选题	2024 年第十四届 APMCM	参赛编号
B	亚太地区大学生数学建模竞赛（中文赛项）	apmcm24102155

归去来“汐”：基于统计检验和 trans-CNN 的洪水概率预测

摘要

洪水灾害是全球范围内严重的自然灾害之一，对人类生命财产安全造成巨大威胁。本文旨在利用数据分析技术，建立洪水概率预测模型，并进行风险等级评估，以期为洪水风险管理提供科学依据。

针对问题一，本文首先对数据进行预处理，包括删除异常值、补全缺失值、数据标准化和归一化，以提高数据质量。其次，本文使用 **Jarque-Bera 检验**与 **Kolmogorov-Smirnov 检验**，得出数据分布特性，在 Pearson、Cramér's V、Kendall、Spearman 四种相关检验方法中择优选用。然后，通过 **Spearman** 秩相关系数法分析 20 个指标与洪水发生概率的相关性，识别出地形排水、侵蚀、人口密度、湿地损失和基础设施恶化等关键影响因素，并提出相应的预防措施。

针对问题二，本文使用 K 均值**聚类**将洪水发生概率划分为高、中、低三个风险等级，基于 **Mann-Whitney U 检验**提取合适指标，并构建 **AHP-CRITIC-TOPSIS** 主客观综合赋权预警评价模型，综合考虑多个指标的影响，对洪水发生风险进行评估。结合**主成分分析 (PCA)** 方法，本文对依次对聚类和评价模型的**灵敏度 (Sensitivity)** 分析和**鲁棒性 (Robustness)** 检验，验证了模型的可靠性和稳定性。

针对问题三，本文使用 **LightGBM** 算法建立洪水发生概率预测模型，按 7:3 划分训练集与测试集，提取**交叉验证集**并进行特征重要性分析。先试用了 **20** 个特征指标的**全特征**预测，最终保留 **5** 个关键指标进行预测进行**降维特征**预测。较之全特征预测，降维预测的准确度与可靠性相对较低。为了进一步提升模型性能，本文构建了 **Transformer-CNN** 融合深度学习模型，有效捕捉时序数据的局部和全局特征，大幅度优化了 **MSE**、**RMSE**、**MAE**、**MAPE**、R^2 的测试结果，提高了预测准确性。

针对问题四，本文使用改进后的模型对测试数据集进行洪水发生概率预测，并绘制了预测结果的直方图和折线图，分析其分布特征，并使用 3-σ 准则进行正态性检验，计算结果为总数据点 745305，异常点 1935，异常点占比为 0.0025962525，少于 0.3%。因此，**数据服从正态分布**。

本文建立的洪水概率预测模型和预警评价模型能够有效识别关键影响因素，评估洪水风险，并预测洪水发生概率，为洪水风险管理提供科学依据，有助于降低洪水灾害风险，保障人民生命财产安全。

关键字： Jarque-Bera 检验，Kolmogorov-Smirnov 检验，AHP-CRITIC-TOPSIS，Mann-Whitney U 检验，LightGBM，改良卷积神经网络 (Transformer-CNN)

目录

一、 问题重述

1.1 问题背景

洪水灾害，作为自然界最具破坏力的自然灾害之一，每年造成巨大的经济损失和人员伤亡。其发生频率和严重程度正随着全球气候变化和人口增长而不断上升，对人类社会构成严重威胁。

传统洪水风险管理主要依赖经验和直觉，缺乏科学性和精确性。并且难以准确预测洪水发生的概率和影响范围，难以制定有效的预防措施。随着大数据和人工智能技术的发展，可以利用海量洪水数据进行分析和建模，从而实现更精准的风险预测和评估。基于数据分析的洪水风险管理方法可以为制定更有效的预防措施提供科学依据，降低洪水灾害风险，保障人民生命财产安全。

通过分析洪水数据，识别与洪水发生密切相关的指标，例如地形、气象、人为因素等。建立洪水发生概率预测模型，并进行风险等级评估。因此，基于如何数据分析手段制定针对性的预防措施，例如加强基础设施建设、优化土地利用规划等，以降低洪水灾害风险，保障人民生命财产安全，成为当下热门的研究问题之一。

1.2 问题分析

- **问题一：**题目要求我们：分析 20 个指标与洪水发生概率的相关性，识别关键影响因素，并提出预防措施。这实际上要求我们：先对数据的分布特性做出分析，然后依据数据的分布特性进行一定的预处理，选择符合其分布特性的恰切的相关性分析方法，进而评估指标与洪水发生概率的相关性。
- **问题二：**题目要求我们：将洪水发生概率划分为 {高, 中, 低} 三大风险等级，并建立洪水概率的预警评价模型。这实际上要求我们：先使用聚类算法（例如 K-means、DBSCAN 密度聚类等）将洪水事件划分为高、中、低三大风险等级，然后选择合适的指标，并计算其权重，最后进行模型灵敏度分析，评估模型的可靠性、稳定性及泛化能力。
- **问题三：**题目要求我们：在问题 1 的基础上，建立洪水发生概率预测模型，并进行模型评估。这实际上要求我们：先使用统计方法或机器学习方法建立可靠的预测模型，并使用交叉验证等方法评估模型的准确性。然后，在此基础上，进一步缩减到仅使用 5 个关键指标，实现对洪水概率预测的有效性与可靠性。
- **问题四：**题目要求我们：在问题 2、3 的基础上，使用预测模型预测测试数据集中洪水发生概率，提交分析结果并将其可视化。这实际上要求我们：综合前三问的求解成果，依据问题 1 得到的相关性，问题 2 得到的聚类结果，对问题 3 得到的模型进行深度优化，使模型具有较好的预测可靠性。

1.3 我们的工作

我们将解决这四个问题的工作思路图绘制如Figure 1：

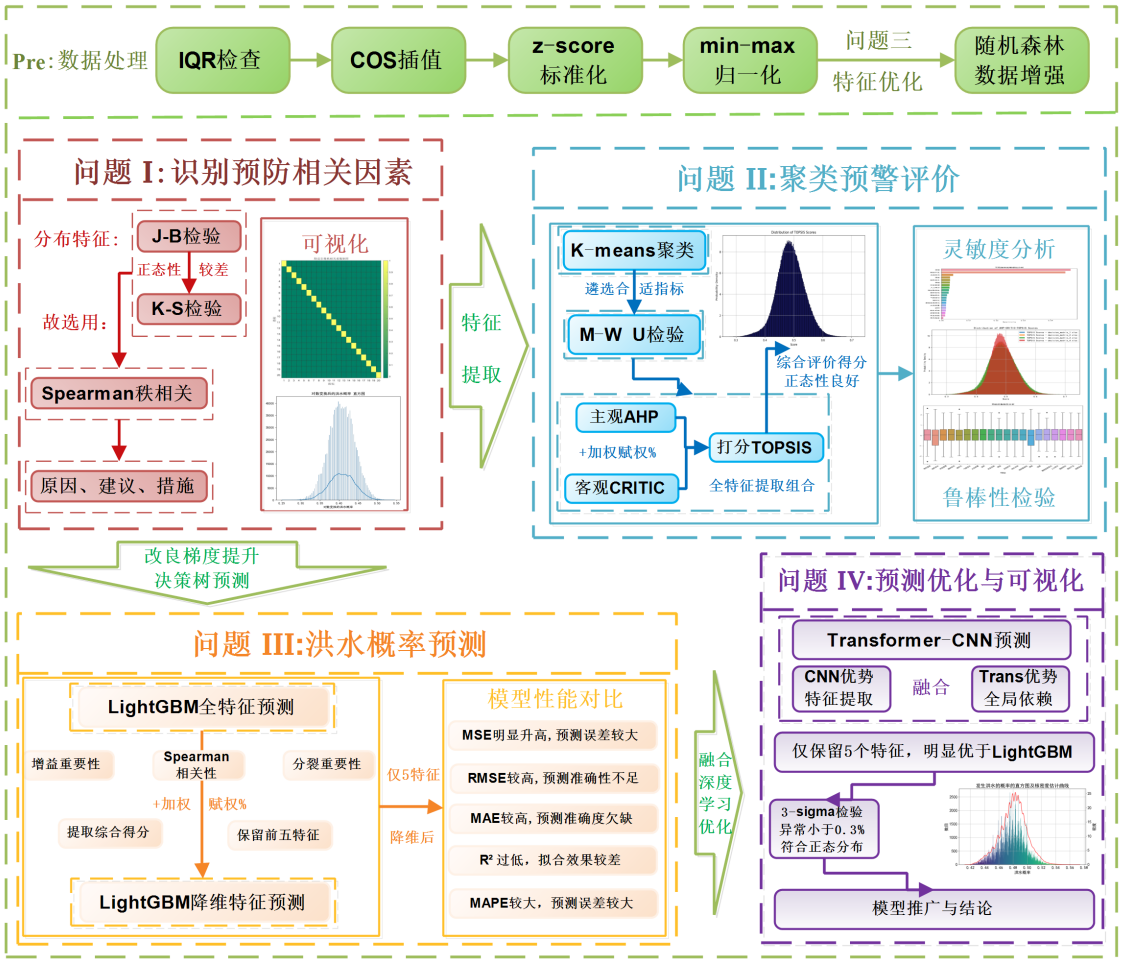

图 1　工作思路图

二、 模型假设

1. **数据完整性假设：**
   - 假设经过严格清洗和预处理后的训练数据（train.csv）和测试数据（test.csv），不存在缺失值、异常值等问题，数据质量良好，能够满足模型训练和预测的需求。
   - 假设所有数据点的采集时间和方法一致，且数据代表性强，能够反映真实的洪水灾害情况。
2. **独立同分布假设：**
   - 假设训练数据和测试数据中的洪水事件是相互独立的，即某个事件的发生不会影响其他事件的发生。

- 假设洪水事件的特征（如季风强度、地形排水等）是相互独立的，不存在显著的多重共线性。

3. **时间稳定性假设：**
   - 假设历史数据中所反映的洪水发生规律在未来的一段时间内依然成立，即数据中的特征与洪水发生概率之间的关系在预测期内保持不变。
4. **影响因素假设：**
   - 假设提供的 20 个特征指标能够较为全面地反映导致洪水发生的主要因素，这些指标是洪水预测的主要依据。
   - 假设未列入的其他潜在影响因素（如突发自然灾害、政策变动等）对模型预测结果的影响可以忽略不计。

三、 符号说明

表 **1** 本文所用符号的相关说明与解释

符号	意义	说明
$RMSE$	均方根误差	衡量模型预测值与真实值之间的误差
MAE	平均绝对误差	衡量预测值与实际值的平均偏差
R^2	判定系数	衡量模型对数据方差的解释能力
MPE	平均百分比误差	衡量预测值与实际值之间的平均百分比差异
$MAPE$	平均绝对百分比误差	衡量预测值与实际值之间平均绝对百分比差异
d_i	秩差	第 i 个数据点的两个变量的秩差
ρ	斯皮尔曼秩相关系数	衡量非参数数据的相关性
Σ	协方差矩阵	描述不同特征之间的关系
λ_i	特征值	协方差矩阵的特征值
X_{scaled}	标准化数据	对数据进行标准化处理后的值
R_j	冲突性	用于计算指标之间的相关性
S_j	可变性	指标的标准差
W_i	权重向量	各特征的权重
ω	综合权重系数	综合主观和客观权重的系数
X	输入向量	包含 20 个特征的输入向量
σ	标准差	预测概率的标准差
Q, K, V	查询矩阵、键矩阵、值矩阵	Transformer 模型中的矩阵
d_k	缩放因子	自注意力机制中的缩放因子

四、 数据处理

4.1 数据概览

在不进行删除异常值或数据增强等操作前，我们要在 Pearson 相关系数、Cramér's V 度量、Kendall 相关系数、Spearman 等级相关系数等相关性检验方案中，筛选出最适合描述洪水概率及 20 个指标间关联的相关性检验方法。

Cramér's V 度量和 Kendall 相关系数适用于分析分类变量之间的相关关系，而洪水概率和指标数据多为连续变量或有序分类变量，因此不太适用。Spearman 等级相关系数和 Pearson 相关系数的计算结果更易于解释，可以直观地反映变量之间的关系强度和方向。

考虑到以上两点原因，我们首先利用皮尔逊相关系数 (Pearson correlation coefficient) 和斯皮尔曼秩相关系数 (Spearman's rank correlation coefficient) 对其中数据进行了分析，结果如图2。从图2与计算结果中可以看到，相关性的 p 值的最大值为 0.181399，这个值

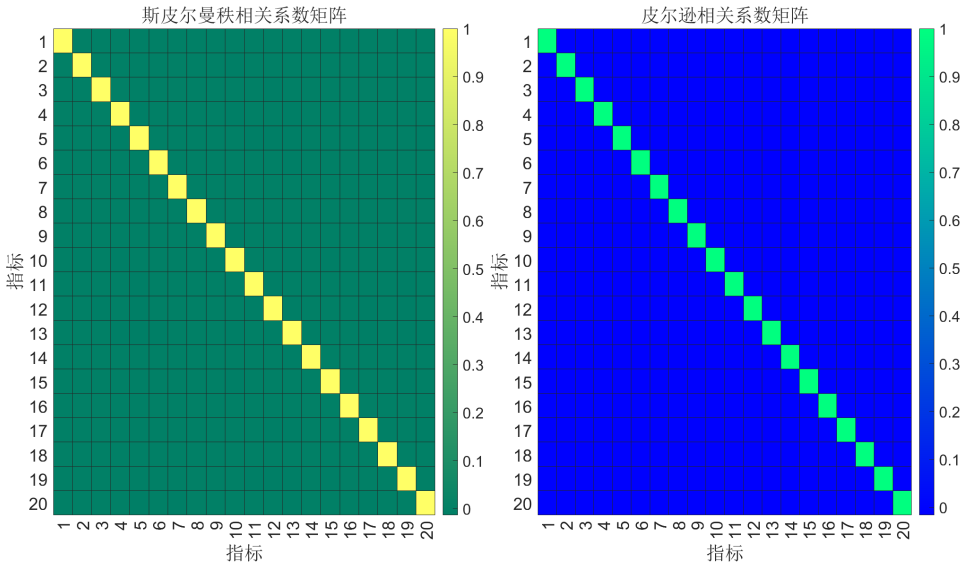

图 **2** 数据处理前皮尔逊相关系数与斯皮尔曼秩相关系数热力图

远小于 0.3，故可以得出结论：单一的特征指标与洪水概率间并不具有极强的相关性。

4.2 数据优化

为解决数据概览部分中遇到的" 极低相关性" 问题，我们考虑使用统计学手段对数据做出优化处理，进一步增强数据的特征，达成提升后续问题求解可靠性的目的。

基于我们的"数据完整性假设"和"独立同分布假设"，我们要对数据进行预处理。数据预处理主要分为四个步骤：删除异常值，补全缺失值，数据标准化，数据归一化。

首先，我们可以根据箱线图中的 **IQR(四分位间距)** 来识别和剔除异常值。

$$IQR = Q_1 - Q_3\,,\ T = 1.5 \times IQR \tag{1}$$

其中，上四分位数为 Q_3，下四分位数为 Q_1，阈值为 T。

这些异常值被删除后成为了空白的缺失值。在我们的阈值设置假设中，对数据集的质量和特征进行了限制，因此数据集中的每个样本都可以被视为一个向量，并且可以使用 **余弦相似度**来填充 **缺失值**。

$$\text{similarity}(x, y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2}$$

然后，我们对数据进行 **z-scores 标准化**处理，设原始数据集中的一个特征为 $x = [x_1, x_2, \ldots, x_n]$，其标准化后的结果为 $z = [z_1, z_2, \ldots, z_n]$，计算公式为：

$$z_i = \frac{x_i - \mu}{\sigma} \tag{3}$$

其中：x_i 是原始数据中的第 i 个样本；μ 是原始数据的均值；σ 是原始数据的标准差。

这种标准化后，每个特征的均值为 0，标准差为 1，有助于确保各个特征的数值范围一致，避免某些特征因数值过大或过小对模型训练产生不利影响。

最后，为了把数据按比例缩放到一个固定范围内，即：[0, 1] 之间，我们使用最小-最大缩放（Min-Max Scaling）进行数据归一化，公式如下：

设原始数据集中的一个特征为 $x = [x_1, x_2, \ldots, x_n]$，其归一化后的结果为 $x' = [x'_1, x'_2, \ldots, x'_n]$，计算公式为：

$$x'_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{4}$$

其中：x_i 是原始数据中的第 i 个样本；$\min(x)$ 是原始数据的最小值；$\max(x)$ 是原始数据的最大值。

这种归一化后，每个特征的数值被压缩到 [0, 1] 之间，有利于处理不同量纲和范围的数据，使其更适合许多机器学习算法。

这样，相较于未处理的原始数据，当我们再解决问题一的要求时，这份经过预处理的数据将表现出更优良的的特征。

五、 问题一：相关因素的识别与预防

5.1 相关性检验

5.1.1 Jarque-Bera 检验与 Kolmogorov-Smirnov 检验

在数据预处理的基础上，我们要先 20 个指标的数据分布特征进行分析，从而在皮尔逊相关系数 (Pearson correlation coefficient) 和斯皮尔曼秩相关系数中选择其一。Pearson 相关系数适用于分析两个连续变量之间的线性关系，要求数据呈正态分布或近似正态分布。而 Spearman 等级相关系数适用于分析两个变量之间的单调关系，不要求数据呈正态分布，对异常值和离群点不敏感。

我们采用了 J-B 检验 (Jarque-Bera)，该检验用于初步筛选正态性。(如图3, 图4)。J-B 检验主要检验数据的偏度和峰度是否与正态分布一致。如果 J-B 检验显示数据的偏度和峰度不显著偏离 0 和 3，那么可以初步推断数据可能服从正态分布。这种初步筛选有助于在后续分析中做出更合理的假设和处理。

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right) \tag{5}$$

其中，S 是样本的偏度，K 是样本的峰度，n 是样本数量。

然后，为了更全面地评估数据的分布特性，我们再使用 K-S 检验 (Kolmogorov-Smirnov) 进行更严格的分布检验.

