中国研究生创新实践系列大赛
“华为杯”第二十届中国研究生数学建模竞赛

题　目：　脑出血患者预测模型的构建与评估

摘　要：

出血性脑卒中是由非外伤性脑内血管破裂引起的脑出血。其病因复杂，常由脑动脉瘤破裂、脑动脉异常等导致血液涌入脑组织，造成机械性损伤。血肿范围扩大是预后不良的重要危险因素之一，需要监测和控制血肿扩张。血肿周围水肿可能导致脑组织受压，进一步损伤神经功能。因此，建立智能诊疗模型，明确导致出血性脑卒中预后不良的危险因素，实现精准个性化的疗效评估和预后预测具有重要意义。

针对出血性脑卒中的医学影像数据化分析问题，我们在对患者信息原始数据进行匹配处理和统计分析的基础上，构建了多种预测模型，包括 LightGBM 和随机森林结合的分类预测模型、混合线性预测和 BiLSTM 结合的混合预测模型、基于 K-Means 的聚类模型，以及基于两独立样本 t 检验的因果推断模型。我们使用 python 编程对全体患者进行亚组聚类，预测各个患者发生血肿扩张事件的概率; 同时探究血肿体积、水肿体积和不同治疗方式之间的内在联系; 最后，我们预测患者预后 90 天的 mRS 评分，并分析出血性脑卒中患者的预后 90 天 mRS 评分与个人史、疾病史、治疗方法及影像特征等之间的关联关系，并依此为临床相关决策提供建议。

针对问题一，分析患者的入院首次影像检查时间间隔和 HM_volume 数据，以确定是否发生了血肿扩张事件，首先需要对原始数据进行预处理。本问通过 pandas 库对原始附件中的表格信息进行匹配与合并，并将发病到首次影像检查的时间间隔与首次影像检查的时间点进行加和，从而确定真正的发病间隔，以此判断患者是否在 48 小时内发生了血肿扩张事件。其次，我们借助 Nelder-Mead 算法将 LightGBM 和随机森林模型结合，并使用 VotingClassifier 集成学习器，通过采用不同的算法和数据特征，降低过拟合风险，提高模型的泛化能力，并获得了更可靠的预测结果。经检验，集成后的模型对患者发生血肿扩张事件概率的预测精度达到了 99%，相较集成之前的单一模型，误差下降了 70%。所有患者样本的血肿扩张情况和发生概率预测值记录在表 4 中。

针对问题二，我们借助 CEEMD 模态分解的功能将混合线性预测模型和 BiLSTM 相结合，研究了血肿周围水肿的发生及进展血肿周围水肿的发生及进展，分别对各患者水肿体

积的初始大小，以及水肿体积的相对大小随时间的变化进行预测，得到模型对患者水肿体积大小和发病时间间隔拟合曲线的 RMSE 为 25670.456。其次，我们通过 feature importance 对影响患者水肿体积大小的特征进行评估，选取前十名影响最大的特征作为 K-Means 的聚类依据，实现了对患者的亚组划分。结果表明，当聚类数为 4 时，模型对患者水肿体积大小随时间进展的拟合残差 RMSE 最小，为 15740.167，相较于聚类前性能提升了 36.6%。接着，我们建立了基于两独立样本 t 检验的因果推断模型，对不同治疗方法对水肿体积进展模式的影响进行分析，得出结论：1. 脑室引流组和对照组之间存在显著差异，p 值非常小（0.00025），T 统计量为负值（-3.7664），平均数差为正值（11.9008），即脑室引流治疗可以有效降低水肿体积的大小；2. 止血治疗可能会导致轻微的增加患者水肿体积的大小，p 值较小（0.00527），T 统计量为正值（2.83855），平均数差为负值（-5.46171）。最后，在研究血肿体积、水肿体积及治疗方法三者间的关系时，我们沿用第二小问中设立的患者亚组类别，并借助时间序列相关性分析模型与因果推断模型分别对血肿体积和水肿体积的关系、血肿体积和治疗方法的关系以及水肿体积和治疗方法的关系进行分析。

针对问题三，为了预测患者 90 天的 mRS 评分，需要先对患者的个人史、疾病史、发病相关及首次影像结果等特征进行 one-hot 编码，并将连续特征归一化到 [0，1] 范围，然后使用深度森林模型对 mRS 评分进行预测，经检验模型二次拟合后的训练精度达到了 0.87。接着，为了预测含随访影像检查的患者 90 天的 mRS 评分，我们在数据处理阶段加入了时间间断来引入时间因素，并借助 LSTM 模型 mRS 评分进行预测，经检验模型二次拟合的训练精度达到了 0.995。最后，我们借助 pearson 相关性分析和 feature importance 模型对出血性患者的预后与其个人史、疾病史、治疗方法以及影像特征之间的关系进行了探究。结果表明，对血性脑卒中患者的预后 90 天 mRS 评分影响最大的特征前几名分别为：糖尿病史、右前动脉血肿分布比例和发病到影像时间间隔。

