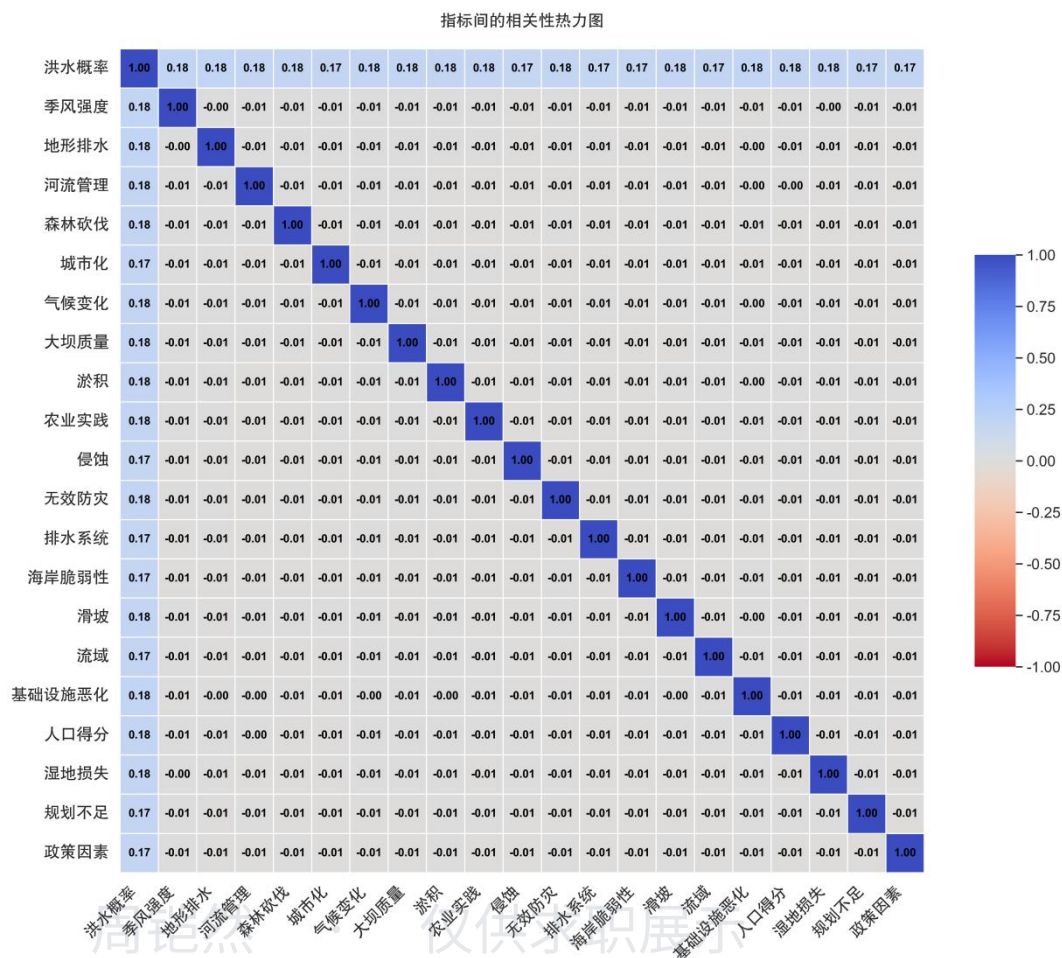
-  .ipynb_checkpoints
-  model_evaluation_comparison.png
-  model_evaluation_comparison_updated.png
-  Q3 (修改1) .ipynb
-  Q3 (修改2) .ipynb
-  TPE超参数调优.png
-  train.csv

Q4:

-  .ipynb_checkpoints
 -  GDBT模型残差图 and 对比图.png
 -  Q4 (修改) .ipynb
 -  Q4.ipynb
 -  QQ图.png
 -  submit.csv
 -  test.csv
 -  train.csv
 -  xgboostmodel残差图 and 散点图.png
 -  xgboostmodel对比图.png
 -  XGBoost模型残差图 and 对比图.png
 -  洪水概率的频率分布直方图 and 折线图.png
-  决策树模型.png
 -  随机森林 (残差图 and 对比图) .png
 -  随机森林模型.png

附录 1

介绍：问题一热力图



附录 1								
介绍：问题 1 的描述性统计分析								
指标名	Count	Mean	Std	Min	25%	50%	75%	Max
季风强度	1000	5.2	1.5	1	4	5	6	8
地形排水	1000	5.3	1.5	1	4	5	6	8
河流管理	1000	4.6	1.6	1	4	4	6	8
森林砍伐	1000	5.6	1.7	1	4	6	7	8
城市化	1000	5	1.6	1	4	5	6	8
气候变化	1000	5.2	1.7	1	4	5	7	8
大坝质量	1000	3.4	1.5	1	2	3	5	8
淤积	1000	5.1	1.6	1	4	5	6	8
农业实践	1000	4.1	1.5	1	3	4	5	8
土壤渗透性	1000	5.1	1.6	1	4	5	6	8
降雨量	1000	4.9	1.7	1	4	5	6	8
排水系统	1000	4.2	1.7	1	3	4	6	8
海岸脆弱性	1000	3.1	1.5	1	2	3	4	8
滑坡	1000	4.3	2.5	0	3	5	6	8
流域	1000	4.8	1.5	1	4	5	6	8
基础设施恶化	1000	4.9	1.5	1	4	5	6	8
人口得分	1000	5	2.4	1	3	5	7	8
湿地损失	1000	3.6	1.5	1	2	3	5	8
规划不足	1000	4.5	1.7	1	3	4	6	8
政策因素	1000	4	1.6	1	3	4	5	8
洪水概率	1000	0.51	0.07	0.35	0.46	0.5	0.56	0.7

附录 1

介绍：问题一代码

```
1. import pandas as pd
2. data=pd.read_csv("train.csv",encoding='gbk')
3. data
4. data.columns
5. import pandas as pd
6. from sklearn.model_selection import train_test_split
7. import numpy as np
8. from scipy import stats
9. # 加载数据
10.
11. # 定义特征和目标变量
12. features = ['季风强度', '地形排水', '河流管理', '森林砍伐', '城市化', '气候变化', '大坝质量', '淤积', '农业实践', '侵蚀', '无效防灾', '排水系统', '海岸脆弱性', '滑坡', '流域', '基础设施恶化', '人口得分', '湿地损失', '规划不足', '政策因素']
13. x = data[features]
14. y = data['洪水概率']
15.
16.
17. # 定义函数来执行Shapiro-Wilk 检验
18. def shapiro_wilk_test(data, feature_name):
19.     statistic, p_value = stats.shapiro(data)
20.     alpha = 0.05
21.     result = "符合正态分布" if p_value > alpha else "不符合正态分布"
22.     return {
23.         'Feature': feature_name,
24.         'Shapiro-Wilk 检验统计量': round(statistic, 3),
25.         'p 值': f"{p_value:.3f}",
26.         '正态分布结果': result
27.     }
28.
29. # 计算每个特征的Shapiro-Wilk 检验
30. results = []
31. for feature in features:
32.     result = shapiro_wilk_test(X[feature], feature)
33.     results.append(result)
34.
35. # 转换结果为DataFrame 并显示
36. results_df = pd.DataFrame(results)
37. print(results_df)
38.
39. import pandas as pd
40.
```

```

41. # 读取数据, 注意根据你的实际文件路径进行调整
42. file_path = 'train.csv'
43. data = pd.read_csv(file_path, encoding='gbk')
44.
45. # 首行和首列互换
46. data_transposed = data.transpose()
47.
48. # 打印转换后的数据
49. data_transposed
50.
51. import matplotlib.pyplot as plt
52. import seaborn as sns
53. from matplotlib import font_manager as fm, rcParams
54. import warnings
55.
56. # Ignore warnings
57. warnings.filterwarnings("ignore")
58. # 添加并设置中文字体
59. font_path = 'D:/jupyterproject/FangZhengHeiTiJianTi-1.ttf'
60. font_prop = fm.FontProperties(fname=font_path)
61. plt.rcParams['font.sans-serif'] = [font_prop.get_name()] # 使用方正黑体简体
62. plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
63.
64. # 计算相关性矩阵
65. correlation = data[['洪水概率', '季风强度', '地形排水', '河流管理', '森林砍伐', '城市化', '气候
变化', '大坝质量', '淤积',
66.                    '农业实践', '侵蚀', '无效防灾', '排水系统', '海岸脆弱性', '滑坡', '流域', '基础设施恶化', '人口得分',
67.                    '湿地损失', '规划不足', '政策因素']].corr(method='spearman')
68.
69. # 设置风格
70. sns.set(style='white', font_scale=1.2)
71.
72. # 创建热力图, 使用浅色调的配色方案
73. plt.figure(figsize=(14, 12))
74. heatmap = sns.heatmap(
75.     correlation,
76.     cmap='coolwarm_r', # 使用浅色调的配色方案
77.     annot=True,
78.     fmt=".2f",
79.     linewidths=.5,
80.     cbar_kws={'shrink': .5, 'aspect': 10},
81.     annot_kws={'size': 10, 'weight': 'bold', 'color': 'black'},
82.     vmin=-1, vmax=1 # 调整颜色映射范围以使颜色更浅