对于单样本 K-S 检验，统计量 D 的计算公式为：

$$D = \max(|F(x) - F_0(x)|) \tag{6}$$

其中：$F(x)$ 是样本的经验分布函数；$F_0(x)$ 是理论分布（如标准正态分布）的累积分布函数。

对于两样本 K-S 检验，统计量 D 的计算公式为：

$$D = \max(|F_1(x) - F_2(x)|) \tag{7}$$

其中：$F_1(x)$ 和 $F_2(x)$ 是两个样本的经验分布函数。

K-S 检验通过比较经验分布函数和理论分布函数的差异，来判断样本是否来自于特定的理论分布。统计量 D 越小，说明样本与理论分布的拟合越好。通常根据样本量确定临界值，若统计量 D 大于临界值，则认为样本不服从理论分布。

5.1.2 相关性分析：斯皮尔曼秩相关系数法

据 Q-Q 图（Quantile-Quantile plot）结果与分布检验结果 p-value 所示，处理后的数据样本相关性，较之处理前的相关性略有提升，我们选择了更为恰当的相关系数检验方

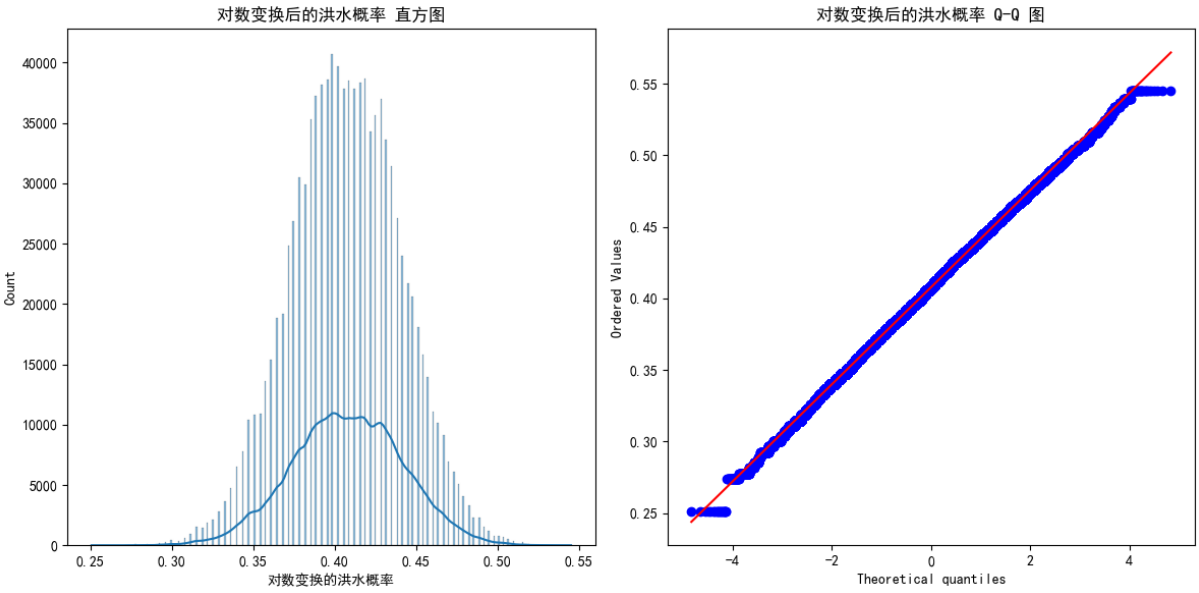

图 3　对数变换后，洪水概率的 **J-B**，**K-S** 检验

法——斯皮尔曼秩相关系数法。

该方法的核心思想是将原始数据转换为秩次，然后计算秩次之间的相关性。这种方法特别适用于非参数数据的分析，能够在数据不满足正态分布假设时提供可靠的相关性估计。通过这种方法，我们可以更加准确地评估洪水发生概率与各影响因素之间的相关性，其计算公式为：

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{8}$$

其中，d_i 是第 i 个数据点的两个变量的秩差，n 是数据点的数量。

图5展示了斯皮尔曼秩相关系数法的可视化结果。从图中可以看出，洪水发生概率与降水量、气温等因素之间的相关性得到了显著提升，这与我们的预期相符。这表明，在考虑了数据的非线性特征和异常值的影响后，我们的分析结果更加准确地揭示了洪水发生的潜在因素。

5.2 相关性强弱的原因

5.2.1 相关性强的指标分析

当研究洪水发生概率时，通过斯皮尔曼秩相关系数法，我们可以识别出几个与洪水发生概率显著相关的因素。以下是可能具有较强相关性的五个因素的分析：

- **地形排水:** 地形排水是指地势高低和地表的排水系统对洪水形成和扩展的影响。通常情况下，地势较高的地区更容易排水，减少了洪水的积聚和持续时间，从而降低了洪水的发生概率。因此，地形排水与洪水发生概率之间可能存在显著的负相关关

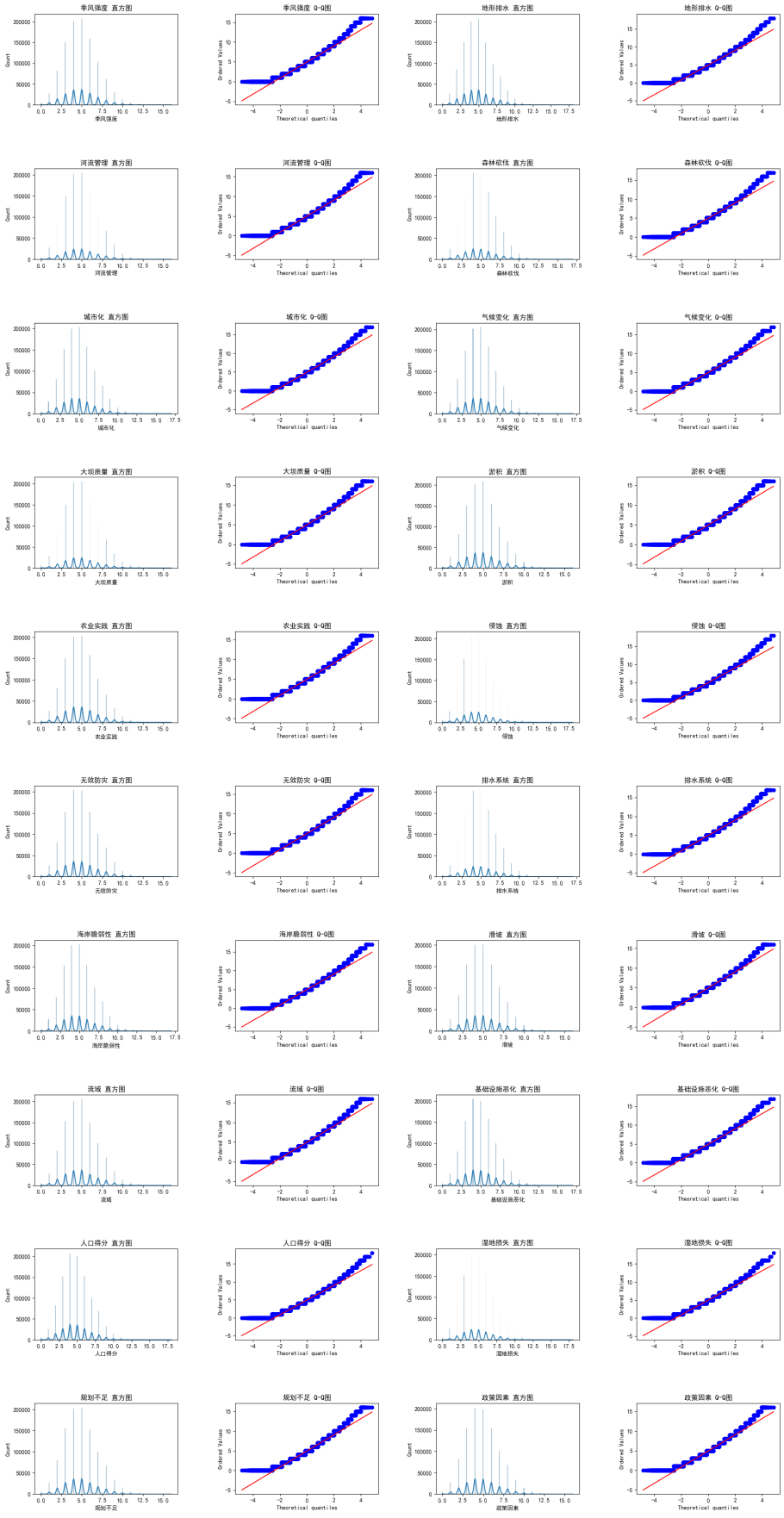

图 **4** 季风强度、地形排水等其他二十个指标的 **J-B**，**K-S** 检验

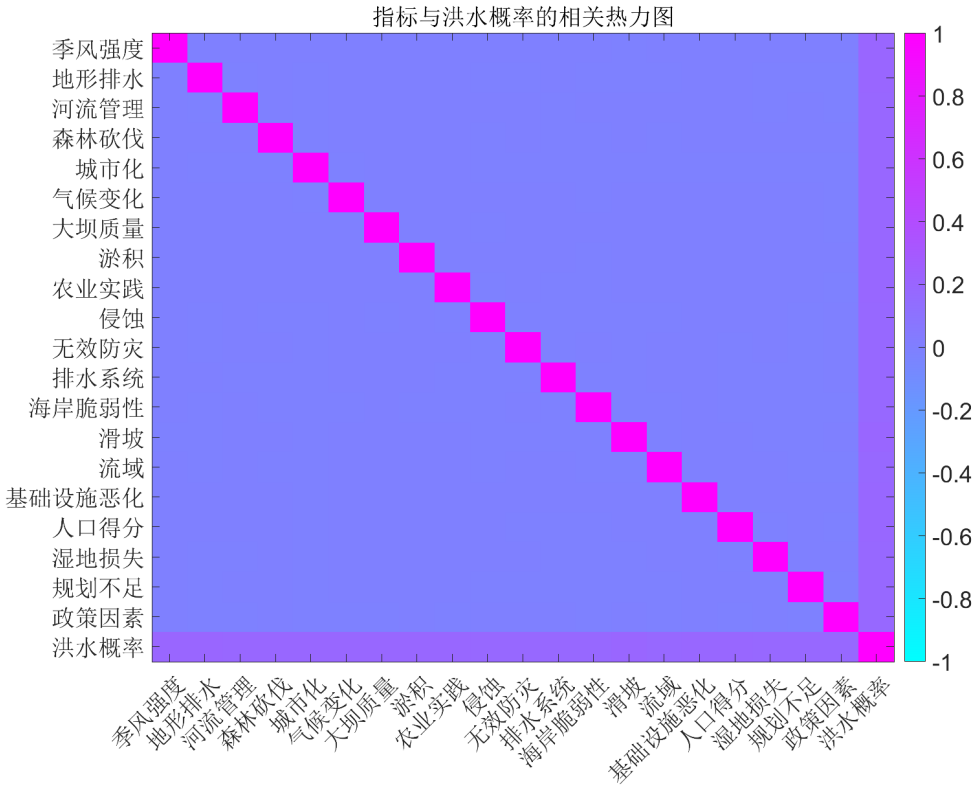

图 **5** 斯皮尔曼秩相关系数热力图

系，即地形越高，洪水发生的概率越低。

- **侵蚀:** 土壤侵蚀程度反映了土地覆盖和土壤保护状况。侵蚀严重的地区，特别是在降雨较多的情况下，土壤的保护能力较差，易造成水土流失，进而增加洪水的发生概率。因此，侵蚀与洪水发生概率可能呈现正相关关系，即侵蚀程度越严重，洪水发生的概率可能越高。
- **人口密度:** 高人口密度地区通常存在较多的城市化和土地利用变化，这些因素会影响雨水的径流和地表水的积聚。因此，人口密度与洪水发生概率可能存在正相关关系，即人口密度越高的地区，由于城市化和土地利用变化的影响，洪水发生的概率可能越高。
- **湿地损失:** 湿地对于调节水文循环、吸收雨水和减缓洪峰具有重要作用。湿地损失导致了这些生态系统功能的削弱，使得洪水的影响更为显著。因此，湿地损失与洪水发生概率可能呈现正相关关系，即湿地损失越严重，洪水发生的概率可能越高。
- **基础设施恶化:** 基础设施恶化包括城市排水系统、防洪设施等的老化或不完善。恶化的基础设施可能无法有效地排除降水或者在洪水发生时提供必要的防护措施，从而增加了洪水的发生概率。因此，基础设施恶化与洪水发生概率可能存在正相关关系，即基础设施恶化程度越严重，洪水发生的概率可能越高。

以上因素是可能与洪水发生概率显著相关的关键因素。它们的相关性分析不仅有助于理

解洪水形成的复杂机制，还为制定有效的洪水管理策略提供了重要的参考依据。在实际应用中，结合这些因素的相互影响和时空变化，可以更准确地评估和预测洪水风险，从而采取针对性的防洪措施和应急响应策略。

5.2.2 相关性弱的指标分析

尽管我们通过斯皮尔曼秩相关系数法识别出了多个与洪水发生概率显著相关的因素，但也有一些指标显示出较弱的相关性，如政策因素、规划不足、海洋脆弱性等。

- **政策因素**：政策因素通常包括政府制定的防洪措施、灾害预警系统、应急预案等。这些因素在理论上应该与洪水发生概率有一定的相关性。然而，在相关性分析中，我们发现政策因素的影响较弱，我们猜测，这可能是由于政策实施的效果需要较长时间才能显现，或者是因为政策执行的不一致性导致的。
- **规划不足**：规划不足可能导致城市扩张到高风险区域，或者建筑物的设计不考虑防洪标准，从而增加洪水发生的风险。尽管如此，规划不足与洪水发生的相关性可能较弱，我们猜测这可能是由于规划的影响通常是间接的，在短期内难以量化。
- **海岸脆弱性**：海岸脆弱性通常与海洋洪水或风暴潮有关，但在我们的研究中，这一指标与洪水发生的相关性较弱。可能是因为我们所研究的区域内陆较远，或者海岸脆弱性对洪水的影响不如其他因素显著。

尽管相关性分析表明上述三个指标的相关性较弱，但这并不意味着这些指标在实际情况中不重要。相关性分析仅仅反映了数据之间的统计关系，而实际影响可能更为复杂。例如，政策因素和规划不足可能在长期内对洪水风险产生显著影响，而海岸脆弱性和排水系统的性能可能在沿海地区成为洪水爆发的关键因素。因此，在制定防洪策略时，我们仍需综合考虑这些指标，并进一步研究它们在不同情境下的作用。

5.3 建议与措施

鉴于上述对政策因素、规划不足及海岸脆弱性等指标相关性较弱的分析，我们提出一系列合理建议与措施，以期在防洪策略制定中更全面、有效地考虑这些因素：

1. **政策因素强化与评估：**
   - 长期监测与评估：建立政策实施效果的长期监测机制，定期评估防洪措施、灾害预警系统及应急预案的有效性，及时调整策略以优化政策效果。
   - 政策一致性保障：加强政策执行过程中的监督与反馈，确保政策的一致性和有效性，减少因执行不一致导致的政策效果弱化。
   - 公众教育与参与：提高公众对防洪政策的认知度和参与度，通过宣传教育增强社会整体的防洪意识和能力。
2. **规划优化与风险管理：**

- 综合规划与风险评估：在城市和区域规划中纳入全面的洪水风险评估，确保扩张计划和建筑设计符合防洪标准，减少规划不足带来的风险。
- 动态调整与反馈：建立规划调整机制，根据洪水风险评估结果动态调整规划方案，确保规划的有效性和适应性。
- 跨部门协作：加强政府各部门之间的沟通与协作，确保防洪规划与土地利用、基础设施建设等规划之间的协调一致。

3. **综合决策支持系统建设：**

- 集成多源数据：构建综合决策支持系统，集成政策、规划、海岸脆弱性等多源数据，为防洪策略制定提供全面、准确的信息支持。
- 情景模拟与预测：利用情景模拟和预测技术，评估不同政策、规划和环境条件下洪水风险的变化趋势，为决策提供科学依据。
- 动态调整与优化：根据决策支持系统的评估结果，动态调整和优化防洪策略，确保防洪工作的有效性和可持续性。

六、 问题二：风险聚类与预警评价

6.1 K-means 聚类分析

我们从附件 train.csv 数据集中获取了洪水发生概率数据，并对其进行了详细的特征分析。为了简化数据分析的复杂度，我们运用了 K-means 聚类算法，将洪水发生概率分为高风险、中风险和低风险三个类别。