综上，本文针对出血性脑卒中的医学影像数据化分析问题，通过构建多种智能诊疗模型，包括预测血肿扩张事件概率、水肿体积及其进展模式的预测、预测患者 90 天的 mRS 评分，并探究出血性脑卒中预后与个人史、疾病史、治疗方法和影像特征等的关系。本研究的成果对于实现精准个性化的疗效评估和预后预测具有重要的现实意义，可以为出血性脑卒中的临床决策提供科学依据。

关键词：医学影像数据化分析　深度森林　BiLSTM　LightGBM

目录

1 问题重述

1.1 引言

出血性脑卒中是由非外伤性脑内血管破裂引起的脑出血 [1]，占脑卒中病例的 10-15%。其病因复杂，通常源于脑动脉瘤破裂、脑动脉异常等因素，导致血液流入脑组织，引发脑部机械性损伤和生理病理反应。出血性脑卒中具有急性起病和快速进展的特点，其急性期病死率高达 45-50%，大多数患者伴有神经功能障碍，给社会和家庭带来沉重的负担。因此，准确评估其发病风险、精准预测患者的预后，以优化临床决策，显得至关重要。

血肿范围扩大是出血性脑卒中不良预后的一个重要危险因素 [2, 3]。血肿范围可能由脑组织受损和炎症逐渐扩大，导致颅内压力急剧上升，进一步损害神经功能，甚至危及生命。因此，监测和控制血肿扩张成为临床关注的焦点之一。同时，血肿周围水肿也备受关注，它可能导致脑组织受到压迫，影响神经元功能，加重神经功能损伤。因此，早期发现和预测血肿扩张和周围水肿对于患者的预后至关重要。

医学影像技术和人工智能为监测脑组织损伤提供了强大的工具 [4, 5]。因此，利用人工智能技术深度分析影像数据，结合个体信息、治疗方案和预后数据，构建智能诊疗模型，明确不良预后的危险因素，实现精确评估和预测引起了医疗界的广泛关注。

1.2 问题的提出

围绕出血性脑卒中临床智能诊疗的建模工作，本文依次提出如下问题：

1.2.1 问题一

a： 基于给定的数据表格” 表 1” 和” 表 2”，针对患者 sub001 至 sub100，分析各患者入院首次影像检查时间间隔和 HM_volume 数据，确定患者在发病后的 48 小时内是否发生了血肿扩张事件。若发生了血肿扩张事件，需记录事件发生的时间，并将结果填写到” 表 4” 中的相应字段。

b： 基于给定的前 100 例患者（sub001 至 sub100）的个人信息、疾病史以及与发病相关的信息，以及这些患者首次影像检查结果，构建一个预测模型来预测所有患者（sub001 至 sub160）是否在发病后发生血肿扩张事件。同时，借助建立的预测模型估计每位患者发生血肿扩张的概率，从而降低早期识别患者的风险，以便于采取适当的医疗干预措施。

1.2.2 问题二

a： 使用前 100 个患者的水肿体积数据和相关重复检查时间点，以建立描述水肿体积随时间变化的数学模型。使用建立好的数学模型将用于拟合患者的实际水肿体积数据。随后，需要计算实际水肿体积值与模型预测值之间的残差，以评估模型的准确性。这个任务的完成，有助于了解水肿随时间的动态变化，为进一步的研究提供基础数据。

b：在此任务中，需要研究不同患者在水肿体积随时间变化方面的个体差异。首先，将患者分为 3 至 5 个亚组，每个亚组代表一组具有相似水肿体积随时间变化模式的患者。随后，为每个亚组构建水肿体积随时间变化的曲线，以捕捉不同亚组之间的模式差异。接着，需要计算前 100 位患者的实际水肿体积值与其所属亚组曲线预测值之间的残差。最后，记录患者所属的亚组信息以及计算得到的残差结果，并将它们填写在规定的表格位置。这个任务有助于深入了解不同患者群体之间水肿体积随时间变化的差异，同时为个体化治疗提供重要见解。

c：在此任务中，需要分析不同治疗方法对水肿体积进展模式的影响。具体而言，研究将基于患者的治疗方案，即“表 1”字段 Q 至 W 中的信息，来评估治疗方法对水肿体积的影响程度。这个分析旨在揭示不同治疗方法是否对水肿的发展模式产生显著影响，以及不同治疗方法之间可能存在的差异。这个研究有助于为患者提供更个性化的治疗建议，以优化脑水肿的管理和治疗策略。

d：在此任务中，需要分析血肿体积、水肿体积以及不同治疗方法（“表 1”字段 Q 至 W）之间的关系。具体而言，研究将调查这三个因素之间是否存在相关性或相互影响。这种分析有助于深入了解不同治疗方法如何影响血肿和水肿的体积，并探究它们之间的关联，以便更好地理解脑部损伤的发展和治疗效果。这项研究有助于为医护人员提供有关最佳治疗方法和策略的重要见解，以实现更好的临床管理和患者护理。