```

```

83. )
84.
85. # 设置标题和标签
86. plt.title('指标间的相关性热力图
', fontsize=18, weight='bold', pad=20, fontproperties=font_prop)
87. plt.xticks(rotation=45, horizontalalignment='right', fontsize=12, weight='bold', fontpro
perties=font_prop)
88. plt.yticks(rotation=0, fontsize=12, weight='bold', fontproperties=font_prop)
89.
90. # 去除多余的图框线
91. heatmap.spines['top'].set_visible(False)
92. heatmap.spines['right'].set_visible(False)
93. heatmap.spines['left'].set_visible(False)
94. heatmap.spines['bottom'].set_visible(False)
95.
96. # 显示图形
97. plt.tight_layout()
98. # 保存图片
99. plt.savefig('correlation_heatmap.png', dpi=300)
100. plt.show()
101.
102.
103. correlation_matrix = correlation
104.
105.
106. flood_correlation = correlation_matrix['洪水概率'].sort_values()
107. flood_correlation
108. import matplotlib.pyplot as plt
109. import seaborn as sns
110. plt.rcParams['font.sans-serif']=['SimHei'] # 中文支持
111. plt.rcParams['axes.unicode_minus']=False
112. import warnings
113. warnings.filterwarnings('ignore')
114.
115. plt.figure(figsize=(12, 8))
116.
117. # Plotting the bar chart
118. bar_colors = sns.color_palette("coolwarm", len(flood_correlation.index))
119. sorted_indices = flood_correlation.head(-1).sort_values().index
120. sns.barplot(x=flood_correlation.head(-1).sort_values(), y=sorted_indices, palette=ba
r_colors)
121.
122. plt.title('各指标与洪水概率的相关性')
123. plt.xlabel('指标')

```

```
124.     plt.ylabel('相关系数')
125.
126.     # Show the plot
127.     plt.show()
128.
129.     import statsmodels.api as sm
130.
131.     # 准备数据, X 为自变量 (除去洪水概率的其它指标), y 为因变量 (洪水概率)
132.     x = data.drop(['洪水概率', 'id'], axis=1) # 移除'id'字段, 因为它是唯一标识, 不应影响模型
133.     y = data['洪水概率']
134.
135.     # 添加常数项, 因为线性回归模型需要截距
136.     # X = sm.add_constant(X)
137.
138.     # 创建OLS 模型
139.     model = sm.OLS(y, X)
140.     results = model.fit()
141.
142.     # 输出回归分析结果
143.     results.summary()
144.
145.     from sklearn.model_selection import train_test_split
146.     from sklearn.tree import DecisionTreeRegressor
147.     from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
148.     from xgboost import XGBRegressor
149.
150.
151.     # 训练模型
152.     # 重新初始化模型
153.     decision_treemodel = DecisionTreeRegressor(random_state=42)
154.     random_forestmodel = RandomForestRegressor(random_state=42)
155.     xgboostmodel = XGBRegressor(random_state=42)
156.     gbdtmodel = GradientBoostingRegressor(random_state=42)
157.
158.     # 使用全部数据训练模型
159.     decision_treemodel.fit(X, y)
160.     random_forestmodel.fit(X, y)
161.     xgboostmodel.fit(X, y)
162.     gbdtmodel.fit(X, y)
163.
164.
165.     from sklearn.metrics import r2_score, mean_absolute_percentage_error
166.
167.     # 预测训练集
```

```

168. y_pred_dt = decision_treemodel.predict(X)
169. y_pred_rf = random_forestmodel.predict(X)
170. y_pred_xgb = xgboostmodel.predict(X)
171. y_pred_gbdtd = gbdtdmodel.predict(X)
172.
173. # 计算R2
174. r2_dt = r2_score(y, y_pred_dt)
175. r2_rf = r2_score(y, y_pred_rf)
176. r2_xgb = r2_score(y, y_pred_xgb)
177. r2_gbdtd = r2_score(y, y_pred_gbdtd)
178.
179. # 计算MAPE
180. mape_dt = mean_absolute_percentage_error(y, y_pred_dt)
181. mape_rf = mean_absolute_percentage_error(y, y_pred_rf)
182. mape_xgb = mean_absolute_percentage_error(y, y_pred_xgb)
183. mape_gbdtd = mean_absolute_percentage_error(y, y_pred_gbdtd)
184.
185. # 输出结果
186. print("决策树 R^2:", r2_dt)
187. print("随机森林 R^2:", r2_rf)
188. print("XGBoost R^2:", r2_xgb)
189. print("GBDT R^2:", r2_gbdtd)
190. print("决策树 MAPE:", mape_dt)
191. print("随机森林 MAPE:", mape_rf)
192. print("XGBoost MAPE:", mape_xgb)
193. print("GBDT MAPE:", mape_gbdtd)
194.
195. import numpy as np
196. # 获取各个模型的特征重要性
197. feature_importances_dt = decision_treemodel.feature_importances_
198. feature_importances_rf = random_forestmodel.feature_importances_
199. feature_importances_xgb = xgboostmodel.feature_importances_
200. feature_importances_gbdtd = gbdtdmodel.feature_importances_
201. features=X.columns
202. # 输出决策树的特征重要性
203. importance_dt = pd.Series(feature_importances_dt, index=features).sort_values(ascending=False)
204.
205. # 输出随机森林的特征重要性
206. importance_rf = pd.Series(feature_importances_rf, index=features).sort_values(ascending=False)
207.
208. # 输出XGBoost 的特征重要性
209. importance_xgb = pd.Series(feature_importances_xgb, index=features).sort_values(ascending=False)

```

```

nding=False)
210.
211.     # 输出 GBDT 的特征重要性
212.     importance_gbdtdt = pd.Series(feature_importances_gbdtdt, index=features).sort_values(ascending=False)
213.
214.
215.     # 打印每个模型的特征重要性
216.     print("决策树特征重要性: ")
217.     display(importance_dt)
218.     print("\n 随机森林特征重要性: ")
219.     display(importance_rf)
220.     print("\nXGBoost 特征重要性: ")
221.     display(importance_xgb)
222.     print("\nGBDT 特征重要性: ")
223.     display(importance_gbdtdt)
224.     import pandas as pd
225.     import matplotlib.pyplot as plt
226.
227.     # 设置中文字体
228.     plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
229.
230.     # 特征重要性数据
231.     features = [
232.         "滑坡", "湿地损失", "海岸脆弱性", "森林砍伐", "农业实践", "规划不足", "淤积",
233.         "排水系统", "季风强度", "城市化", "大坝质量", "侵蚀", "政策因素", "无效防灾",
234.         "气候变化", "流域", "基础设施恶化", "人口得分", "河流管理", "地形排水"
235.     ]
236.
237.     decision_tree_importance = [
238.         0.051412, 0.051210, 0.050936, 0.050860, 0.050804, 0.050661, 0.050593,
239.         0.050556, 0.050515, 0.050495, 0.050400, 0.050130, 0.050098, 0.050001,
240.         0.049505, 0.049387, 0.049080, 0.048406, 0.048215, 0.046736
241.     ]
242.
243.     random_forest_importance = [
244.         0.050811, 0.050801, 0.050737, 0.050714, 0.050693, 0.050604, 0.050590,
245.         0.050454, 0.050361, 0.050329, 0.050224, 0.050113, 0.050064, 0.049900,
246.         0.049664, 0.049637, 0.049073, 0.048930, 0.048284, 0.048018
247.     ]
248.
249.     xgboost_importance = [
250.         0.056026, 0.055686, 0.055278, 0.054310, 0.052956, 0.052925, 0.052191,
251.         0.051726, 0.051283, 0.050458, 0.049932, 0.049285, 0.048824, 0.047779,

```

```

252.         0.046821, 0.046790, 0.046319, 0.045111, 0.043528, 0.042773
253.     ]
254.
255.     gbd_t_importance = [
256.         0.052912, 0.051795, 0.051500, 0.051257, 0.051163, 0.050925, 0.050704,
257.         0.050527, 0.050527, 0.050326, 0.049950, 0.049922, 0.049904, 0.049380,
258.         0.048915, 0.048735, 0.048582, 0.048042, 0.047762, 0.047172
259.     ]
260.
261.     # 创建数据框
262.     df = pd.DataFrame({
263.         'Feature': features,
264.         'Decision Tree': decision_tree_importance,
265.         'Random Forest': random_forest_importance,
266.         'XGBoost': xgboost_importance,
267.         'GBDT': gbd_t_importance
268.     })
269.
270.     # 设置图形大小
271.     plt.figure(figsize=(15, 10))
272.
273.     # 绘制折线图
274.     plt.plot(df['Feature'], df['Decision Tree'], marker='o', label='Decision Tree')
275.     plt.plot(df['Feature'], df['Random Forest'], marker='o', label='Random Forest')
276.     plt.plot(df['Feature'], df['XGBoost'], marker='o', label='XGBoost')
277.     plt.plot(df['Feature'], df['GBDT'], marker='o', label='GBDT')
278.
279.     # 添加图形标题和标签
280.     plt.title('Feature Importance by Model', fontsize=16)
281.     plt.xlabel('Features', fontsize=12)
282.     plt.ylabel('Importance', fontsize=12)
283.     plt.xticks(rotation=45, ha='right')
284.     plt.legend(title='Model')
285.
286.     # 显示图形
287.     plt.tight_layout()
288.
289.     # 保存图形到本地文件
290.     plt.savefig('feature_importance_by_model.png')
291.
292.     # 显示图形
293.     plt.show()
294.
295.     xgboostmodel = XGBRegressor(random_state=42)