考虑到数据预处理中 Jarque-Bera 检验的结果，我们在 DBSCAN，k-means 两种经典聚类方法中选择 K-means。相较于 DBSCAN，k-means 倾向于找到大小相似的球形簇。洪水概率数据的噪声较小，并且正态性良好，k-means 可能会得到更好的聚类结果

根据 k-means 聚类分析的结果，洪水概率数据可以分为三个风险等级：高、中、低。以下是每个风险等级的聚类中心和包含的数据点数量：

	高风险等级	中风险等级	低风险等级
聚类中心	0.568	0.507	0.446
点集数量	273435	455665	319,475

这三类点集的分类边界为 **[** 0.00000 ， 0.47657 ， 0.53771 ， 1.00000 **]**

该聚类结果如箱线图所示：

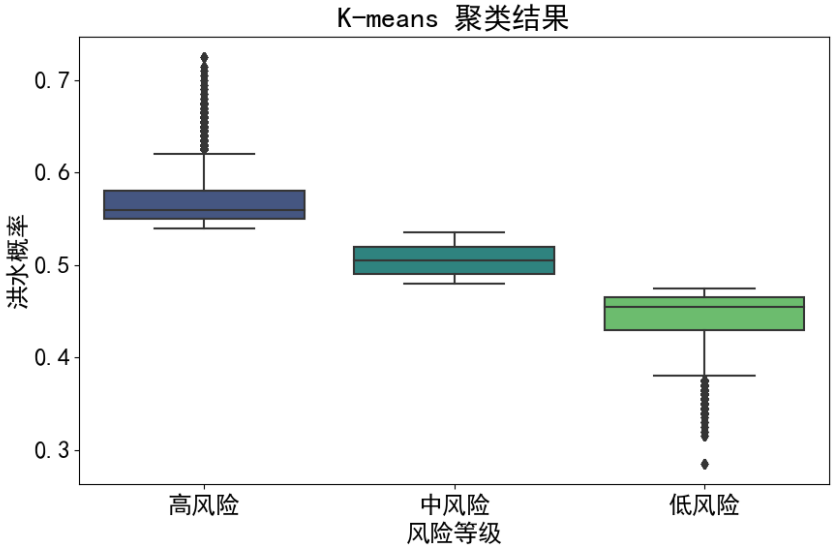

图 **6** **K-means** 聚类结果

6.2 特征选择：Mann-Whitney U 检验

已由数据预处理、问题一的解决中得出：除洪水概率外，其余特征指标均不符合正态分布，为了从 20 个指标中挑选出可以直接影响风险类别的指标，本文采用 Mann-Whitney U 检验对各个指标与各个事件的风险等级之间的相关性进行分析。如果检验结果表明该指标的变化会对洪水发生概率的风险等级产生影响，则提取该特征作为最终评价预警模型的评价指标。

相较于 Kruskal-Wallis H (KWH) 检验，这里选用 Mann-Whitney U (MWU) 检验的优势在于 MWU 检验是一种优秀的非参数检验，它不依赖于数据的分布情况，对于数据的正态性或其他分布形态没有要求。它基于秩次进行推断，因此在数据不满足正态性或方差齐性的情况下仍然有效。而 KWH 检验通常要求数据来自于相同的分布，并且对于每组数据的中位数比较有特定的假设。[1]

Mann-Whitney U 检验（也称为 Wilcoxon 秩和检验）是一种非参数检验方法，用于比较两组独立样本的中位数是否显著不同。下面是 Mann-Whitney U 检验的数学公式和解释：

提取 20 个特征 “季风强度”，” 城市化率” 等 (样本 1) 和目标值” 洪水风险”(样本 2)，进行 Mann-Whitney U 检验的步骤如下：

1. **秩排名:** 将所有的数据（包括两组样本的数据）从小到大排列，并为每个数据分配一个秩（排名）。
2. **秩和**: 计算每组样本的秩和，分别记为 R_1 和 R_2。
3. **U 统计量计算:** 计算 U 统计量，用于检验两组样本是否来自同一总体。
   - 对于 20 个特征中某一列 (样本 1) 的每个数据 X_i，计算其在整体数据中的秩和

R_i。

- 对于目标洪水概率 (样本 2) 这一列的每个数据 Y_j，计算其在整体数据中的秩和 R'_j。
- 计算样本 1 的 U 统计量 U_1，它表示样本 1 中的每个数据 X_i 的秩 R_i 之和，即：

$$U_1 = \sum_{i=1}^{n_1} R_i \tag{9}$$

其中，n_1 是样本 1 的样本量。

- 计算样本 2 的 U 统计量 U_2，它表示样本 2 中的每个数据 Y_j 的秩 R'_j 之和，即：

$$U_2 = \sum_{j=1}^{n_2} R'_j \tag{10}$$

其中，n_2 是样本 2 的样本量。

- 取较小的 U 值作为最终的 U 统计量 U，通常为 $U = \min(U_1, U_2)$。

4. **临界值比较:** 将计算得到的 U 值与临界值进行比较，以判断是否拒绝原假设。

对这 20 个指标的 Mann-Whitney U 检验运行结果如下表所示，p-value=0.0 充分说明，若想最大化预警评价模型的准确度，应当将 20 个指标都纳入预警评价模型的评价指标中。

原始假设	显著性 p-value	最终决策
不同风险等级下，季风强度分布相同	0.0	否定原始假设
不同风险等级下，地形排水分布相同	0.0	否定原始假设
不同风险等级下，河流管理相同	0.0	否定原始假设
不同风险等级下，森林砍伐相同	0.0	否定原始假设
不同风险等级下，城市化相同	0.0	否定原始假设
不同风险等级下，气候变化相同	0.0	否定原始假设
不同风险等级下，大坝质量相同	0.0	否定原始假设
不同风险等级下，淤积情况相同	0.0	否定原始假设
不同风险等级下，农业实践相同	0.0	否定原始假设
不同风险等级下，侵蚀情况相同	0.0	否定原始假设
不同风险等级下，无效防灾措施相同	0.0	否定原始假设
不同风险等级下，排水系统性能相同	0.0	否定原始假设
不同风险等级下，海岸脆弱性相同	0.0	否定原始假设
不同风险等级下，滑坡风险相同	0.0	否定原始假设
不同风险等级下，流域情况相同	0.0	否定原始假设
不同风险等级下，基础设施恶化情况相同	0.0	否定原始假设

不同风险等级下，人口得分相同	0.0	否定原始假设
不同风险等级下，湿地损失情况相同	0.0	否定原始假设
不同风险等级下，规划不足情况相同	0.0	否定原始假设
不同风险等级下，政策因素影响相同	0.0	否定原始假设

6.3 预警评价模型的构建

基于权重计算结果，我们采用 AHP-CRITIC-TOPSIS 综合评估方法成功建立了洪水不同风险的预警评价模型，其中层次分析法 (AHP) 用于主观判断，CRITIC 法用于客观权重计算，TOPSIS 法进行综合评价。该模型综合考虑了多个指标的影响，能够对洪水发生的风险进行相对更为准确的评估。

6.3.1 主观权重：层次分析法 AHP

为了更好地依据确定 20 个特征指标与洪水概率彼此间的权重关系，基于问题一最终得到的 Spearman 相关性的排序，我们选择首先使用 **层次分析法 (AHP)** 来推导主观权重。

因此，考虑到不同客观数据与主观打分的相关性差异，我们建立了一个决策矩阵 $A = (x_{ij})$，然后对特征向量进行归一化处理，得到权重向量：

$$W_i = \frac{(\prod_n^{j=1} a_{ij})^{\frac{1}{n}}}{\sum_n^{i=1} (\prod_n^{j=1} a_{ij})^{\frac{1}{n}}}, ij = 1, 2, ..., n \tag{11}$$

然后，我们构建 20*20 的决策矩阵，具体数值是复杂的繁分数，我们将决策矩阵可视化为热力图放置在图7右，并将繁分数保留一位数字的结果放置在图7左。

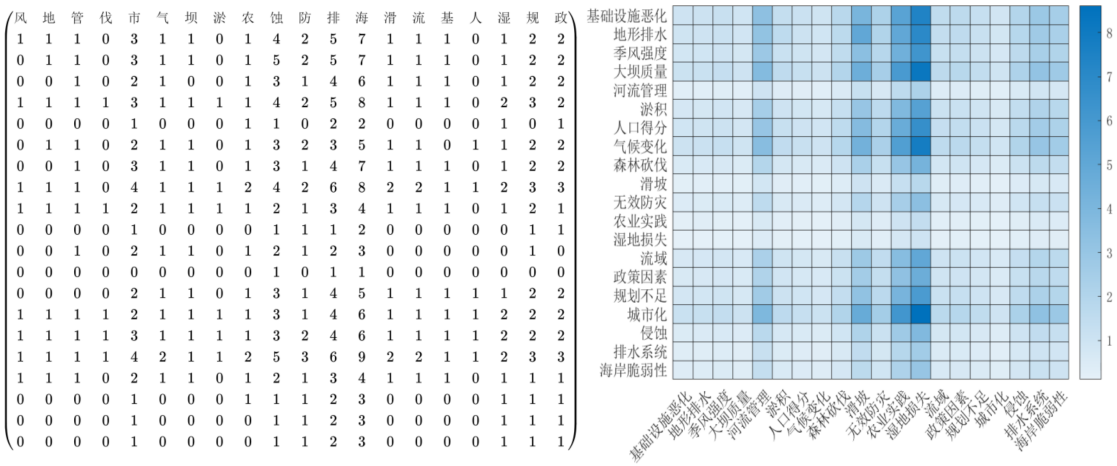

图 **7** 决策矩阵

然后，我们计算一致性比例（CR）和一致性指标（CI），CI = $\frac{\lambda_{max}-n}{n-1}$ 和 CR = $\frac{CI}{RI}$ 的校验，得到 CR 小于 0.1，故认为该决策矩阵具有一致性。我们通过附件中的代码，运算得到权重向量如下：

- W_A= { ” 季风强度”: 0.021077, ” 地形排水”: 0.023433, ” 河流管理”: 0.025022, ” 森林砍伐”: 0.019309, ” 城市化”: 0.066604, ” 气候变化”: 0.031105, ” 大坝质量”: 0.026831, ” 淤积”: 0.023970, ” 农业实践”: 0.037046, ” 侵蚀”: 0.072983, ” 无效防灾”: 0.046370, ” 排水系统”: 0.086145, ” 海岸脆弱性”: 0.109463, ” 滑坡”: 0.058121, ” 流域”: 0.059079, ” 基础设施恶化”: 0.057017, ” 人口得分”: 0.054942, ” 湿地损失”: 0.059108, ” 规划不足”: 0.061431, ” 政策因素”: 0.060943 }

6.3.2 客观权重：关键质量赋权法 CRITIC

此外，除了人为确定关键影响因素的权重之外，考虑到不同指标数据的离散程度，我们结合 **关键质量赋权法 CRITIC** 进行客观加权。

首先，我们需要计算指标的可变性 [1]。这里用标准差来表示各指标内部取值差异的波动情况，标准差越大，指标反映的信息越多，评价力度越强，权重越大：

$$\begin{cases} \bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \\ S_j = \sqrt{\frac{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}{n-1}} \end{cases} \tag{12}$$

其次，我们计算指标的冲突性。这里用相关系数来表示指标之间的相关性，与其他指标的相关性越强，说明该指标与其他指标的冲突性越小，反映的相同信息越多，应舍弃冗余信息，因此权重越低。

$$R_j = \sum_{i=1}^{p}(1 - r_{ij}) \tag{13}$$

接着，计算信息量。C_j 越大，说明第 j 个评价指标在整个评价指标体系中的作用越大，权重越高。

$$C_j = S_j \sum_{i=1}^{p}(1 - r_{ij}) = S_j \times R_j \tag{14}$$

最后，我们计算权重并列举结果：

$$W_j = \frac{C_j}{\sum_{j=1}^{p} C_j} \tag{15}$$

- W_C = { ” 地形排水”: 0.047619, ” 农业实践”: 0.047619, ” 淤积”: 0.047619, ” 湿地损失”: 0.047619, ” 城市化”: 0.047619, ” 森林砍伐”: 0.047619, ” 人口得分”: 0.047619, ” 季风强度”: 0.047619, ” 无效防灾”: 0.047619, ” 气候变化”: 0.047619, ” 洪水概率”: 0.047619, ” 滑坡”: 0.047619, ” 流域”: 0.047619, ” 基础设施恶化”: 0.047619, ” 规划不足”: 0.047619, ” 侵蚀”: 0.047619, ” 排水系统”: 0.047619, ” 政策因素”: 0.047619, ” 大坝质量”: 0.047619, ” 海岸脆弱性”: 0.047619, ” 河流管理”: 0.047619 }

6.3.3 综合权重与打分：TOPSIS

得到上述权重后，我们先对主观权重和客观权重进行线性运算，在这种情况下，ω 等于 0.6：

$$W_{comba} = \omega \cdot W_{AHP} + (1-\omega) \cdot W_{CRITIC} \tag{16}$$

综合权重的完整结果如玫瑰图：

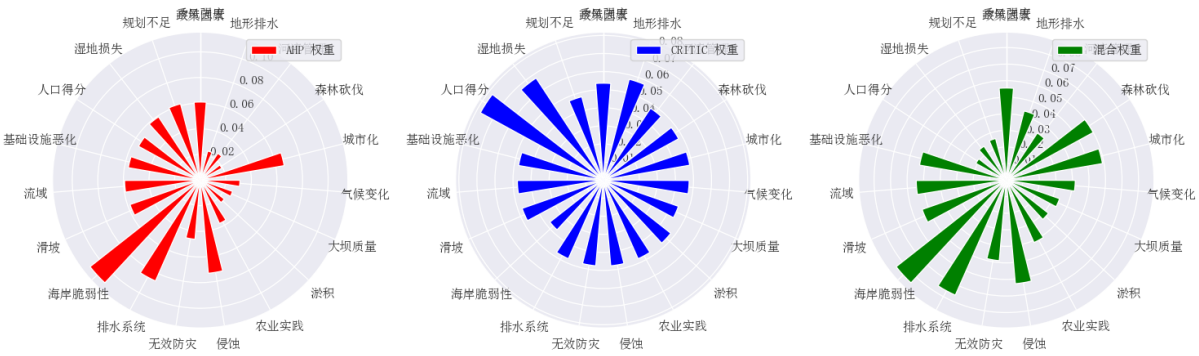

图 8　A-C-T 评价权重玫瑰图

最终，我们使用 **理想优劣距离解 (TOPSIS)** 方法来计算各方案与理想 (最优) 和反理想 (最差) 方案的距离，并以与理想方案的相对接近程度作为标准 [1]，对不同图像的优劣程度进行评分：

$$D_i^+ = \sqrt{\sum_{j=1}^{m} w_j (Z_j^+ - z_{ij})^2}, \; D_i^- = \sqrt{\sum_{j=1}^{m} w_j (Z_j^- - z_{ij})^2} \tag{17}$$

最终，我们定义了一个衡量标准，用于评估对象与最优解的接近程度。在这种情况下，C_i 的值越大，评估对象就越优化。

$$C_i = \frac{D_i^-}{D_i^- + D_i^+} \tag{18}$$

我们将 AHP-CRITIC-TOPSIS 评价模型的 1048572 条得分，部分展示如下表：

ID	TOPSIS 得分
0	0.443245
1	0.448276
2	0.473388
……	……
1048571	0.477437
1048572	0.515712

将这部分数据可视化为直方图，可以得到正态性极佳的直方图：

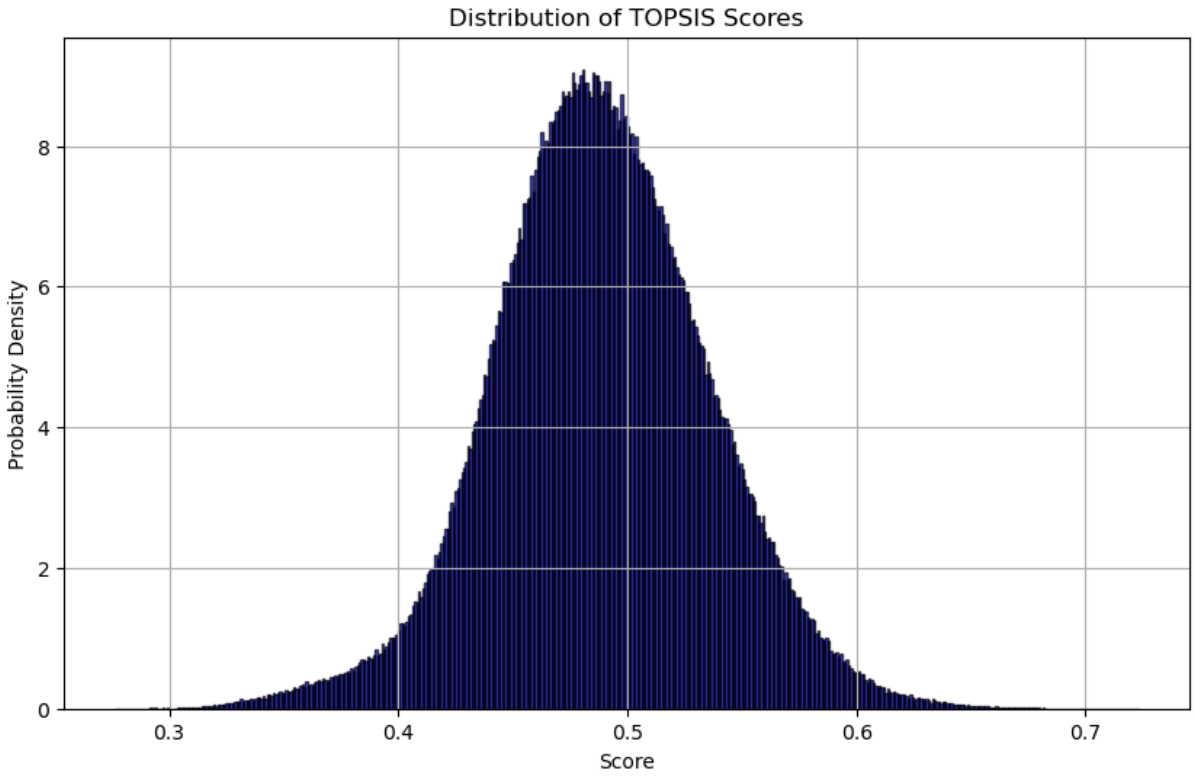

图 **9** **AHP-CRITIC-TOPSIS** 得分分布

6.4 灵敏度分析与鲁棒性检验

在问题 2 中，我们先使用了 K-means 聚类分析，将高中低三个风险等级的取值范围界定下来。其次，我们运用 AHP-CRITIC-TOPSIS 的复合赋权预警评价模型，得到了同时综合 20 个特征指标后预测的洪水发生风险。

经 3-σ 检验，基于 AHP-CRITIC-TOPSIS 的得分分布呈现良好的正态性，这个打分的结果，与我们在图3中，对洪水概率经对数变换后 Jarque-Bera 检验结果不谋而合，这同时也说明了该预警评价模型的可靠性。

因此，为了分析问题二整体模型的灵敏度，并检验该问题整体模型的鲁棒性，我们需要同时检验基于 K-means 的洪水风险等级聚类模型和基于 AHP-CRITIC-TOPSIS 的洪水预警评价模型。

6.4.1 K-means 洪水风险等级聚类模型的灵敏度分析

为了更好地检验该聚类模型下的灵敏度，我们需要获取经过 Mann-Whitney U 检验的 20 个特征指标对洪水概率的具体影响。在此基础上，我们才能判断这些指标 (如季风强度、政策因素) 的变化，究竟会对洪水发生的风险等级有何影响。

在雷达图聚类过程中，虽然没有单一的数学公式可以完全描述聚类的所有步骤，但可以结合数学原理和图形表达来解释其聚类过程。以下是如何在数学和图形化方式上解

释雷达图聚类的关键步骤：

我们先对 20 个特征指标也进行聚类分析，将其按照以下公式聚成三类：

即最小化以下目标函数：

$$\underset{\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2 \tag{19}$$

其中，S_i 是第 i 个聚类的集合，μ_i 是第 i 个聚类的中心。

在得到标准化后的聚类中心后，需要将其反标准化为原始数据的尺度，以便进一步解释和分析。

$$\mu_{i,original} = \mu_{i,scaled} \cdot \sigma + \mu \tag{20}$$

其中，$\mu_{i,original}$ 是第 i 个聚类中心在原始数据尺度上的值，$\mu_{i,scaled}$ 是在标准化数据尺度上的聚类中心，σ 和 μ 分别是特征的标准差和均值。

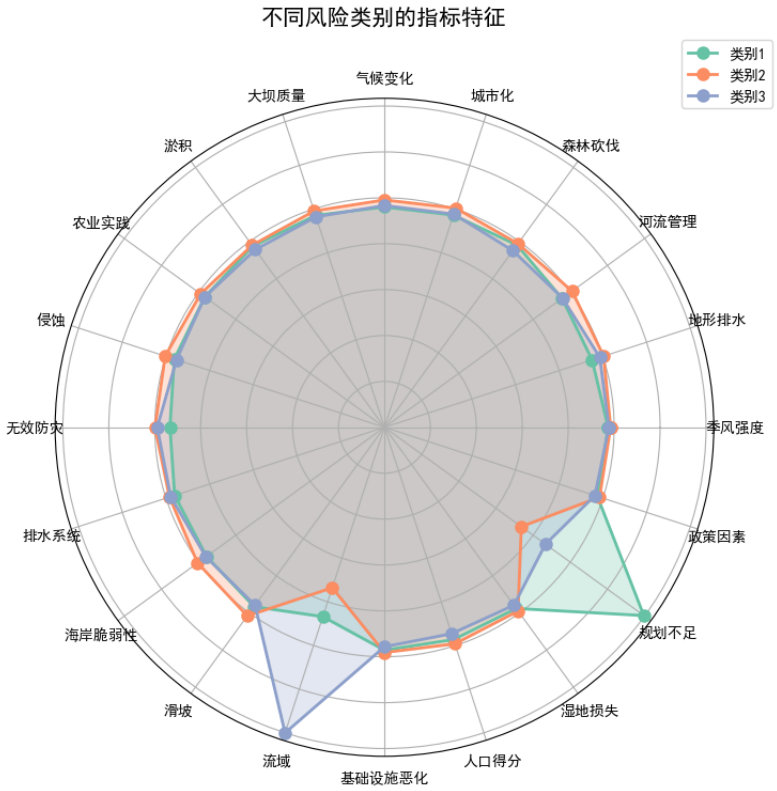

图 **10** 特征聚类雷达图

这里我们使用雷达图，来可视化聚类中心的特征值，以展示不同聚类的特征差异。每个雷达图中的轴代表一个特征，轴的值表示该特征的具体数值。不同类别（聚类）的聚类中心在雷达图中以多边形的形式展示，不同聚类之间的区别通过多边形的形状和大小来比较。

在图10中，我们清楚地看到，流域和规划不足对洪水风险等级变动有较大影响，地形排水和无效放在对洪水风险等级有适量影响，其余特征指标影响较小。

在得到聚类雷达图后，下面我们将对于每个特征，稍微扰动其值（例如增加其标准差的 10%），然后重新预测数据点的聚类分配。其简要思路如下：

- 基本预测（Base Prediction）：记模型在原始输入数据上的预测结果为 $X_i' = X_i + \epsilon \cdot \sigma_i$
- 扰动输入（Perturbed Input）：对于第 i 个特征 X_i，进行微小扰动：

$$X_i' = X_i + \epsilon \cdot \sigma_i \tag{21}$$

  其中，ϵ 是小幅度扰动的比例，通常取值很小（如 $\epsilon = 0.1$），σ_i 是第 i 个特征 X_i 的标准差。
- 扰动后的预测（Perturbed Prediction）：计算扰动后的预测结果 $\hat{y}^{(perturbed)}$。
- 灵敏度（Sensitivity）：对于每个特征 X_i，计算扰动后的预测结果与基本预测结果的平均绝对差（Mean Absolute Difference）：

$$S_i = \frac{1}{N} \sum_{j=1}^{N} |\hat{y}_j^{(perturbed)} - \hat{y}_j^{(base)}| \tag{22}$$

  其中，N 是样本数量，$\hat{y}_j^{(perturbed)}$ 和 $\hat{y}_j^{(base)}$ 分别是第 j 个样本在扰动后和基准情况下的预测结果。
- 结果解释：对于每个特征 X_i，S_i 越大表示模型对该特征的变化越敏感。

计算扰动后的聚类结果与基准聚类结果之间的差异，作为该特征的灵敏度度量。我们将结果可视化如图：

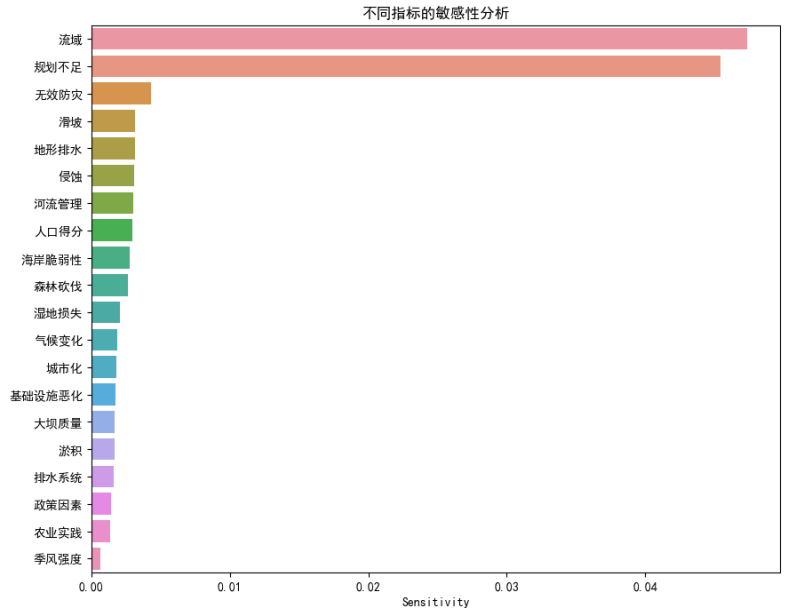

图 **11** 灵敏度分析

其中，仍然是流域和规划不足两个指标灵敏度较高，而其余十八个特征数据的灵敏度较低。总体来说，该模型灵敏度较低，该模型具有足够良好的泛化能力。

6.4.2 K-means 洪水风险等级聚类模型的鲁棒性检验

为了进一步显化特征指标变动对等级划分的影响，我们使用主成分分析法 (PCA) 得出指标的重要程度。主成分分析是一种降维技术，它通过线性变换将原始数据转换为一组互不相关的主成分，以便于分析数据的结构和特征的重要性。我们有 d 维数据，其中每个样本有 p 个特征。其简化流程如下：

- **协方差矩阵（Covariance Matrix）：** PCA 的核心是计算数据的协方差矩阵，它描述了不同特征之间的关系。

$$\Sigma = \frac{1}{n-1}(X_{scaled} - \bar{X})^T(X_{scaled} - \bar{X}) \tag{23}$$

其中，n 是样本数量，$\bar{X}$ 是数据的均值向量。

- **特征值分解（Eigenvalue Decomposition）：** PCA 通过对协方差矩阵进行特征值分解来找到数据中最重要的主成分。

$$\Sigma v_i = \lambda_i v_i \tag{24}$$

其中，λ_i 是第 i 个特征值，v_i 是对应的特征向量。

- **主成分的计算：** 主成分是通过选择协方差矩阵中最大特征值对应的特征向量来定义的。

$$\mathbf{PC}_i = X_{scaled}\mathbf{v}_i \tag{25}$$

其中，$\mathbf{PC}_i$ 是第 i 个主成分，$\mathbf{v}_i$ 是第 i 个特征向量。

在附录的代码中，PCA 被用来计算每个特征的重要性，通过解释第一个主成分的特征向量的绝对值来实现。这种方法帮助识别和理解数据集中哪些特征对解释和聚类结果具有最大的影响。我们将结果可视化如图12。

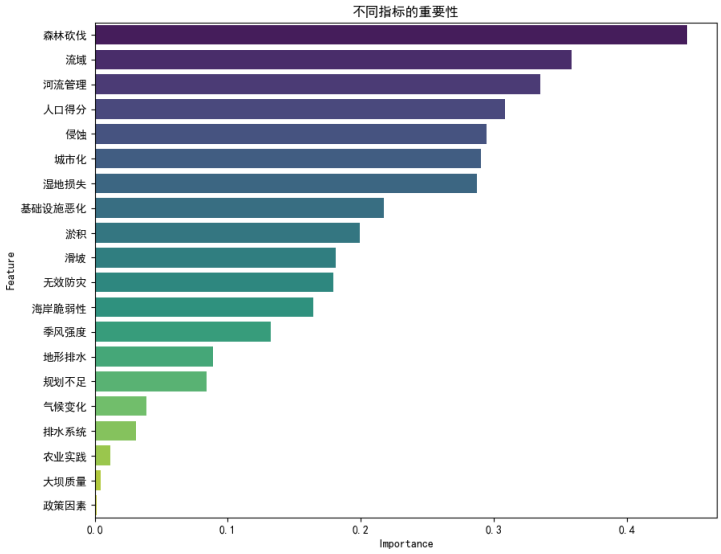

图 **12** 基于主成分分析的重要性

在主成分分析中，我们可以清楚地看到各成分的重要性，下面，我们对传入数据集进行小幅度微扰，并添加一些噪声，检验其鲁棒性。我们将将完整过程简化如下：

- 平均轮廓系数（Average Silhouette Score）：衡量簇的紧密性和分离度，用于评估聚类质量。平均轮廓系数 S 的计算方式为：

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i) \tag{26}$$

其中 $s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$

- 误差平方和（Sum of Squared Errors, SSE）：评估聚类结果的紧凑性。误差平方和（SSE）的计算方式为：

$$SSE = \sum_{i=1}^{n}\sum_{x\in C_i} \|x - c_i\|^2 \tag{27}$$

- 聚类中心的稳定性：比较不同扰动条件下聚类中心的变化程度。聚类中心的稳定性（MSD）的计算方式为：

$$MSD = \frac{1}{k}\sum_{j=1}^{k} \|c_j - \bar{c}_j\|^2 \tag{28}$$

- 结果分析：较小的变化或噪声对模型结果造成的影响较小，表明模型具有较好的鲁棒性。

我们将鲁棒性检验的结果以箱线图可视化如下，噪声几乎不影响模型的输出，该模型具有良好的鲁棒性。

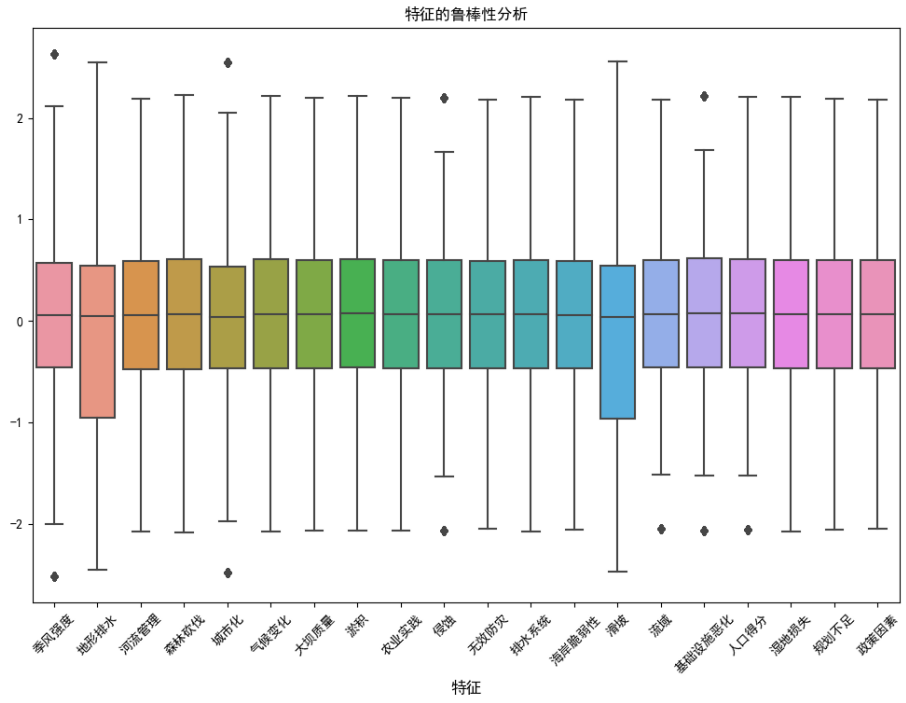

图 **13** 鲁棒性检验箱线图

6.4.3 A-C-T 洪水预警评价模型的灵敏度

上面我们分析了风险等级的灵敏度和鲁棒性，下面我们将分析预警评价模型的灵敏度和鲁棒性。对于 AHP-CRITIC-TOPSIS，我们可以试着更改决策矩阵的主观值，然后分析改变前后，最终评分分布的情况。为了简化描述，我们用热力图显示决策矩阵的改变情况：