```

```

296.     xgboostmodel.fit(X, y)
297.     import pandas as pd
298.     import matplotlib.pyplot as plt
299.     import seaborn as sns
300.
301.     # 设置中文字体
302.     plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
303.
304.     # SHAP 特征重要度数据
305.     shap_data = {
306.         "Feature": ["侵蚀", "大坝质量", "地形排水", "基础设施恶化", "季风强度", "城市化", "
滑坡", "规划不足", "政策因素",
307.                     "农业实践", "森林砍伐", "湿地损失", "淤积", "海岸脆弱性", "无效防灾", "
流域", "河流管理", "排水系统",
308.                     "气候变化", "人口得分"],
309.         "Importance": [0.009550, 0.009456, 0.009452, 0.009307, 0.009257, 0.009240, 0.009
218, 0.009218, 0.009212,
310.                        0.009178, 0.009161, 0.009145, 0.009131, 0.009102, 0.009097, 0.009
093, 0.009069, 0.009026,
311.                        0.008965, 0.008919]
312.     }
313.
314.     # 创建数据框
315.     df_shap = pd.DataFrame(shap_data)
316.
317.     # 设置图形大小
318.     plt.figure(figsize=(12, 8))
319.
320.     # 使用Seaborn 绘制带有特征重要度的条形图
321.     sns.barplot(x='Importance', y='Feature', data=df_shap, palette='viridis')
322.
323.     # 添加图形标题和标签
324.     plt.title('SHAP 特征重要度', fontsize=16)
325.     plt.xlabel('重要度', fontsize=12)
326.     plt.ylabel('特征', fontsize=12)
327.
328.     # 显示图形
329.     plt.tight_layout()
330.
331.     # 保存图形到本地文件
332.     plt.savefig('shap_feature_importance.png')
333.
334.     # 显示图形
335.     plt.show()

```

```

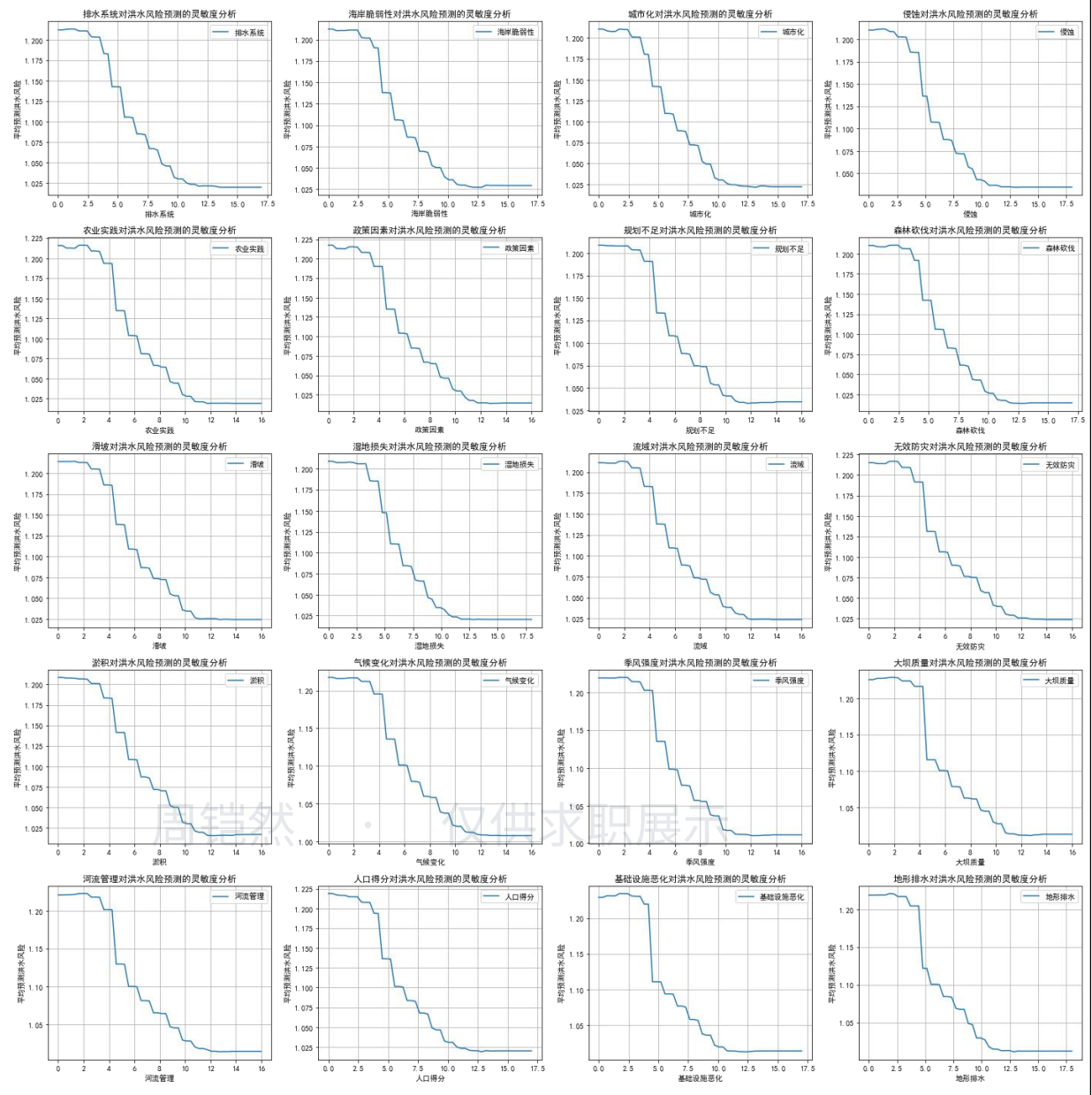
336.
337.     # 计算并打印 SHAP 特征重要度
338.     shap_importance = np.abs(shap_values.values).mean(axis=0)
339.     feature_importance = pd.DataFrame(list(zip(X.columns, shap_importance)), columns=['F
eature', 'Importance'])
340.     feature_importance = feature_importance.sort_values(by='Importance', ascending=False
)
341.
342.     print("SHAP 特征重要度:")
343.     feature_importance
344.     import pandas as pd
345.     import matplotlib.pyplot as plt
346.     import seaborn as sns
347.     from matplotlib import font_manager as fm
348.
349.     # 设置中文字体路径（需要根据你的系统调整路径）
350.     font_path = 'D:/jupyterproject/FangZhengHeiTiJianTi-1.ttf' # 这里需要替换为你系统中的
中文字体路径
351.     font_prop = fm.FontProperties(fname=font_path)
352.
353.     # SHAP 特征重要度数据
354.     data = {
355.         "Feature": ["侵蚀", "大坝质量", "地形排水", "基础设施恶化", "季风强度", "城市化", "
滑坡", "规划不足", "政策因素",
356.         "农业实践", "森林砍伐", "湿地损失", "淤积", "海岸脆弱性", "无效防灾", "
流域", "河流管理", "排水系统",
357.         "气候变化", "人口得分"],
358.         "Importance": [0.009550, 0.009456, 0.009452, 0.009307, 0.009257, 0.009240, 0.009
218, 0.009218, 0.009212,
359.         0.009178, 0.009161, 0.009145, 0.009131, 0.009102, 0.009097, 0.009
093, 0.009069, 0.009026,
360.         0.008965, 0.008919],
361.         "Normalized_Importance": [1.000000, 0.850297, 0.845129, 0.614165, 0.535520, 0.50
9303, 0.474257, 0.473693,
362.         0.464238, 0.410348, 0.383459, 0.357783, 0.336132, 0.29
0185, 0.282580, 0.275731,
363.         0.237076, 0.168757, 0.072031, 0.000000]
364.     }
365.
366.     # 创建数据框
367.     df_shap = pd.DataFrame(data)
368.
369.     # 按归一化重要度排序
370.     df_shap = df_shap.sort_values(by='Normalized_Importance', ascending=False)

```

```
371.
372.     # 设置Seaborn 的风格和调色板
373.     sns.set_theme(style="whitegrid")
374.     palette = sns.color_palette("viridis", len(df_shap))
375.
376.     # 绘制归一化特征重要度条形图
377.     plt.figure(figsize=(12, 8))
378.     bar_plot = sns.barplot(x='Normalized_Importance', y='Feature', data=df_shap, palette
=palette)
379.
380.     # 设置轴标签和标题的字体
381.     plt.xlabel('归一化重要度', fontproperties=font_prop)
382.     plt.ylabel('特征', fontproperties=font_prop)
383.     plt.title('归一化 SHAP 特征重要度', fontproperties=font_prop)
384.
385.     # 设置y 轴刻度标签的字体
386.     bar_plot.set_yticklabels(bar_plot.get_yticklabels(), fontproperties=font_prop)
387.
388.     # 反转 y 轴，让重要度高的特征显示在顶部
389.     plt.gca().invert_yaxis()
390.
391.     # 保存图形到本地文件
392.     plt.savefig('shap_feature_importance_new_style.png', bbox_inches='tight')
393.
394.     # 显示图形
395.     plt.tight_layout()
396.     plt.show()
397.
398.     feature_importance
```