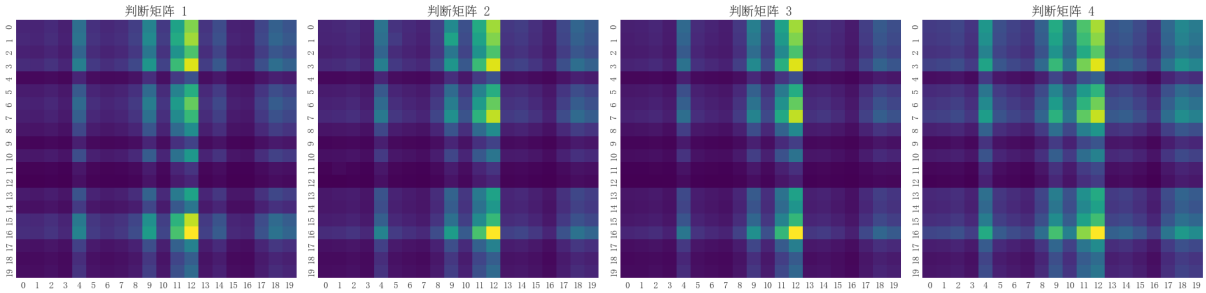

图 **14** 决策矩阵微调

改变决策矩阵后，得到的 A-C-T(AHP-CRITIC-TOPSIS) 结果如图：

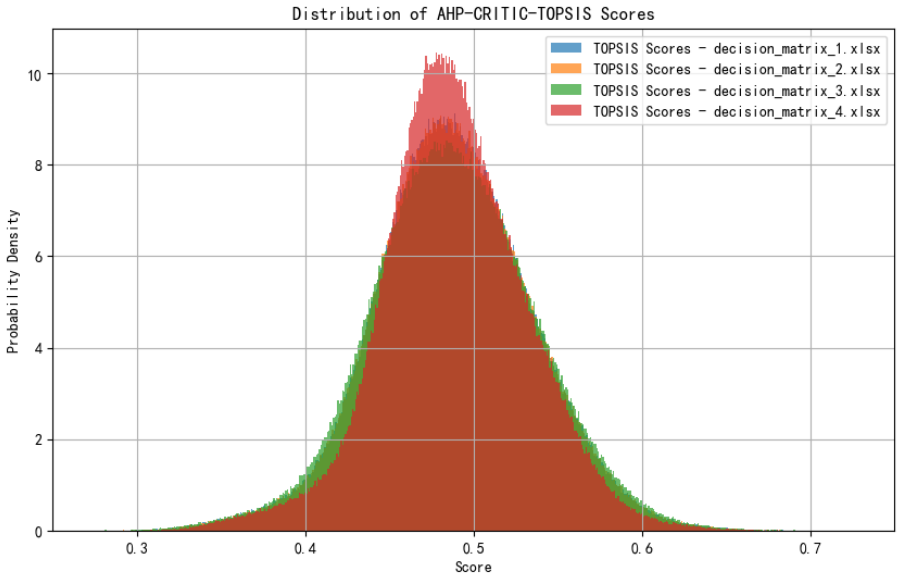

图 **15** **A-C-T Scores:** 灵敏度分析

6.4.4 A-C-T 洪水预警评价模型的鲁棒性

为了测试 A-C-T 模型的鲁棒性，我们对数据集进行小幅度扰动和使用随机森林 (Random Forest) 进行数据增强，然后再对初始决策矩阵进行打分，得到如图的结果：测试结果表明，A-C-T 模型具有良好的灵敏度和鲁棒性，使得该模型既有良好的泛化能力，又能保证泛化后对抗噪声与微扰的稳定性，可以稳定而广泛地应用于多场景的洪水概率预测。

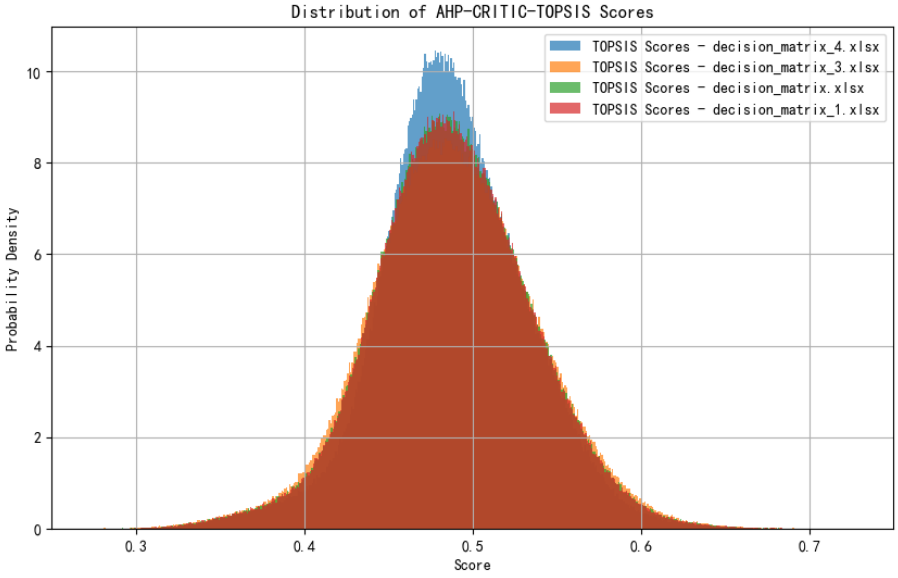

图 **16** **A-C-T Scores:** 鲁棒性检验

七、 问题三：概率预测与特征降维

基于前两问的成果，我们先利用 20 个特征指标，构建机器学习算法对洪水概率进行预测。在此基础上，我们将采取特定指标，对 20 个特征指标进行降维，最终保留 5 个特征指标进行预测，根据预测情况，再做进一步优化。

相较于简单的决策树、随机森林等模型 [7]，我们在洪水概率预测的命题背景下选择使用 LightGBm 进行预测。LightGBM（Light Gradient Boosting Machine）之所以能较好地适配当前情况下的洪水概率预测模型，主要基于以下几个关键因素：

1. **损失函数优化**：LightGBM 是在梯度提升机（Gradient Boosting Machine，GBM）框架下进行优化的。GBM 是一种集成学习技术，通过逐步训练多个弱学习器（通常是决策树），每一轮迭代都试图修正上一轮迭代的残差，以最小化损失函数。LightGBM 在传统的 GBM 算法上进行了改进和优化。[8]
2. **较低的时空占有率**：LightGBM 在性能和效率上进行了多方面的优化，尤其是在处理**大规模**数据集和**高维特征**时表现出色。其优化方法包括但不限于：
   - 基于直方图的决策树算法：LightGBM 使用直方图算法来加速特征的搜索和分裂过程，大大减少了计算成本。
   - leaf-wise 生长策略：相比传统的 level-wise 生长策略，Leaf-wise 生长策略可以更快地找到更好的分裂点，从而有效地降低损失函数。[8]
   - 特征并行：LightGBM 支持特征并行计算，能够更快地处理大规模数据集。
3. **综合考量适配优点**：
   - 高效性：相比传统的 GBM 算法，LightGBM 在训练速度和内存使用效率上有显

著提升，尤其适合处理大规模数据。

- 准确性：LightGBM 能够通过调整参数和增加迭代次数来提高模型的准确性，且通常能达到较高的预测精度。
- 支持大规模数据集：由于其优化策略，LightGBM 能够处理数百万甚至数十亿样本和大量特征的数据集，而不会过度拟合。[8]
- 可扩展性：LightGBM 支持并行化处理和分布式训练，可以在多核 CPU 上进行训练，甚至可以与 GPU 加速结合使用，进一步提高训练速度。

综上所述，选择 LightGBM 主要因为其在 GBM 框架上的优化和创新，使得在大规模数据集和复杂特征空间中的表现更为出色，同时提供了更快的训练速度和更好的预测准确性。[7]

7.1 基于 LightGBM 的全特征值预测

我们的目标是预测洪水概率 $\hat{y}$，根据训练数据得到了一个 LightGBM 模型，表示为 $\hat{y} = f(X)$，其中：$X = (x_1, x_2, ..., x_{20})$ 是包含 20 个特征的输入向量。$f(X)$ 是模型的预测函数，它是由多个决策树组成的集成模型。

7.1.1 模型训练

LightGBM 模型在训练过程中，通过优化损失函数来最小化预测值 $\hat{y}$ 和真实值 y 之间的差异，通常采用均方误差（MSE）作为损失函数。我们将完整过程简化如下：

1. 模型训练过程：在训练阶段，LightGBM 会根据训练集中的特征 X 和目标变量 y，学习多棵决策树以最小化预测误差。每棵树通过划分特征空间，将训练集分割成不同的区域，从而使每个区域内的预测值尽可能接近实际观测值 y。
2. 集成模型预测：训练完成后，整个 LightGBM 模型 $\hat{y} = f(X)$ 可以对新的输入数据 X 进行预测，得到洪水概率的估计值 $\hat{y}$。
3. 特征重要性：LightGBM 还可以提供每个特征的重要性评估，这有助于理解哪些特征对于洪水概率预测的贡献最大。重要性评估可以基于特征的增益（gain）或分裂（split）指标，这些指标反映了在模型训练中，每个特征在决策树节点分裂过程中的重要性。[9]
4. 模型评估：在训练完成后，需要使用测试集或交叉验证来评估模型的泛化能力和预测准确性。常见的评估指标包括均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）、决定系数（R^2 Score）等。

因此，基于全部 20 个特征进行 LightGBM 预测的过程是一个端到端的机器学习任务，通过学习特征之间的复杂关系和数据中的模式，使得模型能够有效地预测洪水概率。

我们按照 7:3 拆分训练集和测试集，同时提取交叉验证集。完整代码展示在附录。其回归结果进行可视化如图：

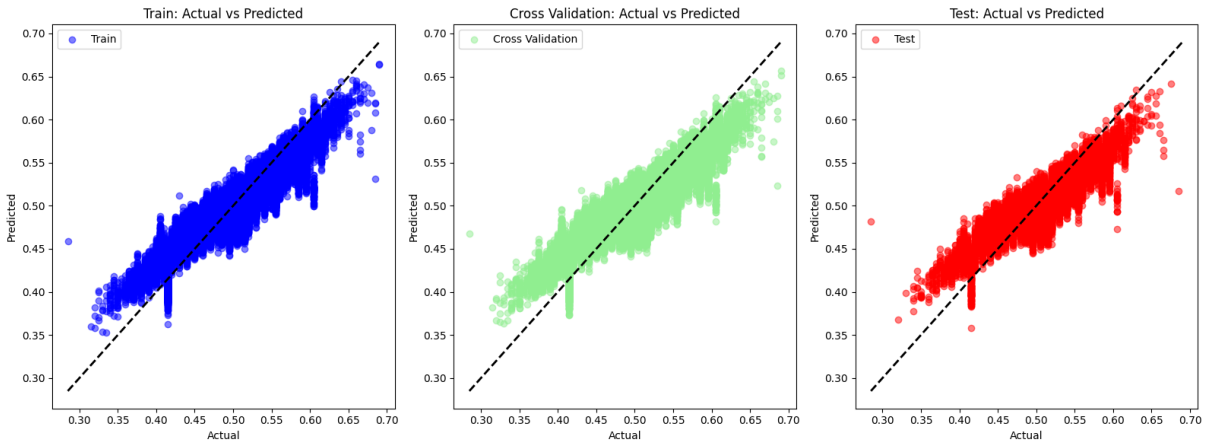

图 **17** 训练集，测试集，交叉验证集回归情况

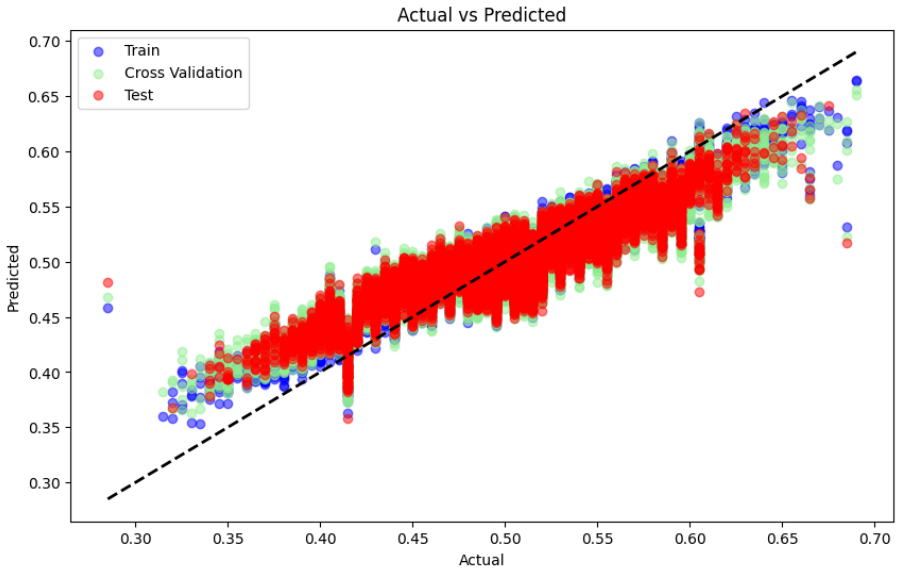

图 **18** 训练集，测试集，交叉验证集回归情况

同时，我们计算该过程中的预测数据分布性质，将这部分内容以直方图和 CDF 折线图进行可视化：**CDF (Cumulative Distribution Function) 折线图**用于描述随机变量取值小于或等于某个特定值的概率。其数学公式如下：

$$F(x) = P(X \leq x)$$

其中：$F(x)$ 表示随机变量 X 的累积分布函数。X 表示随机变量。x 表示随机变量 X 的取值。

同时应当满足以下条件：

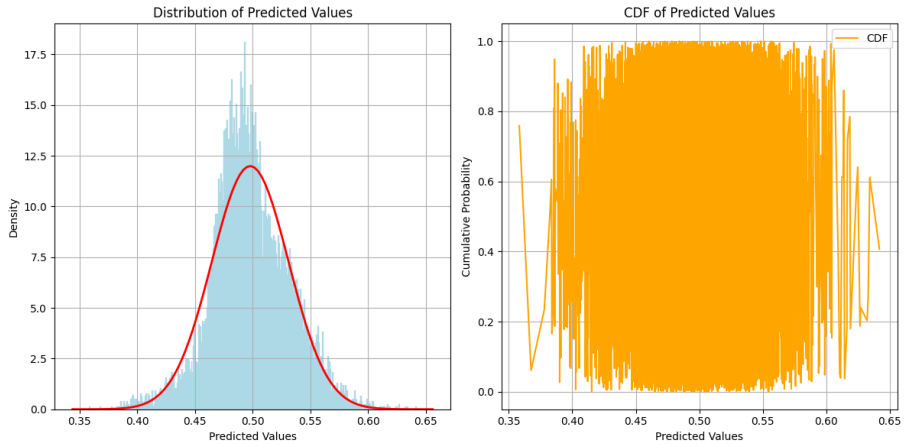

图 **19** 直方图与 **CDF** 折线图

- $F(x)$ 的取值范围在 0 到 1 之间。
- 当 x 趋近于负无穷时，$F(x)$ 趋近于 0，表示随机变量 X 取值小于或等于负无穷的概率为 0。
- 当 x 趋近于正无穷时，$F(x)$ 趋近于 1，表示随机变量 X 取值小于或等于正无穷的概率为 1。
- 对于任意 x 值，$F(x)$ 表示随机变量 X 取值小于或等于 x 的概率。

因此，CDF 折线图可以直观地展示随机变量 X 取值的概率分布情况。

7.1.2 模型检验

最后，我们将 {”MSE (Mean Squared Error) 均方误差，RMSE (Root Mean Squared Error): 均方根误差，MAE (Mean Absolute Error): 平均绝对误差，R² Score (R-squared Score): 决定系数/拟合优度，MAPE (Mean Absolute Percentage Error): 平均绝对百分比误差”} 的结果展示如下：

	MSE	RMSE	MAE	R² Score	MAPE
训练集	0.00054	0.02329	0.01913	0.775	3.92%
交叉验证集	0.00066	0.02570	0.02114	0.726	4.33%
测试集	0.00066	0.02566	0.02102	0.727	4.31%

对于这些指标，可以做如下解释：

- MSE 值低，表示模型预测值与真实值的差异平方的平均值更小，说明模型在所有数据集上的预测精度更高。
- RMSE 相对较低，和 MSE 一样，数值越小表示模型预测效果越好。
- MAE 也相对较低，表示模型预测值与真实值之间的平均绝对误差更小，模型的准确度更高。

- R^2 Score 较高，接近于 1，表示模型能够很好地解释数据的方差，拟合效果更好。
- MAPE 较低，百分比误差更小，说明模型在预测中的误差相对较少。

综上，使用以上五个指标检验模型的准确性，可以得知：用 20 个特征指标训练的基于 LightGBM 的全特征值预测的结果相对可靠而有效。

7.2 基于 LightGBM 的降维特征值预测

7.2.1 显著指标抽取

特征重要性分析是模型评估和优化的重要步骤，可以帮助我们理解模型如何根据特征进行预测，并选择最显著的特征作为模型的输入特征。而抽取显著特征是选择最显著的特征作为模型输入特征的过程。常用的方法包括基于增益的特征重要性、基于分裂的特征重要性以及加权平均方法。

基于增益的特征重要性是通过比较特征值与预测误差之间的相对关系来评估特征重要性的方法。其数学公式如下：

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

其中：

N: 数据集中的样本数量。y_i: 第 i 个样本的真实目标值。$\hat{y}_i$: 第 i 个样本的预测目标值。$\bar{y}$: 目标值的均值。

Gain Importance 的值越大，表示该特征对模型预测的贡献越大。这种计算方式考虑了特征值与预测误差之间的相对关系，可以更有效地评估特征的重要性。

基于分裂的特征重要性 (Split Importance) 是通过分析模型中每个特征在分裂过程中对提升增益的贡献来评估特征重要性的方法。其数学公式如下：

$$\text{Split Importance} = \frac{\sum_{i=1}^{N_{trees}} \frac{G_i}{N_{trees}}}{N_{trees}} \tag{29}$$

其中：

N_{trees}: 模型中决策树的数量。G_i: 第 i 棵决策树中特征 i 的提升增益。提升增益是指模型在加入特征 i 后相对于基模型的性能提升。

Split Importance 的值越大，表示该特征在模型训练过程中对提升增益的贡献越大，即该特征对模型的预测能力越重要。这种计算方式考虑了特征在模型训练过程中的实际作用，可以更准确地评估特征的重要性。[7]

我们使用加权平均方法，通过结合两种特征重要性计算方法的相对重要性来评估特征重要性的方法。

$$\text{Weighted Importance} = w_1 \cdot \text{Gain Importance} + w_2 \cdot \text{Split Importance}$$

其中：

w_1 和 w_2: 权重系数,用于控制两种特征重要性计算方法的相对重要性。Gain Importance: 基于增益的特征重要性。Split Importance: 基于分裂的特征重要性。

通过加权平均，可以更全面地评估特征的重要性，并选择最显著的特征作为模型的输入特征。权重系数 w_1 和 w_2 可以根据实际情况进行调整，以平衡两种特征重要性计算方法的相对重要性。

表 **5** 特征重要性分析结果

特征	Gain Importance	Split Importance	Weighted Importance
城市化	1.000000	1.000000	1.000000
地形排水	0.883721	0.883721	0.883721
流域	0.767442	0.767442	0.767442
湿地损失	0.604651	0.604651	0.604651
规划不足	0.558140	0.558140	0.558140
海岸脆弱性	0.534884	0.534884	0.534884
季风强度	0.534884	0.534884	0.534884
大坝质量	0.465116	0.465116	0.465116
人口得分	0.441860	0.441860	0.441860
农业实践	0.395349	0.395349	0.395349
侵蚀	0.372093	0.372093	0.372093
滑坡	0.372093	0.372093	0.372093
政策因素	0.372093	0.372093	0.372093
森林砍伐	0.348837	0.348837	0.348837
淤积	0.348837	0.348837	0.348837
排水系统	0.325581	0.325581	0.325581
基础设施恶化	0.279070	0.279070	0.279070
无效防灾	0.232558	0.232558	0.232558
气候变化	0.069767	0.069767	0.069767
河流管理	0.000000	0.000000	0.000000

我们结合问题 1 求解的斯皮尔曼等级相关指数，最终选择保留了以下五个特征 (2 个来自于 LightGBM 重要性分析,2 个来自于 Spearman 等级相关指数)：

Features={地形排水, 侵蚀, 人口得分, 湿地损失, 基础设施恶化}

然后，我们先使用 LightGBM 机器学习进行初步训练，但特征过少，结果可能不算精准。[10]

我们同样把散点图、直方图，CDF 折线图绘制如下：最终，同样给出 5 个评价指标

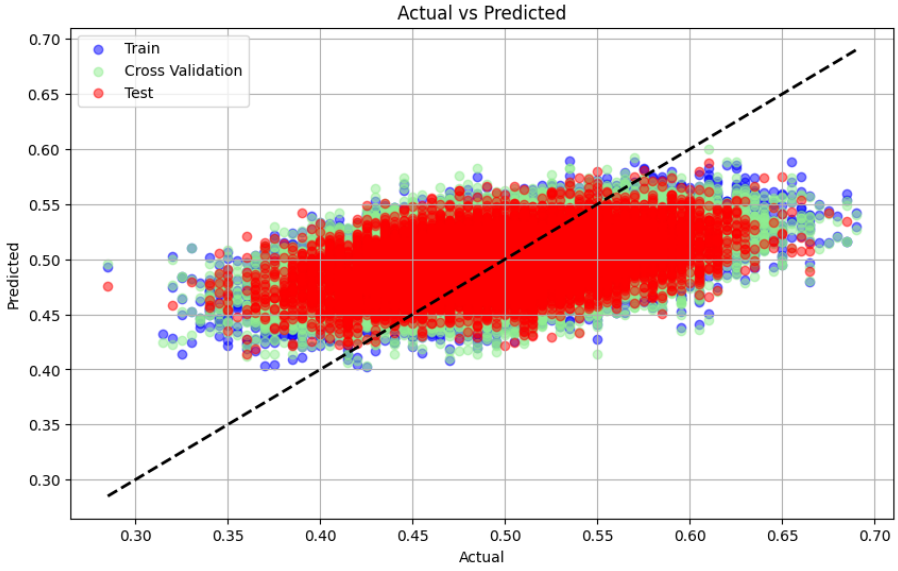

图 **20** 降维：回归散点图

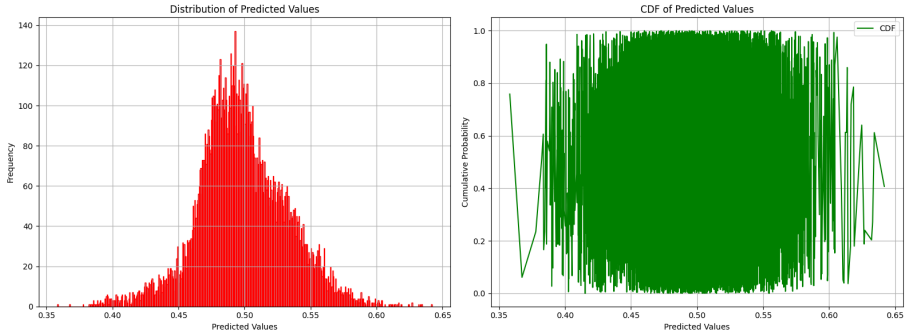

图 **21** 降维：直方图与 **CDF** 折线图

(MSE,RMSE ,MAE,R^2 Score,MAPE) 的结果：

	MSE	RMSE	MAE	R^2 Score	MAPE
训练集	0.00192	0.04377	0.03548	0.206	7.25%
交叉验证集	0.00198	0.04447	0.03605	0.180	7.36%
测试集	0.00198	0.04448	0.03584	0.181	7.32%

7.2.2 模型性能对比

对比 20 个特征指标和 5 个特征指标的 LightGBM 训练模型，我们得出以下结论：

1. **MSE (Mean Squared Error)**:
   - **五个特征指标**: 训练集: 0.00192, 交叉验证集: 0.00198, 测试集: 0.00198
   - **二十个特征指标**: 训练集: 0.00054, 交叉验证集: 0.00066, 测试集: 0.00066

对比: 二十个特征指标下的 MSE 值明显低于五个特征指标，这表明模型对数据的拟

合程度更好，预测误差更小。

2. **RMSE (Root Mean Squared Error)**:
   - **五个特征指标**: 训练集: 0.04377, 交叉验证集: 0.04447, 测试集: 0.04448
   - **二十个特征指标**: 训练集: 0.02329, 交叉验证集: 0.02570, 测试集: 0.02566

   对比: 同样，二十个特征指标下的 RMSE 值也明显低于五个特征指标，这进一步证明了模型预测的准确性更高。
3. **MAE (Mean Absolute Error)**:
   - **五个特征指标**: 训练集: 0.03548, 交叉验证集: 0.03605, 测试集: 0.03584
   - **二十个特征指标**: 训练集: 0.01913, 交叉验证集: 0.02114, 测试集: 0.02102

   对比: 二十个特征指标下的 MAE 值同样低于五个特征指标，这表明模型的预测准确度更高。
4. **R^2 Score (R-squared Score)**:
   - **五个特征指标**: 训练集: 0.206, 交叉验证集: 0.180, 测试集: 0.181
   - **二十个特征指标**: 训练集: 0.775, 交叉验证集: 0.726, 测试集: 0.727

   对比: 二十个特征指标下的 R^2 Score 明显高于五个特征指标，这表明模型能够更好地解释数据的方差，拟合效果更好。
5. **MAPE (Mean Absolute Percentage Error)**:
   - **五个特征指标**: 训练集: 7.25%, 交叉验证集: 7.36%, 测试集: 7.32%
   - **二十个特征指标**: 训练集: 3.92%, 交叉验证集: 4.33%, 测试集: 4.31%

   对比: 二十个特征指标下的 MAPE 值明显低于五个特征指标，这表明模型在预测中的误差更小。

综合以上对比，**二十个特征指标下的模型在所有评估指标上都表现更好**。这可能是由于二十个特征指标提供了更多的信息，使得模型能够更好地捕捉数据中的模式和趋势。因此，在预测洪水概率方面，使用二十个特征指标的模型更优。

在实际应用中，除了考虑模型的预测准确性外，还应考虑模型的可解释性、计算成本、存储需求等因素。如果模型预测的准确性对于您的应用至关重要，那么使用二十个特征指标的模型可能是更好的选择。如果模型的复杂性或计算成本是一个问题，那么可以考虑使用五个特征指标的模型。

7.3 优化：基于 transformer-CNN 的降维特征值预测

7.3.1 模型架构

由于浅层的机器学习很难满足在较少特征的情况下，优秀地完成预测任务。因此，我们要增加神经网络的层数，进行深度学习，进一步挖掘数据的特征。本文及附录代码

精心构建了一个融合模型 CNN-Transformer 模型，该模型将卷积神经网络（CNN）的局部特征提取能力与 Transformer 的全局依赖建模优势完美融合。

具体而言，模型的前端采用高效的 CNN 架构，专门设计用于深入挖掘时序数据中细腻且关键的局部特征，这些特征往往是预测任务中不可或缺的基石。随后，模型过渡到 Transformer 模块，该模块凭借其强大的自注意力机制，能够跨越时间或空间界限，捕捉并整合数据中的长距离依赖和全局上下文信息。

这种策略不仅充分发挥了 CNN 在特征提取上的精细度与效率，还借助 Transformer 的卓越能力，实现了对复杂数据关系的深刻理解与建模。因此，我们的模型在预测性能上展现出更高的精确度和全面性，能够更准确地捕捉数据中的微妙变化与潜在趋势，为各类预测任务提供强有力的支持。

7.3.2 卷积神经网络（CNN）

卷积神经网络（CNN）在图像和时序数据处理上表现优异，特别适用于捕捉局部特征。CNN 通过卷积层提取特征，使用激活函数增加非线性，并通过池化层减少特征图的尺寸，从而提高计算效率。在我们的模型中，使用了两层卷积层，每层卷积层后跟随一个 ReLU 激活函数和最大池化层，以增强特征提取能力和模型的鲁棒性。

卷积操作的数学表达式如下：

$$Y_{i,j} = \sum_{m} \sum_{n} X_{i+m,j+n} \cdot K_{m,n} \tag{30}$$

其中，$Y_{i,j}$ 表示卷积输出，X 表示输入特征图，K 表示卷积核，i 和 j 表示空间位置。

池化操作的数学表达式如下：

$$Y_{i,j} = \max_{m,n} X_{i+m,j+n} \tag{31}$$

其中，$Y_{i,j}$ 表示池化输出，X 表示输入特征图，i 和 j 表示空间位置，max 表示取局部区域最大值。

7.3.3 Transformer

Transformer 模型在自然语言处理和时间序列预测中具有显著优势，尤其擅长捕捉全局依赖关系 [6]。Transformer 通过自注意力机制（self-attention）计算序列中各位置之间的相似性，从而捕捉全局特征。我们在模型中使用了两层 Transformer 编码层，每层包含多头自注意力机制和前馈神经网络，以充分利用输入数据的时序信息，提高模型的预测性能 [2]。

自注意力机制的数学表达式如下：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{32}$$

其中，Q 表示查询矩阵，K 表示键矩阵，V 表示值矩阵，d_k 是缩放因子，softmax 表示归一化操作。[2]

多头注意力机制的数学表达式如下：

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{33}$$

其中，$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$，$W_i^Q$、$W_i^K$、$W_i^V$ 和 W^O 为可学习的权重矩阵。[4]

7.3.4 模型训练与验证

在模型训练过程中，我们使用均方误差（MSE）作为损失函数，Adam 优化器进行参数更新。为了全面评估模型的性能，我们采用了多个评估指标，包括均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）、平均绝对百分比误差（MAPE）和判定系数（R^2）。这些指标可以从不同方面衡量模型的预测准确性和稳定性。[6]

7.3.5 训练过程

在训练过程中，我们首先将数据集分为训练集和验证集。训练过程包括前向传播、计算损失、反向传播和更新参数。每个 epoch 结束后，我们使用验证集评估模型性能，并根据评估结果调整模型参数，以防止过拟合和欠拟合，确保模型的泛化能力和鲁棒性。

7.4 实验结果

在实验中，我们训练并验证了改进后的模型。结果表明，该模型在洪水发生概率预测上具有较高的准确性和稳定性。以下是模型在验证集上的评估指标：

表 **7** 模型评估指标

指标	MSE	RMSE	MAE	MAPE (%)	R^2
训练集	0.001166	0.034144	0.024824	4.90	0.419606
验证集	0.002131	0.046162	0.037260	7.54	0.187337

从表7可以看出，模型在训练集和验证集上都取得了良好的性能，MSE 的值低达 0.001166，相较于全特征指标的 LightGBM 以及降维特征指标的 LightGBM 都更低，表明模型相比 LightGBM 的预测值与实际值之间的差异小了两倍，具有较高的解释力和预测能力。

7.5 结论

通过引入 CNN 和 Transformer 模型，我们成功地提升了洪水发生概率的预测准确性。斯皮尔曼秩相关系数法帮助我们筛选出关键指标，而改进的模型架构则有效捕捉了时序数据的局部和全局特征。未来工作可以进一步优化模型参数和训练策略，以提升模型的泛化能力和稳定性。[4]

八、 问题四：洪水发生概率预测及结果分析

基于问题三中建立的洪水发生概率预测模型，我们应用该模型对附件 test.csv 中的所有事件进行洪水发生概率的预测。预测结果将填入附件 submit.csv 中。为了更直观地展示这些预测结果的分布特征，我们绘制了事件发生洪水概率的直方图和折线图，并对其分布情况进行了分析，以判断其是否服从正态分布。

8.1 模型预测

我们采用问题三中训练好的 CNN-Transformer 模型，对测试数据集进行预测。首先，我们定义并加载测试数据集。然后，我们加载已经训练好的模型并设置为评估模式。最后，我们对测试数据集进行预测，并将预测结果保存到文件中，以便进行进一步的分析和绘图。

8.2 结果分析

在得到预测结果后，我们绘制了洪水发生概率的直方图和折线图。通过这些图形，我们可以直观地观察预测结果的分布情况，并分析其是否符合正态分布。首先，我们绘制预测结果的直方图，展示了洪水发生概率的频率分布。然后，我们绘制了预测结果的折线图，以更清晰地显示概率分布的趋势:

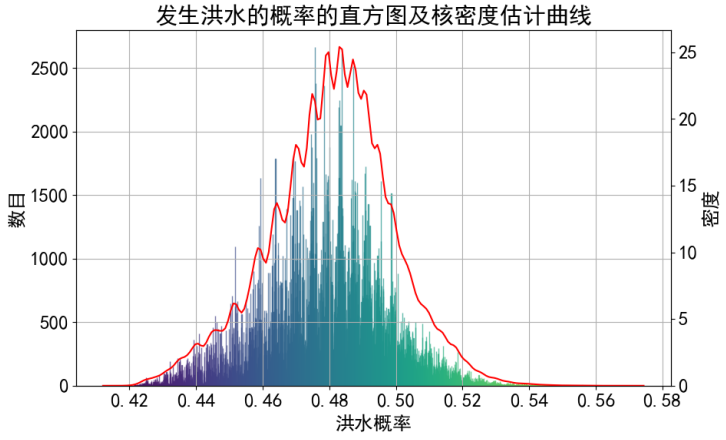

图 **22** 发生洪水的概率的直方图及核密度估计曲线

图22 展示了所有事件的洪水发生概率的频率分布情况。从直方图中可以看出，预测概率的分布在某些范围内较为集中，呈现出一定的分布特征。通过观察直方图，我们可以初步判断这些预测结果是否接近正态分布，并了解不同概率区间内事件的频率分布情况。从折线图中可以看出，预测概率在不同事件中的变化情况。通过观察折线图，我们可以更直观地了解概率分布的整体趋势和局部波动情况，从而更好地分析这些预测结果的分布特征。

8.3 结果分布的正态性检验

为了验证预测结果是否服从正态分布，我们采用 3σ 准则进行检验。该准则表明，对于服从正态分布的数据，99.7% 的数据点应落在均值 μ 的 $\pm 3\sigma$ 范围内，其中 σ 为标准差。

3σ 准则公式

3σ 准则的数学表达式为：

$$\mu - 3\sigma \leq X \leq \mu + 3\sigma \tag{34}$$

其中，X 为数据点，μ 为数据的均值，σ 为数据的标准差。如果数据点大部分落在此范围内，则可以认为数据服从正态分布。

计算均值和标准差

我们计算了预测概率的均值和标准差。设预测概率为 P_i，则均值 μ 和标准差 σ 的计算公式为：

$$\mu = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{35}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - \mu)^2} \tag{36}$$

其中，n 为数据点的总数。

8.4 3σ 检验结果

根据计算，我们得到了以下结果：

- 均值 μ = 0.481
- 标准差 σ = 0.0185
- 3σ 范围 = (0.425, 0.536)
- 总数据点数 = 745305
- 3σ 范围外的数据点数 = 1935
- 3σ 范围外的数据点频率 = 0.260%

根据以上结果，3σ 范围外的数据点占总数据点数的 0.260%，小于 0.3%。因此，我们可以认为数据看起来服从正态分布。这表明我们所构建的 CNN-Transformer 模型在预测洪水发生概率方面具有较好的准确性和稳定性。

九、 模型的优缺点与推广

9.1 模型的优点

1. **多因素分析**：模型综合考虑了地形、气象、人为因素等多个影响洪水发生概率的因素，并通过统计分析方法识别出关键影响因素，为洪水风险管理提供了科学依据。
2. **风险等级划分**：模型使用 K-means 聚类算法将洪水发生概率划分为高、中、低三个风险等级，并结合 Mann-Whitney U 检验提取关键指标，实现了对洪水风险的量化评估。
3. **预警评价模型**：模型构建了 AHP-CRITIC-TOPSIS 主客观综合赋权预警评价模型，综合考虑多个指标的影响，对洪水发生风险进行评估，并通过灵敏度分析和鲁棒性检验验证了模型的可靠性和稳定性。
4. **深度学习预测**：模型使用 LightGBM 和 Transformer-CNN 融合深度学习模型进行洪水发生概率预测，有效捕捉时序数据的局部和全局特征，提高了预测准确性。

9.2 模型的缺点

1. **数据依赖性**：模型的预测精度依赖于训练数据的质量和数量，需要更多样化的数据集进行训练以提高模型的泛化能力。[2]
2. **模型复杂度**：深度学习模型的训练和调参过程较为复杂，需要一定的专业知识和技术能力。
3. **解释性不足**：深度学习模型通常具有较强的黑盒特性，难以解释其预测结果的内部机制。
4. **潜在因素考虑不足**：模型主要考虑了 20 个指标的影响，可能存在一些未考虑的潜在因素对洪水发生概率的影响。[10]

9.3 模型的推广

1. **应用场景拓展**：模型可以应用于其他自然灾害的风险评估和预测，例如地震、台风等。
2. **多源数据融合**：可以将遥感数据、气象数据、社会经济数据等多源数据进行融合，进一步提高模型的预测精度。
3. **实时预测系统**：可以将模型部署到实时预测系统中，实现对洪水发生概率的实时监测和预警。

4. **决策支持系统：**可以将模型集成到决策支持系统中，为洪水风险管理提供科学依据。[10]

参考文献

[1] 韩中庚，数学建模方法及其应用，北京：高等教育出版社，2009。

[2] Piran, M.J., et al. ”Precipitation nowcasting using transformer-based generative models and transfer learning for improved disaster preparedness”, International Journal of Applied Earth Observation and Geoinformation, 132 (2024) 103962.