附录 1

介绍：问题二的灵敏度分析总图



附录 1

介绍：问题二的全部代码

```
1. import pandas as pd
2. data=pd.read_csv("train.csv",encoding='gbk')
3. data
4. # import matplotlib.pyplot as plt
5. import seaborn as sns
6. import pandas as pd
7.
8. # 设置中文字体
9. plt.rcParams['font.sans-serif'] = ['SimHei'] # 中文支持
10. plt.rcParams['axes.unicode_minus'] = False
11.
12. # 创建图形
13. plt.figure(figsize=(12, 8))
14.
15. # 使用 Seaborn 绘制频率分布直方图和概率密度图
16. # 改变颜色为黑色，边框颜色为白色
17. sns.histplot(data['洪水概率'], bins=30, kde=True, color='black', edgecolor='white')
18.
19. # 添加图形标题和标签
20. plt.title('洪水概率的频率分布直方图和概率密度图', fontsize=16)
21. plt.xlabel('洪水概率', fontsize=12)
22. plt.ylabel('频率', fontsize=12)
23.
24. # 显示网格
25. plt.grid(True)
26. # 保存图形到本地文件
27. plt.savefig('洪水概率的频率分布直方图和概率密度图.png')
28. # 显示图形
29. plt.show()
30.
31. from sklearn.cluster import KMeans
32. from sklearn.metrics import silhouette_score, calinski_harabasz_score
33. import numpy as np
34.
35.
36. x = data['洪水概率'].values.reshape(-1, 1)
37. import warnings
38.
39. # Ignore warnings
40. warnings.filterwarnings("ignore")
41. # 设置中文字体
42. plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
```

```
43. # 计算不同簇数的平方和距离 (SSD)
44. ssd = []
45. for k in range(1, 11):
46.     kmeans = KMeans(n_clusters=k, random_state=0).fit(X)
47.     ssd.append(kmeans.inertia_)
48.
49. # 绘制肘部法则图
50. plt.figure(figsize=(10, 6))
51. plt.plot(range(1, 11), ssd, marker='o')
52. plt.title('肘部法则确定最优簇数')
53. plt.xlabel('簇的数量 (k)')
54. plt.ylabel('平方和距离 (SSD)')
55. plt.grid(True)
56. plt.show()
57.
58.
59. kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
60. labels = kmeans.labels_
61. # 假设 data DataFrame 和 labels 已经定义
62. data['风险类别'] = labels
63.
64. # 设置图形大小
65. plt.figure(figsize=(10, 6))
66.
67. # 绘制箱线图, 使用 patch_artist=True 来允许填充颜色
68. boxplot = data.boxplot(column='洪水概率', by='风险类别', grid=True, patch_artist=True)
69.
70. # 设置颜色
71. colors = ['lightblue', 'lightgreen', 'lightyellow', 'lightcoral']
72. for patch, color in zip(boxplot.artists, colors): # 使用 boxplot 的 artists 属性来访问箱子
73.     patch.set_facecolor(color)
74.
75. # 设置图的标题和坐标轴标签
76. plt.title('不同风险类别的洪水概率箱线图')
77. plt.xlabel('风险类别')
78. plt.ylabel('洪水概率')
79. plt.suptitle('') # 清除默认的副标题
80.
81. # 保存图片到当前目录, 文件名为 'flood_risk_boxplot.png'
82. plt.savefig('flood_risk_boxplot.png', bbox_inches='tight')
83.
84. # 显示图形
85. plt.show()
86. import matplotlib.pyplot as plt
```

```

87. import seaborn as sns
88. import pandas as pd
89.
90. # 假设你的数据集已经加载到data中，并且包含'洪水概率'这一列
91. # X = data[['洪水概率']]
92.
93. # 选择k=3进行K-means聚类
94. kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
95. data['cluster'] = kmeans.labels_
96.
97. # 分析不同类别的指标特征
98. features = ['季风强度', '地形排水', '河流管理', '森林砍伐', '城市化', '气候变化', '大坝质量',
               '淤积', '农业实践', '侵蚀', '无效防灾', '排水系统', '海岸脆弱性', '滑坡', '流域', '基础设施恶化',
               '人口得分', '湿地损失', '规划不足', '政策因素']
99.
100. # 计算每个特征在不同簇中的均值
101. cluster_means = data.groupby('cluster')[features].mean().T
102.
103. # 绘制折线图
104. plt.figure(figsize=(14, 8))
105. for cluster in cluster_means.columns:
106.     plt.plot(cluster_means.index, cluster_means[cluster], marker='o', label=f'风险类别 {cluster}')
107.
108. # 设置图形标题和标签
109. plt.title('不同风险类别的各特征均值', fontsize=16)
110. plt.xlabel('特征', fontsize=12)
111. plt.ylabel('均值', fontsize=12)
112. plt.xticks(rotation=45, ha='right')
113. plt.legend(title='风险类别')
114. plt.grid(True)
115. plt.tight_layout()
116.
117. # 保存图形到本地文件
118. plt.savefig('不同风险类别的各特征均值.png')
119.
120. plt.show()
121.
122. # 各个风险类别的描述性统计
123. descriptive_stats = data.groupby('风险类别')['洪水概率'].describe()
124. print(descriptive_stats)
125.
126. from sklearn.model_selection import train_test_split
127. from sklearn.tree import DecisionTreeClassifier

```

```

128.     from sklearn.metrics import classification_report
129.
130.     # 准备决策树模型的数据集
131.     feature_cols = data.columns[1:-2] # 排除 'id' 和最后两列 ('风险类别', '洪水概率')
132.     X = data[feature_cols]
133.     y = data['风险类别']
134.     train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.05, random_state=5)
135.     # 创建和训练决策树模型
136.     decision_tree = DecisionTreeClassifier(random_state=0)
137.     decision_tree.fit(train_X, train_y)
138.     train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=5)
139.
140.     # 进行预测并评估模型
141.     y_pred = decision_tree.predict(test_X)
142.     evaluation = classification_report(test_y, y_pred)
143.     print(evaluation)
144.
145.     from sklearn.metrics import confusion_matrix
146.     import matplotlib.pyplot as plt
147.
148.
149.     # 横坐标是真实类别数, 纵坐标是预测类别数
150.     cf_matrix = confusion_matrix(test_y, y_pred)
151.
152.
153.
154.     # 混淆矩阵
155.     ax = sns.heatmap(cf_matrix, annot=True, fmt='g', cmap='Blues')
156.     ax.title.set_text("混淆矩阵")
157.     ax.set_xlabel("预测值")
158.     ax.set_ylabel("真实值")
159.     plt.show()
160.
161.     # 获取特征重要性
162.     feature_importance = decision_tree.feature_importances_
163.     # 创建一个数据框以更清晰地查看特征重要性
164.     importance_df = pd.DataFrame({'特征': feature_cols, '重要性': feature_importance}).sort_values(by='重要性', ascending=False)
165.
166.     importance_df
167.     # 绘制特征重要度图表
168.     plt.figure(figsize=(10, 6))
169.     plt.barh(importance_df['特征'], importance_df['重要性'], color='skyblue')
170.     plt.xlabel('重要性')

```

```

171. plt.ylabel('特征')
172. plt.title('特征重要度')
173. plt.gca().invert_yaxis() # 反转Y轴, 使重要性最高的特征在最上面
174. plt.show()
175. from sklearn.tree import export_graphviz
176. import graphviz
177.
178. #库安装方法: https://blog.csdn.net/qq_45956730/article/details/126689318
179. # 导出决策树图表
180. dot_data = export_graphviz(decision_tree, out_file=None, feature_names=feature_cols,
filled=True, rounded=True, special_characters=True)
181. import sys
182. ![sys.executable] -m pip install pydotplus -i https://pypi.tuna.tsinghua.edu.cn/simple
183.
184. from pydotplus.graphviz import graph_from_dot_data
185.
186. from sklearn.tree import export_graphviz
187.
188. dot_data = export_graphviz(
189.
190. decision_tree, feature_names= feature_cols, filled=True, rounded=True,
191. out_file=None
192.
193. )
194. dot_data=dot_data.replace('helvetica','Microsoft Yahei') # 修改字体, 支持中文
195. graph = graph_from_dot_data(dot_data) # Create graph from dot data
196.
197. # 将图形对象保存为 PDF 文件
198. output_pdf_path = '决策树结构图.pdf' # 指定输出的 PDF 文件路径
199. graph.write_pdf(output_pdf_path)
200.
201.
202. # 定义特征变量
203. features_to_analyze = [
204.     '排水系统', '海岸脆弱性', '城市化', '侵蚀', '农业实践', '政策因素', '规划不足',
205.     '森林砍伐', '滑坡', '湿地损失', '流域', '无效防灾', '淤积', '气候变化',
206.     '季风强度', '大坝质量', '河流管理', '人口得分', '基础设施恶化', '地形排水'
207. ]
208. # 定义一个函数来进行单个特征的灵敏度分析
209. def sensitivity_analysis(model, X, feature, feature_range):
210.     mean_predictions = []
211.     for value in feature_range:
212.         X_temp = X.copy()