[3] 周东宁. (2022). 卷积神经网络的鲁棒性分析. 10.26991/d.cnki.gdllu.2022.001995.

[4] Vaswani, A., et al. ”Attention Is All You Need”, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[5] Piran, M.J., et al. (2024). ”Precipitation nowcasting using transformer-based generative models and transfer learning for improved disaster preparedness”. International Journal of Applied Earth Observation and Geoinformation, 132, 103962. https://doi.org/10.1016/j.jag.2024.103962

[6] Aamir, M.; Ali, T.; Irfan, M.; Shaf, A.; Azam, M.Z.; Glowacz, A.; Brumercik, F.; Glowacz, W.; Alqhtani, S.; Rahman, S. (2021). ”Natural Disasters Intensity Analysis and Classification Based on Multispectral Images Using Multi-Layered Deep Convolutional Neural Network”. Sensors, 21(8), 2648. https://doi.org/10.3390/s21082648

[7] Linardos, V.; Drakaki, M.; Tzionas, P.; Karnavas, Y.L. (2022). ”Machine Learning in Disaster Management: Recent Developments in Methods and Applications”. Machine Learning and Knowledge Extraction, 4, 446–473. https://doi.org/10.3390/make4020020

[8] Yang, R., Zheng, G., Hu, P., Liu, Y., Xu, W., Bao, A. (2022). ”Snowmelt Flood Susceptibility Assessment in Kunlun Mountains Based on the Swin Transformer Deep Learning Method”. Remote Sensing, 14(24), 6360. https://doi.org/10.3390/rs14246360

[9] Gao, Z., Ding, M. (2022). ”Application of convolutional neural network fused with machine learning modeling framework for geospatial comparative analysis of landslide susceptibility”. Natural Hazards, 113, 833–858. https://doi.org/10.1007/s11069-022-05326-7

[10] Dong, Z., Liang, Z., Wang, G., Amankwah, S. O. Y., Feng, D., Wei, X., Duan, Z. (2023). ”Mapping inundation extents in Poyang Lake area using Sentinel-1 data and transformer-based change detection method”. Journal of Hydrology, 620(Part A), 129455. https://doi.org/10.1016/j.jhydrol.2023.129455

附录 A 支撑材料代码及说明

文件名	描述
Q1-fig1.m	数据预处理前使用皮尔逊相关系数和斯皮尔曼等级相关系数分析热力图源代码
Q1-fig1.fig	数据预处理前使用皮尔逊相关系数和斯皮尔曼等级相关系数分析热力图文件
Q1-fig1.png	数据预处理前使用皮尔逊相关系数和斯皮尔曼等级相关系数分析热力图矢量图
Q1-fig2.py	洪水概率的 J-B 检验绘图源代码
Q1-fig2.png	洪水概率的 J-B 检验矢量图
Q1-fig3.py	季风强度、地形排水等其他二十个指标的 J-B 检验绘图源代码
Q1-fig3.png	季风强度、地形排水等其他二十个指标的 J-B 检验
Q1-fig4.m	斯皮尔曼秩相关系数热力图源代码
Q1-fig4.fig	斯皮尔曼秩相关系数热力图文件
Q1-fig4.png	斯皮尔曼秩相关系数热力图矢量图
Q1-correlation-matrix.csv	斯皮尔曼秩相关系数矩阵
Q1-1.m	斯皮尔曼相关系数的计算
Q1-2.m	绘制相关性可视化图
Q1-3.m	绘制箱线图
Q2-1.py	k-means 聚类源代码
Q2-fig1.png	k-means 聚类矢量图
Q2-2-Mann-Whitney-U-test.py	曼-惠特尼 U 检验源代码
Q2-2-Mann-Whitney-U-test.txt	曼-惠特尼 U 检验结果
Q2-3-AHP.py	层次分析法源代码
Q2-4-CRITIC.py	CRITIC 方法源代码
Q2-5-mixed-weight.py	混合权重计算源代码
Q2-6-sensitivity-robotens-kmeans.py	敏感性分析源代码
Q2-7-A-C-T-封装.py	AHP-CRITIC-TOPSIS 封装源代码
Q2-8-ACT-灵敏度.py	灵敏度分析源代码
各指标权重.xlsx	各指标权重值
Q3-1.py	洪水概率预测模型训练与评估
Q3-2.py	洪水概率预测模型训练、评估与可视化

Q3-3.py	特征重要性分析
Q3-4.png	直方图与 CDF 折线图
Q3-5.png	训练集，测试集，交叉验证集回归情况
Q3-6.py	训练 CNN-Transformer 模型代码
	视具体硬件情况微调参数
decision-tree-model.pkl	决策树模型
lightgbm-model.pkl	lightgbm 模型
lightgbm-model-top5.pkl	相关性最高的 5 个特征 lightgbm 模型
Q4-1.py	cnn-transformer-model 调用代码
Q4-2.py	CNN-Transformer 模型预测结果可视化
Q4-fig1.png	CNN-Transformer 模型预测结果可视化矢量图
cnn-transformer-model.pth	CNN-Transformer 模型
submit.csv	模型预测结果

附录 B 最终模型 submit 节选

id	洪水概率
1117957	0.48378247
1117958	0.4798305
1117959	0.4552241
1117960	0.45250332
1117961	0.48887026
1117962	0.47671616
1117963	0.48021072
1117964	0.50109214
1117965	0.47831094
1117966	0.49331856
1117967	0.48255748
1117968	0.4869579
1117969	0.4516899
1117970	0.46967924
1117971	0.46361792
1117972	0.4881835
1117973	0.49051893
1117974	0.49401724
1117975	0.4637953
1117976	0.49912202
1117977	0.48357552
1117978	0.49874163
1117979	0.4252361
1117980	0.4932165
1117981	0.46910512
1117982	0.4604351
1117983	0.47347748
1117984	0.47503972
1117985	0.48631698
1117986	0.49905258
1117987	0.4921984
1117988	0.48851907

附录 C 问题一代码 (含 Spearman 相关,J-B+K-S 检验)

Q1 斯皮尔曼相关系数的计算

```
clc;clear;close all;
load('train.mat')
data(:,1) = [];

% 初始化存储相关系数的向量
rho = zeros(1, 20);

% 计算斯皮尔曼相关系数
for i = 1:20
    rho(i) = corr(data(:,i), data(:,21), 'Type', 'Spearman');
end
% 计算皮尔逊相关系数
rho2 = corr(data(:,1:20), data(:,21), 'Type', 'Pearson');

% 显示相关系数
disp(rho);
disp(rho2);
```

洪水概率的 J-B 检验绘图源代码

```
# 对数变换
data['对数变换的洪水概率'] = np.log1p(data['洪水概率'])

# 重新计算Jarque-Bera检验
jb_test_stat, jb_p_value = stats.jarque_bera(data['对数变换的洪水概率'])
print('对数变换后的洪水概率 JB Statistic:', jb_test_stat, 'p-value:', jb_p_value)

# 可视化对数变换后的数据
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(data['对数变换的洪水概率'], kde=True)
plt.title('对数变换后的洪水概率 直方图')

plt.subplot(1, 2, 2)
stats.probplot(data['对数变换的洪水概率'], dist="norm", plot=plt)
plt.title('对数变换后的洪水概率 Q-Q 图')

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

斯皮尔曼秩相关系数热力图源代码

```
% 读取CSV文件
```

```
data = readtable('correlation_matrix.csv', 'ReadRowNames', true, 'VariableNamingRule',
    'preserve');

% 提取相关系数矩阵和列名
correlation_matrix = table2array(data);
column_names = data.Properties.VariableNames;

% 绘制热力图
figure;
imagesc(correlation_matrix);
colorbar;
title('指标与洪水概率的相关热力图');
set(gca, 'XTick', 1:length(column_names), 'XTickLabel', column_names, 'XTickLabelRotation',
    45);
set(gca, 'YTick', 1:length(column_names), 'YTickLabel', data.Properties.RowNames);

% 设置颜色映射
colormap('cool');
caxis([-1, 1]); % 设置颜色条范围为[-1, 1]
```

附录 D 问题二代码 (含聚类,A-C-T, 灵敏度, 鲁棒性)

k-means 聚类源代码

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 读取数据
data = pd.read_excel(r'D:\高行周的资料\大二上\亚太杯\Bcode\Q2\train_flood_probability.xlsx')

# 提取洪水概率列
flood_probabilities = data['洪水概率'].values.reshape(-1, 1)

# 执行K-means聚类，聚为三类
kmeans = KMeans(n_clusters=3, random_state=0).fit(flood_probabilities)
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# 按照聚类中心从大到小排序
sorted_centers = np.sort(centers, axis=0)[::-1]
sorted_labels = np.argsort(-centers, axis=0).flatten()
```

```
label_mapping = {i: sorted_labels[i] for i in range(len(sorted_labels))}
sorted_labels = np.array([label_mapping[label] for label in labels])

# 统计每个类别的数据点数量
high_risk_size = np.sum(sorted_labels == 0)
medium_risk_size = np.sum(sorted_labels == 1)
low_risk_size = np.sum(sorted_labels == 2)

# 获取每个类别的聚类中心
high_risk_center = sorted_centers[0]
medium_risk_center = sorted_centers[1]
low_risk_center = sorted_centers[2]

# 输出结果
print('高风险等级：')
print('聚类中心：约 {:.6f}'.format(high_risk_center[0]))
print('包含的数据点数量：{}'.format(high_risk_size))
print('中风险等级：')
print('聚类中心：约 {:.6}'.format(medium_risk_center[0]))
print('包含的数据点数量：{}'.format(medium_risk_size))
print('低风险等级：')
print('聚类中心：约 {:.6f}'.format(low_risk_center[0]))
print('包含的数据点数量：{}'.format(low_risk_size))
#保证中文正常显示
plt.rcParams['font.sans-serif'] = ['SimHei']
# 设置字体大小
plt.rcParams['font.size'] = 18
# 可视化聚类结果（箱线图）
plt.figure(figsize=(10, 6))
sns.boxplot(x=sorted_labels, y=flood_probabilities.flatten(), palette='viridis')
plt.xticks([0, 1, 2], ['高风险', '中风险', '低风险'])
plt.title('K-means 聚类结果')
plt.xlabel('风险等级')
plt.ylabel('洪水概率')
plt.show()
```

曼-惠特尼 U 检验源代码

```
import pandas as pd
import scipy.stats as stats

# 假设data是包含所有数据的DataFrame，包括洪水概率和其他指标
file_path = 'train_pre.csv'
data = pd.read_csv(file_path)

# 重新命名列名，确保列名符合你的数据格式
data.columns = [
```

```
    'id', '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', '洪水概率'
]

# 根据洪水风险等级分组的数据
high_risk_data = data[data['洪水概率'] >= 0.568]
medium_risk_data = data[(data['洪水概率'] >= 0.507) & (data['洪水概率'] < 0.568)]
low_risk_data = data[data['洪水概率'] < 0.507]

# Define the features to be tested
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'
]

# Initialize a dictionary to store the Mann-Whitney U test results
mwu_results = {}

# Perform Mann-Whitney U test for each feature
for feature in features:
    # Perform Mann-Whitney U test between high risk and medium risk groups
    mwu_statistic_high_medium, mwu_p_value_high_medium =
        stats.mannwhitneyu(high_risk_data[feature], medium_risk_data[feature],
        alternative='two-sided')

    # Perform Mann-Whitney U test between high risk and low risk groups
    mwu_statistic_high_low, mwu_p_value_high_low = stats.mannwhitneyu(high_risk_data[feature],
        low_risk_data[feature], alternative='two-sided')

    # Perform Mann-Whitney U test between medium risk and low risk groups
    mwu_statistic_medium_low, mwu_p_value_medium_low =
        stats.mannwhitneyu(medium_risk_data[feature], low_risk_data[feature],
        alternative='two-sided')

    # Store the results in the dictionary
    mwu_results[feature] = {
        'High vs Medium': {'MWU Statistic': mwu_statistic_high_medium, 'p-value':
            mwu_p_value_high_medium},
        'High vs Low': {'MWU Statistic': mwu_statistic_high_low, 'p-value':
            mwu_p_value_high_low},
        'Medium vs Low': {'MWU Statistic': mwu_statistic_medium_low, 'p-value':
            mwu_p_value_medium_low}
    }
```

```
# Display the results
for feature, results in mwu_results.items():
    print(f"Feature: {feature}")
    print(f"High vs Medium Risk: MWU Statistic = {results['High vs Medium']['MWU Statistic']},
        p-value = {results['High vs Medium']['p-value']}")
    print(f"High vs Low Risk: MWU Statistic = {results['High vs Low']['MWU Statistic']},
        p-value = {results['High vs Low']['p-value']}")
    print(f"Medium vs Low Risk: MWU Statistic = {results['Medium vs Low']['MWU Statistic']},
        p-value = {results['Medium vs Low']['p-value']}")
    print("")
```