```

```

213.         X_temp[feature] = value
214.         predictions = model.predict(X_temp)
215.         mean_predictions.append(predictions.mean())
216.     return mean_predictions
217. # 定义一个函数来进行灵敏度分析
218. def sensitivity_analysis_all_features(model, X, features):
219.     sensitivities = {}
220.     for feature in features:
221.         feature_range = np.linspace(X[feature].min(), X[feature].max(), 50)
222.         sensitivity = sensitivity_analysis(model, X, feature, feature_range)
223.         sensitivities[feature] = sensitivity
224.     return sensitivities
225.
226. # 进行灵敏度分析
227. sensitivities = sensitivity_analysis_all_features(decision_tree, test_X, features_to
_analyze)
228.
229. # 绘制灵敏度分析结果
230. plt.figure(figsize=(20, 20))
231. for i, feature in enumerate(features_to_analyze, 1):
232.     plt.subplot(5, 4, i)
233.     feature_range = np.linspace(test_X[feature].min(), test_X[feature].max(), 50)
234.     plt.plot(feature_range, sensitivities[feature], label=feature)
235.     plt.title(f'{feature}对洪水风险预测的灵敏度分析')
236.     plt.xlabel(feature)
237.     plt.ylabel('平均预测洪水风险')
238.     plt.grid(True)
239.     plt.legend()
240.
241. plt.tight_layout()
242. plt.show()
243.
244. import seaborn as sns
245. import matplotlib.pyplot as plt
246. import numpy as np
247.
248. # 创建示例混淆矩阵
249. cm = np.array([[85886, 1470, 5427],
250.                [1335, 77026, 6125],
251.                [5146, 6152, 126006]])
252.
253. # 绘制混淆矩阵热图
254. plt.figure(figsize=(10, 7))
255. sns.heatmap(cm, annot=True, fmt='d', cmap='viridis') # 使用 'viridis' 颜色方案

```

```

256. plt.title('混淆矩阵')
257. plt.xlabel('预测值')
258. plt.ylabel('真实值')
259.
260. # 保存图像到文件
261. plt.savefig('confusion_matrix.png', dpi=300, bbox_inches='tight')
262. plt.show()
263.
264. import pandas as pd
265. import matplotlib.pyplot as plt
266. import seaborn as sns
267.
268. # 设置中文字体
269. plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
270. plt.rcParams['axes.unicode_minus'] = False # 解决坐标轴负号显示问题
271.
272. # SHAP 特征重要度数据
273. shap_data = {
274.     "Feature": ["排水系统", "城市化", "海岸脆弱性", "侵蚀", "淤积", "湿地损失", "滑坡",
275.                 "农业实践", "森林砍伐", "规划不足", "气候变化", "无效防灾", "人口得分",
276.                 "河流管理", "地形排水", "大坝质量", "基础设施恶化"],
277.     "Importance": [0.054544, 0.053760, 0.053418, 0.053005, 0.052322, 0.051996, 0.051921, 0.051799, 0.051608,
278.                    0.051340, 0.051176, 0.051103, 0.049833, 0.049377, 0.049181, 0.047538, 0.047529, 0.046314,
279.                    0.042449, 0.039785]
280. }
281.
282. # 创建数据框
283. df_shap = pd.DataFrame(shap_data)
284.
285. # 设置图形大小
286. plt.figure(figsize=(12, 8))
287.
288. # 使用 Seaborn 绘制带有特征重要度的条形图
289. sns.barplot(x='Importance', y='Feature', data=df_shap, palette='viridis')
290.
291. # 添加图形标题和标签
292. plt.title('特征重要度', fontsize=16)
293. plt.xlabel('重要度', fontsize=12)
294. plt.ylabel('特征', fontsize=12)
295.

```

```
296.     # 显示网格
297.     plt.grid(True, linestyle='--', alpha=0.6)
298.
299.     # 显示图形
300.     plt.tight_layout()
301.
302.     # 保存图形到本地文件
303.     plt.savefig('特征重要性.png')
304.
305.     # 显示图形
306.     plt.show()
```

周铠然 · 仅供求职展示

附录 1

介绍：问题三的所有代码

```
1. import pandas as pd
2. data=pd.read_csv("train.csv",encoding='gbk')
3. data
4. selected_features = [
5.     '政策因素',
6.     '气候变化',
7.     '滑坡',
8.     '基础设施恶化',
9.     '城市化',
10.    '大坝质量',
11.    '河流管理',
12.    '地形排水',
13.    '森林砍伐',
14.    '人口得分'
15. ]
16. X=data[selected_features]
17. y=data['洪水概率']
18. from sklearn.model_selection import train_test_split
19. from sklearn.tree import DecisionTreeRegressor
20. from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
21. from xgboost import XGBRegressor
22. # 将数据拆分为训练集和测试集
23. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
24. decision_treemodel = DecisionTreeRegressor(random_state=42)
25. # 使用全部数据训练模型
26. decision_treemodel.fit(X_train, y_train)
27. import matplotlib.pyplot as plt
28. import numpy as np
29. import warnings
30.
31. # Ignore warnings
32. warnings.filterwarnings("ignore")
33.
34. # 设置中文字体
35. plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
36.
37. # 模型名称
38. models = ['决策树', '随机森林', 'XGBoost', 'GBDT']
39.
40. # 各模型的评估结果
41. mse_values = [0.0035748116693104623, 0.0017208001550090691, 0.0016335250010614292, 0.0016776374722831846]
```

```

42. r2_values = [-0.3711101075741017, 0.3399913886643666, 0.37346555647717805, 0.35654632806
53812]
43. mape_values = [9.60, 6.74, 6.58, 6.69]
44.
45. # 设置图表风格
46. plt.style.use('ggplot')
47.
48. # 创建图形并设置大小
49. fig, ax1 = plt.subplots(figsize=(14, 8))
50.
51. # 绘制均方差 (MSE) 的柱状图
52. bar_width = 0.2
53. index = np.arange(len(models))
54.
55. bar1 = ax1.bar(index, mse_values, bar_width, label='MSE', color='b')
56.
57. # 创建第二个纵轴，共享X轴
58. ax2 = ax1.twinx()
59. bar2 = ax2.bar(index + bar_width, r2_values, bar_width, label='R^2 Score', color='r')
60. bar3 = ax2.bar(index + 2 * bar_width, mape_values, bar_width, label='MAPE', color='g')
61.
62. # 设置图表标题和轴标签
63. ax1.set_xlabel('模型')
64. ax1.set_ylabel('MSE', color='b')
65. ax2.set_ylabel('R^2 Score 和 MAPE', color='r')
66. plt.title('不同模型的评估结果比较')
67.
68. # 设置X轴刻度和标签
69. plt.xticks(index + bar_width, models)
70.
71. # 添加图例
72. fig.legend(loc='upper right', bbox_to_anchor=(0.9, 0.9))
73.
74. # 显示图表
75. plt.show()
76.
77. # 保存图表到文件
78. fig.savefig('model_evaluation_comparison.png', dpi=300, bbox_inches='tight')
79.
80. import matplotlib.pyplot as plt
81. import numpy as np
82.
83. # 模型名称
84. models = ['决策树', '随机森林', 'XGBoost', 'GBDT']

```

```

85.
86. # 各模型的评估结果
87. mse_values = [0.0035748116693104623, 0.0017208001550090691, 0.0016335250010614292, 0.001
6776374722831846]
88. r2_values = [-0.3711101075741017, 0.3399913886643666, 0.37346555647717805, 0.35654632806
53812]
89. mape_values = [9.60, 6.74, 6.58, 6.69]
90.
91. # 设置图表风格
92. plt.style.use('ggplot')
93.
94. # 创建图形并设置大小
95. fig, ax1 = plt.subplots(figsize=(14, 8))
96.
97. # 绘制均方误差 (MSE) 的柱状图
98. bar_width = 0.2
99. index = np.arange(len(models))
100.
101. bar1 = ax1.bar(index, mse_values, bar_width, label='均方误差 (MSE)', color='b')
102.
103. # 创建第二个纵轴, 共享 X 轴
104. ax2 = ax1.twinx()
105. bar2 = ax2.bar(index + bar_width, r2_values, bar_width, label='R² 得分', color='r')
106. bar3 = ax2.bar(index + 2 * bar_width, mape_values, bar_width, label='均绝对百分比误差 (MAPE)', color='g')
107.
108. # 设置图表标题和轴标签
109. ax1.set_xlabel('模型')
110. ax1.set_ylabel('均方误差 (MSE)', color='b')
111. ax2.set_ylabel('R² 得分 和 均绝对百分比误差 (MAPE)', color='r')
112. plt.title('不同模型的评估结果比较')
113.
114. # 设置 X 轴刻度和标签
115. plt.xticks(index + bar_width, models)
116.
117. # 添加图例
118. fig.legend(loc='upper right', bbox_to_anchor=(0.9, 0.9))
119.
120. # 显示图表
121. plt.show()
122.
123.
124. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_er
ror

```

```

125. import matplotlib.pyplot as plt
126. import seaborn as sns
127. plt.rcParams['font.sans-serif']=['SimHei'] # 中文支持
128. plt.rcParams['axes.unicode_minus']=False
129. # 预测值
130. y_pred = decision_treemodel.predict(X_test)
131.
132. # 计算评估指标
133. mse = mean_squared_error(y_test, y_pred)
134. r2 = r2_score(y_test, y_pred)
135. mape = mean_absolute_percentage_error(y_test, y_pred)
136. # 打印评估指标
137. print(f"均方误差 (Mean Squared Error): {mse}")
138. print(f"R^2 得分 (R^2 Score): {r2}")
139. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
140. # 绘制模型预测值与实际值的散点图
141. plt.figure(figsize=(10, 5))
142. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
143. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
144. plt.xlabel('实际值')
145. plt.ylabel('预测值')
146. plt.title('实际值与预测值的对比图')
147. plt.show()
148.
149. # 绘制残差图
150. residuals = y_test - y_pred
151. plt.figure(figsize=(10, 5))
152. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
153. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
154. plt.xlabel('预测值')
155. plt.ylabel('残差')
156. plt.title('残差图')
157. plt.show()
158.
159. random_forestmodel = RandomForestRegressor(random_state=42)
160. random_forestmodel.fit(X_train, y_train)
161. # 预测值
162. y_pred = random_forestmodel.predict(X_test)
163.
164. # 计算评估指标
165. mse = mean_squared_error(y_test, y_pred)
166. r2 = r2_score(y_test, y_pred)
167. mape = mean_absolute_percentage_error(y_test, y_pred)
168. # 打印评估指标