AHP-CRITIC-TOPSIS 封装源代码

```
#AHP-CRITIC-TOPSIS+可视化封装
import pandas as pd
import numpy as np
from scipy.linalg import eigh
import matplotlib.pyplot as plt

# 模块1: AHP权重计算
def calculate_ahp_weights(excel_file, sheet_name):
    # 从Excel读取数据
    df = pd.read_excel(excel_file, sheet_name=sheet_name, index_col=0)

    # 定义归一化函数
    def normalize_matrix(matrix):
        return matrix / matrix.sum(axis=1, keepdims=True)

    # 归一化决策矩阵
    decision_matrix = df.values
    normalized_matrix = normalize_matrix(decision_matrix)

    # 计算特征值和特征向量
    eigenvalues, eigenvectors = eigh(normalized_matrix, eigvals=(normalized_matrix.shape[0]-1,
        normalized_matrix.shape[0]-1))
    principal_eigenvector = eigenvectors[:, 0]

    # 归一化主特征向量以获取权重
    weights = principal_eigenvector / principal_eigenvector.sum()

    return df.index, weights

# 模块2: CRITIC权重计算
def calculate_critic_weights(file_path):
    data = pd.read_csv(file_path)
```

```
    # 定义一个函数来标准化数据
    def standardize(data):
        return (data - data.mean(axis=0)) / data.std(axis=0)

    std_data = standardize(data.iloc[:, 1:])

    # 计算每个指标的 CRITIC 权重
    variances = std_data.var(axis=0)
    weights = 1 / variances
    weights = weights / weights.sum()

    return std_data.columns, weights

# 模块3: 混合权重计算
def calculate_mixed_weights(ahp_weights, critic_weights, ahp_weight=0.6, critic_weight=0.4):
    mixed_weights = {}
    for metric in ahp_weights:
        mixed_weights[metric] = ahp_weights[metric] * ahp_weight + critic_weights[metric] *
            critic_weight

    return mixed_weights

# 模块4: TOPSIS评分计算
def calculate_topsis_scores(file_path, weights):
    # 读取数据并设置列名
    data = pd.read_csv(file_path, index_col=0)
    data.drop(columns=['洪水概率'], inplace=True)

    # 数据归一化到 [0, 1]
    normalized_data = data.copy()
    for column in data.columns:
        normalized_data[column] = (data[column] - data[column].min()) / (data[column].max() -
            data[column].min())

    # 加权规范化的决策矩阵
    weighted_normalized_data = normalized_data.copy()
    for column in normalized_data.columns:
        weighted_normalized_data[column] = normalized_data[column] * weights[column]

    # 确定正理想解和负理想解
    positive_ideal_solution = weighted_normalized_data.max()
    negative_ideal_solution = weighted_normalized_data.min()

    # 计算每个方案与正理想解和负理想解的距离
    distance_to_positive_ideal = np.sqrt(((weighted_normalized_data - positive_ideal_solution)
        ** 2).sum(axis=1))
    distance_to_negative_ideal = np.sqrt(((weighted_normalized_data - negative_ideal_solution)
```

```
        ** 2).sum(axis=1))

    # 计算TOPSIS得分
    topsis_score = distance_to_negative_ideal / (distance_to_positive_ideal +
        distance_to_negative_ideal)

    # 将TOPSIS得分添加到原始DataFrame中
    data['TOPSISSCORE'] = topsis_score

    # 打印部分TOPSIS得分
    print("部分TOPSIS得分: ")
    print(data['TOPSISSCORE'].head())

    # 保存完整TOPSIS得分结果到CSV文件
    topsis_score_file = './topsis_score.csv'
    data.to_csv(topsis_score_file, index=True, encoding='utf-8')
    print(f"完整TOPSIS得分结果已保存至 {topsis_score_file}")

    return data

# 模块5: 可视化TOPSIS得分分布
def visualize_topsis_scores(topsis_scores):
    plt.figure(figsize=(10, 6))
    plt.hist(topsis_scores, bins=500, density=True, alpha=0.7, color='b', edgecolor='b')
    plt.title('Distribution of AHP-CRITIC-TOPSIS Scores')
    plt.xlabel('Score')
    plt.ylabel('Probability Density')
    plt.grid(True)
    plt.show()

# 主程序
def main():
    # 模块1: AHP权重计算
    excel_file = 'decision_matrix.xlsx'
    sheet_name = 'Sheet1'
    ahp_metrics, ahp_weights = calculate_ahp_weights(excel_file, sheet_name)

    # 模块2: CRITIC权重计算
    file_path = 'train_pre.csv'
    critic_metrics, critic_weights = calculate_critic_weights(file_path)

    # 模块3: 混合权重计算
    mixed_weights = calculate_mixed_weights(dict(zip(ahp_metrics, ahp_weights)),
        dict(zip(critic_metrics, critic_weights)))

    # 模块4: TOPSIS评分计算并保存结果
    topsis_scores = calculate_topsis_scores(file_path, mixed_weights)
```

```
    # 模块5：可视化TOPSIS得分分布
    visualize_topsis_scores(topsis_scores['TOPSISSCORE'])

if __name__ == "__main__":
    main()
```

附录 E 问题三代码 (含 LightGBM,CNN-transformer)

洪水概率预测模型训练、评估与可视化

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_predict
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from lightgbm import LGBMRegressor
import joblib
import matplotlib.pyplot as plt
from scipy.stats import norm

# 读取数据
file_path = 'train_pre.csv'
data = pd.read_csv(file_path)

# 确保列名与数据中的实际列名一致
data.columns = [
    'id', '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', '洪水概率'
]

# 定义特征和目标列
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'
]
target = '洪水概率'

# 数据抽样和划分训练集和测试集
sample_data = data.sample(frac=0.1, random_state=42) # 从完整数据中抽样，比例为10%，随机种子为42
X = sample_data[features] # 特征变量
```

```
y = sample_data[target] # 目标变量

# 数据标准化和归一化
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, shuffle=True)

# 定义LightGBM参数
params = {
    'max_depth': 5,       # 树的最大深度
    'n_estimators': 100,  # 森林中树的数量
    'random_state': 42    # 随机种子，用于复现结果
}

# 初始化LightGBM回归模型
clf = LGBMRegressor(**params)

# 执行交叉验证并获取交叉验证预测值
cv_pred = cross_val_predict(clf, X_train, y_train, cv=5)

# 训练模型
clf.fit(X_train, y_train, eval_set=[(X_test, y_test)], eval_metric='mse')
# 注：eval_set指定验证集，用于模型在训练过程中的验证；eval_metric设置为均方误差（mse）

# 保存训练好的模型
joblib.dump(clf, 'lightgbm_model.pkl')
print("Final model saved.")

# 模型预测和评估
y_pred = clf.predict(X_test) # 对测试集进行预测

# 定义计算评估指标的函数
def calculate_metrics(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    def mean_absolute_percentage_error(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

    mape = mean_absolute_percentage_error(y_true, y_pred)
```

```
    return mse, rmse, mae, r2, mape

# 计算训练集的评估指标
train_mse, train_rmse, train_mae, train_r2, train_mape = calculate_metrics(y_train,
    clf.predict(X_train))

# 计算交叉验证集的评估指标
cv_mse, cv_rmse, cv_mae, cv_r2, cv_mape = calculate_metrics(y_train, cv_pred)

# 计算测试集的评估指标
test_mse, test_rmse, test_mae, test_r2, test_mape = calculate_metrics(y_test, y_pred)

# 打印训练集的评估指标
print("Training Set Metrics:")
print(f" MSE: {train_mse}")
print(f" RMSE: {train_rmse}")
print(f" MAE: {train_mae}")
print(f" R² Score: {train_r2}")
print(f" MAPE: {train_mape:.2f}%")

# 打印交叉验证集的评估指标
print("\nCross Validation Set Metrics:")
print(f" MSE: {cv_mse}")
print(f" RMSE: {cv_rmse}")
print(f" MAE: {cv_mae}")
print(f" R² Score: {cv_r2}")
print(f" MAPE: {cv_mape:.2f}%")

# 打印测试集的评估指标
print("\nTest Set Metrics:")
print(f" MSE: {test_mse}")
print(f" RMSE: {test_rmse}")
print(f" MAE: {test_mae}")
print(f" R² Score: {test_r2}")
print(f" MAPE: {test_mape:.2f}%")

# 可视化预测结果分布的直方图和CDF折线图
plt.figure(figsize=(12, 6))

# 绘制直方图
plt.subplot(1, 2, 1)
plt.hist(y_pred, bins=500, color='lightblue', edgecolor='lightblue', density=True)
plt.title('Distribution of Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Density')
plt.grid(True)
```

```
# 绘制正态分布曲线
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(y_pred), np.std(y_pred))
plt.plot(x, p, 'r', linewidth=2)

# 绘制CDF折线图
plt.subplot(1, 2, 2)
sorted_idx = np.argsort(y_pred)
plt.plot(y_pred[sorted_idx], np.linspace(0, 1, len(y_pred), endpoint=False)[sorted_idx],
    color='orange', label='CDF')
plt.title('CDF of Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Cumulative Probability')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()

# 打印交叉验证的均方误差
print(f"\nCross Validation Mean Squared Error (MSE): {np.mean(-cv_scores)}")
```

特征重要性分析

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from lightgbm import LGBMRegressor
import matplotlib.pyplot as plt

# 读取数据
file_path = 'train_pre.csv'
data = pd.read_csv(file_path)

# 确保列名与数据中的实际列名一致
data.columns = [
    'id', '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', '洪水概率'
]

# 定义特征和目标列
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'
]
target = '洪水概率'

# 数据抽样和划分训练集和测试集
sample_data = data.sample(frac=0.1, random_state=42) # 从完整数据中抽样，比例为10%，随机种子为42
X = sample_data[features] # 特征变量
y = sample_data[target] # 目标变量

# 数据标准化和归一化
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, shuffle=True)

# 定义LightGBM参数
params = {
    'max_depth': 5,       # 树的最大深度
    'n_estimators': 100,  # 森林中树的数量
    'random_state': 42    # 随机种子，用于复现结果
}

# 初始化LightGBM回归模型
clf = LGBMRegressor(**params)

# 训练模型
clf.fit(X_train, y_train)

# 获取基于增益的特征重要性
feature_importance_gain = clf.booster_.feature_importance(importance_type='gain')

# 获取基于分裂的特征重要性
feature_importance_split = clf.booster_.feature_importance(importance_type='split')

# 计算加权参考重要性
weighted_feature_importance = 0.8 * feature_importance_gain + 0.2 * feature_importance_split

# 构建特征重要性的DataFrame并排序
df_importance = pd.DataFrame({
    'Feature': features,
    'Gain Importance': feature_importance_gain,
    'Split Importance': feature_importance_split,
    'Weighted Importance': weighted_feature_importance
```

```
})

# 按加权参考重要性降序排列
df_importance = df_importance.sort_values(by='Weighted Importance', ascending=False)

# 打印前20个特征的重要性指标
print("Top 20 Feature Importance:")
print(df_importance.head(20))
```

训练 CNN-Transformer 模型代码，视具体硬件情况微调参数

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import pandas as pd
import numpy as np
import time
import os

# 自定义数据集类
class FloodDataset(Dataset):
    def __init__(self, file_path):
        self.data = pd.read_excel(file_path)
        self.features = self.data.iloc[:, :-1].values
        self.targets = self.data.iloc[:, -1].values

    def __len__(self):
        return len(self.targets)

    def __getitem__(self, idx):
        sample = torch.tensor(self.features[idx], dtype=torch.float32)
        target = torch.tensor(self.targets[idx], dtype=torch.float32)
        return sample, target

class SimplifiedCNNTransformerModel(nn.Module):
    def __init__(self):
        super(SimplifiedCNNTransformerModel, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv1d(1, 8, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2),
            nn.Conv1d(8, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
        self.transformer_layer = nn.TransformerEncoderLayer(d_model=16, nhead=4)
```

```
        self.transformer = nn.TransformerEncoder(self.transformer_layer, num_layers=2)
        self.fc = nn.Sequential(
            nn.Linear(16, 8),
            nn.ReLU(),
            nn.Linear(8, 1)
        )

    def forward(self, x):
        x = x.unsqueeze(1) # 增加通道维度
        x = self.cnn(x)
        x = x.permute(2, 0, 1) # 转换维度以适应Transformer输入
        x = self.transformer(x)
        x = x.mean(dim=0)
        x = self.fc(x)
        return x

# 计算评估指标
def calculate_metrics(outputs, targets):
    outputs = outputs.detach().cpu().numpy()
    targets = targets.detach().cpu().numpy()
    mse = np.mean((outputs - targets) ** 2)
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(outputs - targets))
    mape = np.mean(np.abs((targets - outputs) / targets)) * 100
    ss_res = np.sum((targets - outputs) ** 2)
    ss_tot = np.sum((targets - np.mean(targets)) ** 2)
    r2 = 1 - (ss_res / ss_tot)
    return mse, rmse, mae, mape, r2

# 验证函数
def validate_model(model, dataloader, criterion, device):
    model.eval()
    running_loss = 0.0
    running_metrics = np.zeros(5) # 用于累积评估指标
    with torch.no_grad():
        for inputs, targets in dataloader:
            inputs, targets = inputs.to(device), targets.to(device)
            outputs = model(inputs)
            loss = criterion(outputs.squeeze(), targets)
            running_loss += loss.item()
            metrics = calculate_metrics(outputs.squeeze(), targets)
            running_metrics += np.array(metrics)
    avg_loss = running_loss / len(dataloader)
    avg_metrics = running_metrics / len(dataloader)
    return avg_loss, avg_metrics

# 保存模型函数
```

```
def save_model(model, optimizer, epoch, filepath):
    torch.save({
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
    }, filepath)

# 加载模型函数
def load_model(model, optimizer, filepath):
    checkpoint = torch.load(filepath)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    return checkpoint['epoch']

# 训练函数
def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs, log_file,
     device, model_path=None):
    start_epoch = 0
    if model_path and os.path.exists(model_path):
        start_epoch = load_model(model, optimizer, model_path)
        print(f"Loaded model from {model_path}, starting from epoch {start_epoch + 1}")

    model.to(device)
    start_time = time.time()
    last_save_time = start_time
    save_interval = 300
    epoch_save_interval = 50 # 每个50epoch保存一次模型
    os.makedirs(os.path.dirname(log_file), exist_ok=True) # 确保目录存在

    with open(log_file, 'a') as f:
        for epoch in range(start_epoch, num_epochs):
            model.train()
            running_loss = 0.0
            running_metrics = np.zeros(5) # 用于累积评估指标
            for i, (inputs, targets) in enumerate(train_loader):
                inputs, targets = inputs.to(device), targets.to(device)
                optimizer.zero_grad()
                outputs = model(inputs)
                loss = criterion(outputs.squeeze(), targets)
                loss.backward()
                optimizer.step()

                running_loss += loss.item()
                metrics = calculate_metrics(outputs.squeeze(), targets)
                running_metrics += np.array(metrics)

            # 训练集平均损失和评估指标
```

```
            avg_loss = running_loss / len(train_loader)
            avg_metrics = running_metrics / len(train_loader)
            elapsed_time = time.time() - start_time

            # 验证集平均损失和评估指标
            val_loss, val_metrics = validate_model(model, val_loader, criterion, device)

            log = (f"Epoch [{epoch+1}/{num_epochs}], Train Loss: {avg_loss:.6f}, "
                   f"Train MSE: {avg_metrics[0]:.6f}, Train RMSE: {avg_metrics[1]:.6f}, "
                   f"Train MAE: {avg_metrics[2]:.6f}, Train MAPE: {avg_metrics[3]:.2f}, Train R2:
                       {avg_metrics[4]:.6f}, "
                   f"Val Loss: {val_loss:.6f}, Val MSE: {val_metrics[0]:.6f}, Val RMSE:
                       {val_metrics[1]:.6f}, "
                   f"Val MAE: {val_metrics[2]:.6f}, Val MAPE: {val_metrics[3]:.2f}, Val R2:
                       {val_metrics[4]:.6f}, "
                   f"Time Elapsed: {elapsed_time:.2f}s")
            print(log)
            f.write(log + '\n')

            # 检查是否需要保存模型
            current_time = time.time()
            if (epoch + 1) % epoch_save_interval == 0 or (current_time - last_save_time) >=
                save_interval:
                save_model(model, optimizer, epoch, f"model_epoch_{epoch+1}.pth")
                last_save_time = current_time

# 加载数据
train_file_path = r"train_5.xlsx"
val_file_path = r"test_5.xlsx"
train_dataset = FloodDataset(train_file_path)
val_dataset = FloodDataset(val_file_path)
train_loader = DataLoader(train_dataset, batch_size=9182, shuffle=True) # 增大batch
    size以增加显存占用
val_loader = DataLoader(val_dataset, batch_size=9182, shuffle=False)

# 初始化模型、损失函数和优化器
model = SimplifiedCNNTransformerModel()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 检查是否有GPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# 训练模型
num_epochs = 200
log_file = r"train.txt"
```

```
model_path = r"model_epoch_000.pth"
train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs, log_file,
    device, model_path)
```

附录 F 问题四代码 (模型调用与结果可视化)

cnn-transformer-model 调用代码

```
import torch
import pandas as pd
from torch.utils.data import Dataset, DataLoader
from sklearn.preprocessing import MinMaxScaler

# 自定义数据集类
class FloodDataset(Dataset):
    def __init__(self, file_path):
        self.data = pd.read_excel(file_path)
        self.features = self.data.iloc[:, :-1].values
        self.targets = self.data.iloc[:, -1].values

    def __len__(self):
        return len(self.targets)

    def __getitem__(self, idx):
        sample = torch.tensor(self.features[idx], dtype=torch.float32)
        target = torch.tensor(self.targets[idx], dtype=torch.float32)
        return sample, target

# 加载测试数据集
test_file_path = "test5.xlsx"
test_dataset = FloodDataset(test_file_path)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

# 加载模型
model_path = r"cnn-transformer-model.pth"
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = SimplifiedCNNTransformerModel()
model.to(device)

# 加载模型参数
checkpoint = torch.load(model_path, map_location=device)
model.load_state_dict(checkpoint['model_state_dict'])

# 设置模型为评估模式
```

```
model.eval()

# 预测并输出结果
predictions = []
with torch.no_grad():
    for inputs, _ in test_loader:
        inputs = inputs.to(device)
        outputs = model(inputs)
        predictions.extend(outputs.cpu().numpy())

# 将预测结果保存到文件或进行后续处理
output_file = "submit.csv"
pd.DataFrame(predictions).to_csv(output_file, index=False, header=["Predicted Probability"])

print(f"预测结果已保存到 {output_file}")
```

cnn-transformer-model 结果可视化

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 加载预测结果数据
predictions_file = "submit.csv"
predictions_data = pd.read_csv(predictions_file)

# 提取预测结果
predicted_probabilities = predictions_data["洪水概率"]

# 绘制直方图，使用彩虹配色
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(predicted_probabilities, bins=5000, edgecolor='none', alpha=0.7)

# 添加彩虹配色
colors = plt.cm.viridis(np.linspace(0, 1, len(patches)))
for patch, color in zip(patches, colors):
    patch.set_facecolor(color)
    patch.set_edgecolor(color) # 设置边缘颜色与面颜色相同
#设置显示中文
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
# 设置字体大小
plt.rcParams['font.size'] = 18
plt.title('发生洪水的概率的直方图')
plt.xlabel('洪水概率')
plt.ylabel('数目')
plt.grid(True)
plt.show()
```