```

```

169.     print(f"均方误差 (Mean Squared Error): {mse}")
170.     print(f"R^2 得分 (R^2 Score): {r2}")
171.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
172.     # 绘制模型预测值与实际值的散点图
173.     plt.figure(figsize=(10, 5))
174.     plt.scatter(y_test, y_pred, alpha=0.7, color='b')
175.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
176.     plt.xlabel('实际值')
177.     plt.ylabel('预测值')
178.     plt.title('实际值与预测值的对比图')
179.     plt.show()
180.
181.     # 绘制残差图
182.     residuals = y_test - y_pred
183.     plt.figure(figsize=(10, 5))
184.     plt.scatter(y_pred, residuals, alpha=0.7, color='g')
185.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
186.     plt.xlabel('预测值')
187.     plt.ylabel('残差')
188.     plt.title('残差图')
189.     plt.show()
190.
191.     xgboostmodel = XGBRegressor(random_state=42)
192.     xgboostmodel.fit(X_train, y_train)
193.     # 预测值
194.     y_pred = xgboostmodel.predict(X_test)
195.
196.     # 计算评估指标
197.     mse = mean_squared_error(y_test, y_pred)
198.     r2 = r2_score(y_test, y_pred)
199.     mape = mean_absolute_percentage_error(y_test, y_pred)
200.     # 打印评估指标
201.     print(f"均方误差 (Mean Squared Error): {mse}")
202.     print(f"R^2 得分 (R^2 Score): {r2}")
203.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
204.     # 绘制模型预测值与实际值的散点图
205.     plt.figure(figsize=(10, 5))
206.     plt.scatter(y_test, y_pred, alpha=0.7, color='b')
207.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
208.     plt.xlabel('实际值')
209.     plt.ylabel('预测值')
210.     plt.title('实际值与预测值的对比图')
211.     plt.show()
212.

```

```

213. # 绘制残差图
214. residuals = y_test - y_pred
215. plt.figure(figsize=(10, 5))
216. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
217. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
218. plt.xlabel('预测值')
219. plt.ylabel('残差')
220. plt.title('残差图')
221. plt.show()
222.
223. gbdmodel = GradientBoostingRegressor(random_state=42)
224. gbdmodel.fit(X_train, y_train)
225. # 预测值
226. y_pred = gbdmodel.predict(X_test)
227.
228. # 计算评估指标
229. mse = mean_squared_error(y_test, y_pred)
230. r2 = r2_score(y_test, y_pred)
231. mape = mean_absolute_percentage_error(y_test, y_pred)
232. # 打印评估指标
233. print(f"均方误差 (Mean Squared Error): {mse}")
234. print(f"R^2 得分 (R^2 Score): {r2}")
235. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
236. # 绘制模型预测值与实际值的散点图
237. plt.figure(figsize=(10, 5))
238. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
239. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
240. plt.xlabel('实际值')
241. plt.ylabel('预测值')
242. plt.title('实际值与预测值的对比图')
243. plt.show()
244.
245. # 绘制残差图
246. residuals = y_test - y_pred
247. plt.figure(figsize=(10, 5))
248. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
249. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
250. plt.xlabel('预测值')
251. plt.ylabel('残差')
252. plt.title('残差图')
253. plt.show()

1. import pandas as pd
2. data=pd.read_csv("train.csv",encoding='gbk')

```

```
3. data
4. selected_features = [
5.     '侵蚀',
6.     '大坝质量',
7.     '地形排水',
8.     '基础设施恶化',
9.     '季风强度'
10.
11. ]
12. # 侵蚀
13. # 大坝质量
14. # 地形排水
15. # 基础设施恶化
16. # 季风强度
17. # 城市化
18. X=data[selected_features]
19. y=data['洪水概率']
20. from sklearn.model_selection import train_test_split
21. from sklearn.tree import DecisionTreeRegressor
22. from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
23. from xgboost import XGBRegressor
24. # 将数据拆分为训练集和测试集
25. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
26. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error
27. import matplotlib.pyplot as plt
28. import seaborn as sns
29. plt.rcParams['font.sans-serif']=['SimHei'] # 中文支持
30. plt.rcParams['axes.unicode_minus']=False
31. # 训练模型
32. decision_treemodel = DecisionTreeRegressor(random_state=42)
33. # 使用全部数据训练模型
34. decision_treemodel.fit(X_train, y_train)
35. import matplotlib.pyplot as plt
36. import numpy as np
37. import warnings
38.
39. # Ignore warnings
40. warnings.filterwarnings("ignore")
41.
42. # 设置中文字体
43. plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
44.
45. # 模型名称
46. models = ['决策树', '随机森林', 'XGBoost', 'GBDT']
```

```

47.
48. # 各模型的评估结果
49. mse_values = [0.0024874465795328917, 0.0023268288471887904, 0.00212860114409044, 0.00212
18171931355625]
50. r2_values = [0.04594606296951287, 0.10755059396152122, 0.18358033551480746, 0.1837032901
2603537]
51. mape_values = [8.02, 7.81, 7.51, 7.49]
52.
53. # 设置图表风格
54. plt.style.use('ggplot')
55.
56. # 创建图形并设置大小
57. fig, ax1 = plt.subplots(figsize=(14, 8))
58.
59. # 绘制均方误差 (MSE) 的柱状图
60. bar_width = 0.2
61. index = np.arange(len(models))
62.
63. bar1 = ax1.bar(index, mse_values, bar_width, label='MSE', color='b')
64.
65. # 创建第二个纵轴，共享X轴
66. ax2 = ax1.twinx()
67. bar2 = ax2.bar(index + bar_width, r2_values, bar_width, label='R^2 Score', color='r')
68. bar3 = ax2.bar(index + 2 * bar_width, mape_values, bar_width, label='MAPE', color='g')
69.
70. # 设置图表标题和轴标签
71. ax1.set_xlabel('模型')
72. ax1.set_ylabel('MSE', color='b')
73. ax2.set_ylabel('R^2 Score 和 MAPE', color='r')
74. plt.title('不同模型的评估结果比较')
75.
76. # 设置X轴刻度和标签
77. plt.xticks(index + bar_width, models)
78.
79. # 添加图例
80. fig.legend(loc='upper right', bbox_to_anchor=(0.9, 0.9))
81.
82. # 显示图表
83. plt.show()
84.
85. # 保存图表到文件
86. fig.savefig('model_evaluation_comparison_updated.png', dpi=300, bbox_inches='tight')
87.
88. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

```

```
89. import matplotlib.pyplot as plt
90. import seaborn as sns
91. plt.rcParams['font.sans-serif']=['SimHei'] # 中文支持
92. plt.rcParams['axes.unicode_minus']=False
93.
94. # 预测值
95. y_pred = decision_treemodel.predict(X_test)
96.
97. # 计算评估指标
98. mse = mean_squared_error(y_test, y_pred)
99. r2 = r2_score(y_test, y_pred)
100. mape = mean_absolute_percentage_error(y_test, y_pred)
101. # 打印评估指标
102. print(f"均方误差 (Mean Squared Error): {mse}")
103. print(f"R^2 得分 (R^2 Score): {r2}")
104. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
105. # 绘制模型预测值与实际值的散点图
106. plt.figure(figsize=(10, 5))
107. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
108. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
109. plt.xlabel('实际值')
110. plt.ylabel('预测值')
111. plt.title('实际值与预测值的对比图')
112. plt.show()
113.
114. # 绘制残差图
115. residuals = y_test - y_pred
116. plt.figure(figsize=(10, 5))
117. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
118. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
119. plt.xlabel('预测值')
120. plt.ylabel('残差')
121. plt.title('残差图')
122. plt.show()
123.
124. random_forestmodel = RandomForestRegressor(random_state=42)
125. random_forestmodel.fit(X_train, y_train)
126. # 预测值
127. y_pred = random_forestmodel.predict(X_test)
128.
129. # 计算评估指标
130. mse = mean_squared_error(y_test, y_pred)
131. r2 = r2_score(y_test, y_pred)
132. mape = mean_absolute_percentage_error(y_test, y_pred)
```

```

133. # 打印评估指标
134. print(f"均方误差 (Mean Squared Error): {mse}")
135. print(f"R^2 得分 (R^2 Score): {r2}")
136. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
137. # 绘制模型预测值与实际值的散点图
138. plt.figure(figsize=(10, 5))
139. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
140. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
141. plt.xlabel('实际值')
142. plt.ylabel('预测值')
143. plt.title('实际值与预测值的对比图')
144. plt.show()
145.
146. # 绘制残差图
147. residuals = y_test - y_pred
148. plt.figure(figsize=(10, 5))
149. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
150. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
151. plt.xlabel('预测值')
152. plt.ylabel('残差')
153. plt.title('残差图')
154. plt.show()
155.
156. xgboostmodel = XGBRegressor(random_state=42)
157. xgboostmodel.fit(X_train, y_train)
158. # 预测值
159. y_pred = xgboostmodel.predict(X_test)
160.
161. # 计算评估指标
162. mse = mean_squared_error(y_test, y_pred)
163. r2 = r2_score(y_test, y_pred)
164. mape = mean_absolute_percentage_error(y_test, y_pred)
165. # 打印评估指标
166. print(f"均方误差 (Mean Squared Error): {mse}")
167. print(f"R^2 得分 (R^2 Score): {r2}")
168. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
169. # 绘制模型预测值与实际值的散点图
170. plt.figure(figsize=(10, 5))
171. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
172. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
173. plt.xlabel('实际值')
174. plt.ylabel('预测值')
175. plt.title('实际值与预测值的对比图')
176. plt.show()

```

```

177.
178.     # 绘制残差图
179.     residuals = y_test - y_pred
180.     plt.figure(figsize=(10, 5))
181.     plt.scatter(y_pred, residuals, alpha=0.7, color='g')
182.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
183.     plt.xlabel('预测值')
184.     plt.ylabel('残差')
185.     plt.title('残差图')
186.     plt.show()
187.
188.     gbdtmodel = GradientBoostingRegressor(random_state=42)
189.     gbdtmodel.fit(X_train, y_train)
190.     # 预测值
191.     y_pred = gbdtmodel.predict(X_test)
192.
193.     # 计算评估指标
194.     mse = mean_squared_error(y_test, y_pred)
195.     r2 = r2_score(y_test, y_pred)
196.     mape = mean_absolute_percentage_error(y_test, y_pred)
197.     # 打印评估指标
198.     print(f"均方误差 (Mean Squared Error): {mse}")
199.     print(f"R^2 得分 (R^2 Score): {r2}")
200.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
201.     # 绘制模型预测值与实际值的散点图
202.     plt.figure(figsize=(10, 5))
203.     plt.scatter(y_test, y_pred, alpha=0.7, color='b')
204.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
205.     plt.xlabel('实际值')
206.     plt.ylabel('预测值')
207.     plt.title('实际值与预测值的对比图')
208.     plt.show()
209.
210.     # 绘制残差图
211.     residuals = y_test - y_pred
212.     plt.figure(figsize=(10, 5))
213.     plt.scatter(y_pred, residuals, alpha=0.7, color='g')
214.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
215.     plt.xlabel('预测值')
216.     plt.ylabel('残差')
217.     plt.title('残差图')
218.     plt.show()
219.
220.     from hyperopt import fmin, tpe, hp, STATUS_OK, Trials

```

```

221. import xgboost as xgb
222. from sklearn.metrics import mean_absolute_percentage_error
223. from sklearn.model_selection import train_test_split
224.
225. # 分割数据集
226. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
227.
228. # 记录每次评估的结果
229. a = []
230. b = []
231. best_mape = float('inf')
232.
233. def hyperopt_train_test(params):
234.     global best_mape
235.     try:
236.         # 创建并训练模型
237.         model = xgb.XGBRegressor(**params, random_state=42)
238.         model.fit(X_train, y_train)
239.
240.         # 预测并计算MAPE
241.         y_pred = model.predict(X_test)
242.         mape = mean_absolute_percentage_error(y_test, y_pred)
243.
244.         a.append(mape)
245.         b.append(params)
246.
247.         if mape < best_mape:
248.             best_mape = mape
249.
250.         return mape
251.
252.     except:
253.         return best_mape
254.
255. # 定义搜索空间
256. space = {
257.     'n_estimators': hp.choice('n_estimators', range(50, 500)),
258.     'max_depth': hp.choice('max_depth', range(1, 20)),
259.     'learning_rate': hp.uniform('learning_rate', 0.01, 0.3),
260.     'subsample': hp.uniform('subsample', 0.6, 1.0),
261.     'colsample_bytree': hp.uniform('colsample_bytree', 0.6, 1.0)
262. }
263.

```

```

264.     def f(params):
265.         mape = hyperopt_train_test(params)
266.         return {'loss': mape, 'status': STATUS_OK}
267.
268.     trials = Trials()
269.     best = fmin(f, space, algo=tpe.suggest, max_evals=100, trials=trials)
270.
271.     print("Best hyperparameters:", best)
272.     print("Best MAPE:", best_mape)
273.
274.     best
275.     xgboostmodel = XGBRegressor(**best)
276.     xgboostmodel.fit(X_train, y_train)
277.     # 预测值
278.     y_pred = xgboostmodel.predict(X_test)
279.
280.     # 计算评估指标
281.     mse = mean_squared_error(y_test, y_pred)
282.     r2 = r2_score(y_test, y_pred)
283.     mape = mean_absolute_percentage_error(y_test, y_pred)
284.     # 打印评估指标
285.     print(f"均方误差 (Mean Squared Error): {mse}")
286.     print(f"R^2 得分 (R^2 Score): {r2}")
287.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
288.     # 绘制模型预测值与实际值的散点图
289.     plt.figure(figsize=(10, 5))
290.     plt.scatter(y_test, y_pred, alpha=0.7, color='b')
291.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
292.     plt.xlabel('实际值')
293.     plt.ylabel('预测值')
294.     plt.title('实际值与预测值的对比图')
295.     plt.show()
296.
297.     # 绘制残差图
298.     residuals = y_test - y_pred
299.     plt.figure(figsize=(10, 5))
300.     plt.scatter(y_pred, residuals, alpha=0.7, color='g')
301.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
302.     plt.xlabel('预测值')
303.     plt.ylabel('残差')
304.     plt.title('残差图')
305.     plt.show()
306.
307.     from sklearn.ensemble import StackingRegressor

```

```

308.     from sklearn.linear_model import LinearRegression
309.     # 定义堆叠集成模型
310.     stacking_model = StackingRegressor([
311.         ('decision_tree', decision_treemodel),
312.         ('gbdt', gbdtmodel),
313.         ('xgboost', xgboostmodel)
314.     ], final_estimator=LinearRegression())
315.
316.     # 使用全部数据训练堆叠集成模型
317.     stacking_model.fit(X_train, y_train)
318.
319.     # 进行预测
320.     y_pred_stacking = stacking_model.predict(X_test)
321.
322.
323.     # 预测值
324.     y_pred = y_pred_stacking
325.
326.     # 计算评估指标
327.     mse = mean_squared_error(y_test, y_pred)
328.     r2 = r2_score(y_test, y_pred)
329.     mape = mean_absolute_percentage_error(y_test, y_pred)
330.     # 打印评估指标
331.     print(f"均方误差 (Mean Squared Error): {mse}")
332.     print(f"R^2 得分 (R^2 Score): {r2}")
333.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
334.     # 绘制模型预测值与实际值的散点图
335.     plt.figure(figsize=(10, 5))
336.     plt.scatter(y_test, y_pred, alpha=0.7, color='b')
337.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
338.     plt.xlabel('实际值')
339.     plt.ylabel('预测值')
340.     plt.title('实际值与预测值的对比图')
341.     plt.show()
342.
343.     # 绘制残差图
344.     residuals = y_test - y_pred
345.     plt.figure(figsize=(10, 5))
346.     plt.scatter(y_pred, residuals, alpha=0.7, color='g')
347.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
348.     plt.xlabel('预测值')
349.     plt.ylabel('残差')
350.     plt.title('残差图')
351.     plt.show(

```

附录 1

介绍：问题四的代码

```
1. import pandas as pd
2. data=pd.read_csv("train.csv",encoding='gbk')
3. data
4. data.columns
5. selected_features = [
6.     '季风强度', '地形排水', '河流管理', '森林砍伐', '城市化', '气候变化', '大坝质量', '淤积',
7.     '农业实践', '侵蚀', '无效防灾', '排水系统', '海岸脆弱性', '滑坡', '流域', '基础设施恶化',
8.     '湿地损失', '规划不足', '政策因素'
9. ]
10. X=data[selected_features]
11. y=data['洪水概率']
12. from sklearn.model_selection import train_test_split
13. from sklearn.tree import DecisionTreeRegressor
14. from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
15. from xgboost import XGBRegressor
16. # 将数据拆分为训练集和测试集
17. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
18. # 训练模型
19. # 重新初始化模型
20. decision_treemodel = DecisionTreeRegressor(random_state=42)
21. # 使用全部数据训练模型
22. decision_treemodel.fit(X_train, y_train)
23. import matplotlib.pyplot as plt
24. import seaborn as sns
25. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error
26.
27. # 中文支持
28. plt.rcParams['font.sans-serif'] = ['SimHei']
29. plt.rcParams['axes.unicode_minus'] = False
30.
31. # 预测值
32. y_pred = decision_treemodel.predict(X_test)
33.
34. # 计算评估指标
35. mse = mean_squared_error(y_test, y_pred)
36. r2 = r2_score(y_test, y_pred)
37. mape = mean_absolute_percentage_error(y_test, y_pred)
38.
39. # 打印评估指标
40. print(f"均方误差 (Mean Squared Error): {mse}")
41. print(f"R^2 得分 (R^2 Score): {r2}")
```

```

42. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
43.
44. # 绘制密度图
45. plt.figure(figsize=(10, 5))
46. sns.kdeplot(y_test, label='实际值', fill=True, color='blue', alpha=0.5)
47. sns.kdeplot(y_pred, label='预测值', fill=True, color='red', alpha=0.5)
48. plt.xlabel('值')
49. plt.ylabel('密度')
50. plt.title('预测值与实际值的密度图')
51. plt.legend()
52. plt.show()
53.
54. # 绘制残差分布图
55. residuals = y_test - y_pred
56. plt.figure(figsize=(10, 5))
57. sns.histplot(residuals, kde=True, color='green', bins=30)
58. plt.axvline(x=0, color='red', linestyle='--', linewidth=2)
59. plt.xlabel('残差')
60. plt.ylabel('频率')
61. plt.title('残差分布图')
62. plt.show()
63.
64. random_forestmodel = RandomForestRegressor(random_state=42)
65. random_forestmodel.fit(X_train, y_train)
66. # 预测值
67. y_pred = random_forestmodel.predict(X_test)
68.
69. # 计算评估指标
70. mse = mean_squared_error(y_test, y_pred)
71. r2 = r2_score(y_test, y_pred)
72. mape = mean_absolute_percentage_error(y_test, y_pred)
73. # 打印评估指标
74. print(f"均方误差 (Mean Squared Error): {mse}")
75. print(f"R^2 得分 (R^2 Score): {r2}")
76. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
77. # 绘制模型预测值与实际值的散点图
78. plt.figure(figsize=(10, 5))
79. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
80. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
81. plt.xlabel('实际值')
82. plt.ylabel('预测值')
83. plt.title('实际值与预测值的对比图')
84. plt.show()
85.

```

```

86. # 绘制残差图
87. residuals = y_test - y_pred
88. plt.figure(figsize=(10, 5))
89. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
90. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
91. plt.xlabel('预测值')
92. plt.ylabel('残差')
93. plt.title('残差图')
94. plt.show()
95.
96. import matplotlib.pyplot as plt
97. import seaborn as sns
98. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error
99. from sklearn.ensemble import RandomForestRegressor
100.
101. # 设置中文支持
102. plt.rcParams['font.sans-serif'] = ['SimHei']
103. plt.rcParams['axes.unicode_minus'] = False
104.
105. # 创建并训练随机森林模型
106. random_forestmodel = RandomForestRegressor(random_state=42)
107. random_forestmodel.fit(X_train, y_train)
108.
109. # 预测值
110. y_pred = random_forestmodel.predict(X_test)
111.
112. # 计算评估指标
113. mse = mean_squared_error(y_test, y_pred)
114. r2 = r2_score(y_test, y_pred)
115. mape = mean_absolute_percentage_error(y_test, y_pred)
116.
117. # 打印评估指标
118. print(f"均方误差 (Mean Squared Error): {mse}")
119. print(f"R^2 得分 (R^2 Score): {r2}")
120. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
121.
122. # 使用 seaborn 设置图表风格
123. sns.set(style="whitegrid")
124.
125. # 绘制模型预测值与实际值的散点图
126. plt.figure(figsize=(10, 5))
127. plt.scatter(y_test, y_pred, alpha=0.7, color='darkorange', edgecolor='w', s=100)
128. plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--', color='navy',
linewidth=2)

```

```

129. plt.xlabel('实际值')
130. plt.ylabel('预测值')
131. plt.title('实际值与预测值的对比图')
132. plt.show()
133.
134. # 绘制残差图
135. residuals = y_test - y_pred
136. plt.figure(figsize=(10, 5))
137. plt.scatter(y_pred, residuals, alpha=0.7, color='darkcyan', edgecolor='w', s=100)
138. plt.axhline(y=0, color='crimson', linestyle='--', linewidth=2)
139. plt.xlabel('预测值')
140. plt.ylabel('残差')
141. plt.title('残差图')
142. plt.show()
143.
144. xgboostmodel = XGBRegressor(random_state=42)
145. xgboostmodel.fit(X_train, y_train)
146. # 预测值
147. y_pred = xgboostmodel.predict(X_test)
148.
149. # 计算评估指标
150. mse = mean_squared_error(y_test, y_pred)
151. r2 = r2_score(y_test, y_pred)
152. mape = mean_absolute_percentage_error(y_test, y_pred)
153. # 打印评估指标
154. print(f"均方误差 (Mean Squared Error): {mse}")
155. print(f"R^2 得分 (R^2 Score): {r2}")
156. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
157. # 绘制模型预测值与实际值的散点图
158. plt.figure(figsize=(10, 5))
159. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
160. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
161. plt.xlabel('实际值')
162. plt.ylabel('预测值')
163. plt.title('实际值与预测值的对比图')
164. plt.show()
165.
166. # 绘制残差图
167. residuals = y_test - y_pred
168. plt.figure(figsize=(10, 5))
169. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
170. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
171. plt.xlabel('预测值')
172. plt.ylabel('残差')

```

```
173. plt.title('残差图')
174. plt.show()
175.
176. gbdtdmodel = GradientBoostingRegressor(random_state=42)
177. gbdtdmodel.fit(X_train, y_train)
178. # 预测值
179. y_pred = gbdtdmodel.predict(X_test)
180. #
181. # 计算评估指标
182. mse = mean_squared_error(y_test, y_pred)
183. r2 = r2_score(y_test, y_pred)
184. mape = mean_absolute_percentage_error(y_test, y_pred)
185. # 打印评估指标
186. print(f"均方误差 (Mean Squared Error): {mse}")
187. print(f"R^2 得分 (R^2 Score): {r2}")
188. print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
189. # 绘制模型预测值与实际值的散点图
190. plt.figure(figsize=(10, 5))
191. plt.scatter(y_test, y_pred, alpha=0.7, color='b')
192. plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
193. plt.xlabel('实际值')
194. plt.ylabel('预测值')
195. plt.title('实际值与预测值的对比图')
196. plt.show()
197.
198. # 绘制残差图
199. residuals = y_test - y_pred
200. plt.figure(figsize=(10, 5))
201. plt.scatter(y_pred, residuals, alpha=0.7, color='g')
202. plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
203. plt.xlabel('预测值')
204. plt.ylabel('残差')
205. plt.title('残差图')
206. plt.show()
207.
208. # 设置中文字体
209. plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
210. import warnings
211.
212. # Ignore warnings
213. warnings.filterwarnings("ignore")
214.
215. xgboostmodel = XGBRegressor(random_state=42)
216. xgboostmodel.fit(X, y)
```

```

217.     # 预测值
218.     y_pred = xgboostmodel.predict(X)
219.
220.     # 计算评估指标
221.     mse = mean_squared_error(y, y_pred)
222.     r2 = r2_score(y, y_pred)
223.     mape = mean_absolute_percentage_error(y, y_pred)
224.     # 打印评估指标
225.     print(f"均方误差 (Mean Squared Error): {mse}")
226.     print(f"R^2 得分 (R^2 Score): {r2}")
227.     print(f"均绝对百分比误差 (Mean Absolute Percentage Error): {mape * 100:.2f}%")
228.     # 绘制模型预测值与实际值的散点图
229.     plt.figure(figsize=(10, 5))
230.     plt.scatter(y, y_pred, alpha=0.7, color='b')
231.     plt.plot([min(y), max(y)], [min(y), max(y)], '--', color='r', linewidth=2)
232.     plt.xlabel('实际值')
233.     plt.ylabel('预测值')
234.     plt.title('实际值与预测值的对比图')
235.     plt.show()
236.
237.     # 绘制残差图
238.     residuals = y - y_pred
239.     plt.figure(figsize=(10, 5))
240.     plt.scatter(y, residuals, alpha=0.7, color='g')
241.     plt.axhline(y=0, color='r', linestyle='--', linewidth=2)
242.     plt.xlabel('预测值')
243.     plt.ylabel('残差')
244.     plt.title('残差图')
245.     plt.show()
246.
247.     preX=pd.read_csv("test.csv",encoding='gbk')
248.     preX
249.     preX['洪水概率']=xgboostmodel.predict(preX[selected_features])
250.     preX[['id','洪水概率']].to_csv('submit.csv',index=None)
251.     import pandas as pd
252.     import matplotlib.pyplot as plt
253.     # 设置中文字体
254.     plt.rcParams['font.family'] = ['SimHei'] # 设置中文字体为黑体
255.     # 选择需要分析的列
256.     flood_probabilities = preX['洪水概率']
257.
258.     # 绘制频率分布直方图
259.     plt.figure(figsize=(10, 6))
260.     n, bins, patches = plt.hist(flood_probabilities, bins=30, color='white', edgecolor='

```

```
black', alpha=1)
261.
262.     # 绘制折线图
263.     bin_centers = 0.5 * (bins[:-1] + bins[1:])
264.     plt.plot(bin_centers, n, color='black', marker='o', linestyle='-', linewidth=1.5)
265.
266.     # 添加图表标题和轴标签
267.     plt.title('洪水概率的频率分布直方图和折线图')
268.     plt.xlabel('洪水概率')
269.     plt.ylabel('频率')
270.     plt.grid(True)
271.
272.     # 显示图表
273.     plt.show()
274.
275.     # 使用Kolmogorov-Smirnov 检验
276.     ks_stat, p_value = stats.kstest(flood_probabilities, 'norm', args=(flood_probabilities.mean(), flood_probabilities.std()))
277.     ks_stat, p_value
278.     # 进行Shapiro-Wilk 正态性检验
279.     shapiro_stat, shapiro_p_value = stats.shapiro(flood_probabilities)
280.
281.     # 输出检验结果
282.     shapiro_stat, shapiro_p_value
283.     import scipy.stats as stats
284.     import matplotlib.pyplot as plt
285.
286.     # 绘制Q-Q 图
287.     stats.probplot(flood_probabilities, dist="norm", plot=plt)
288.     plt.title('Q-Q Plot')
289.     plt.show()
```