1.2.3 问题三

a：在此任务中，需要基于前 100 位患者的个人史、疾病史、发病及治疗相关特征以及首次影像检查结果构建一个预测模型，用于预测所有患者在发病后 90 天的 mRS 评分。这个预测模型将考虑患者的首次影像数据，并旨在提供对患者预后的有用信息。预测结果将以有序等级变量（0-6）的形式记录，并填写在“表 4”的 I 字段中。

b：本任务要求利用前 100 位患者的全部临床、治疗信息以及首次和随访影像结果构建一个预测模型，用于预测所有具有随访影像检查的患者在发病后 90 天的 mRS 评分。这个模型将综合考虑患者的多方面信息，以提供更准确的预测结果。预测结果将以有序等级变量（0-6）的形式记录，并填写在“表 4”的 J 字段中。

c：在这个任务中，需要分析出血性脑卒中患者的 90 天 mRS 评分与其个人史、疾病史、治疗方法以及影像特征之间的关系。具体而言，研究将探讨这些因素与患者预后之间的相关性，以提供有关临床决策的建议。这项分析旨在帮助医疗专业人员更好地理解预后因素，并为患者提供更加个性化的治疗方案。

2 模型的假设

1. 假设患者自身携带的其他疾病不会对被研究对象（血肿体积、水肿体积）造成影响；
2. 假设患者除已给出的身体信息外，其余身体状况不具有显著差异；
3. 假设不同治疗方法对患者起到的作用程度近似相等；
4. 假设患者在被研究过程中，不会因其他因素导致身体情况的骤变。

3 符号说明

符号	意义
F	回归树的空间
f_k	相互独立的不同结构的树
c^{K-1}	前 K – 1 次棵树的正则化项和
l	损失函数
Ω	惩罚函数
β	固定效应的系数向量
γ	随机效应的系数向量
θ	随机效应的系数向量
$ReLU$	激活函数
$learning\_rate$	学习率
ACF	自相关函数
$PACF$	偏自相关函数

4 问题一建模与求解

4.1 问题分析

本问一共包含两个小问，第一问需要根据表 1 和表 2 的数据，判断患者 sub001 至 sub100 发病后 48 小时内是否发生血肿扩张事件，并记录血肿扩张发生时间。第二问则需要以是否发生血肿扩张事件为目标变量，基于表 1、表 2 和表 3 的数据，构建模型预测所有患者（sub001 至 sub160）发生血肿扩张的概率。

4.2 第一问数据处理

为了分析患者的入院首次影像检查时间间隔和 HM_volume 数据，以确定是否发生了血肿扩张事件，我们首先需要对原始数据进行预处理。这个过程包括以下步骤：

质量检测：首先，对原始数据进行严格的质量检测，以查找是否存在患者信息中的异常值或缺失值。

数据合并：接着，使用 pandas 库的 merge() 函数来合并表 1 和表 2 中具有相同 ID 的患者信息。这一步骤有助于整合不同数据源的信息，以便进一步分析。

列合并：最后，将患者信息中包含” 流水号” 字符串的列值及其之后的所有列值合并到对应的第一个索引行下。这样做可以实现对同一患者多次影像检查信息的合并，使数据更加整洁和易于分析。预处理后的表格部分展示如表4.1：

表 4.1　预处理后部分数据展示

ID	首次检查流水号	HM_volume	…	ED_Cerebellum_L_Ratio	检查时间
sub001	20161212002136	69714	…	0	2016-12-12 23:32:54
sub001	20161213000009	74902	…	0	2016-12-13 05:19:00
sub001	20161218000100	70952	…	0	2016-12-18 09:09:24
sub001	20161223001020	62831	…	0	2016-12-23 16:47:09
…	…	…	…	…	…
sub159	20200219000588	126642	…	0	2020-02-19 15:01:00
sub160	20200821002584	49019	…	0	2020-08-21 22:32:00
sub160	20200822000947	100470	…	0	2020-08-22 12:01:00

这些预处理步骤为我们提供了高质量的数据，以便进一步探索和分析患者的血肿扩张事件。

4.3 第一问求解

为了判断前 100 名患者是否在发病 48 小时内发生血肿扩张时间，并记录发生血肿扩张时间的时间，我们选择使用 python 的 pandas 库对预处理过后的数据进行进一步加工。患

者血肿扩张情况的检测流程如图4.1所示。

首先，使用 Pandas 读取预处理后的包含患者的临床信息和影像数据的数据集。接下来，编写 query_databas 函数，从数据集中查询特定患者的血肿体积和检查时间信息，并进行患者 ID 的匹配。接着，创建 get_volumes 函数，获取指定患者的血肿体积、检查时间以及发病到首次影像检查的时间间隔。然后，编写 check_expanded 函数，检查特定患者是否在发病后 48 小时内发生血肿扩张事件。通过 check_expanded 函数计算血肿体积的变化和时间间隔，如果满足条件（即血肿扩张），则会记录扩张事件发生的时间和相关信息。接着，通过迭代遍历预处理后表格中的患者数据，对每个患者进行查询和血肿扩张的检查，然后将结果填充到答案文件中的相应位置，以便进行进一步的分析和报告。

需要注意的是，患者在发病到首次影像检查之间具有一定的时间间隔，会影响到 48 小时内发生血肿扩张事件的判定。因此，在对同一患者多次检查时间间隔进行计算时，需加上发病到首次影像检查的时间间隔，以保证血肿扩张时间判定的准确性。

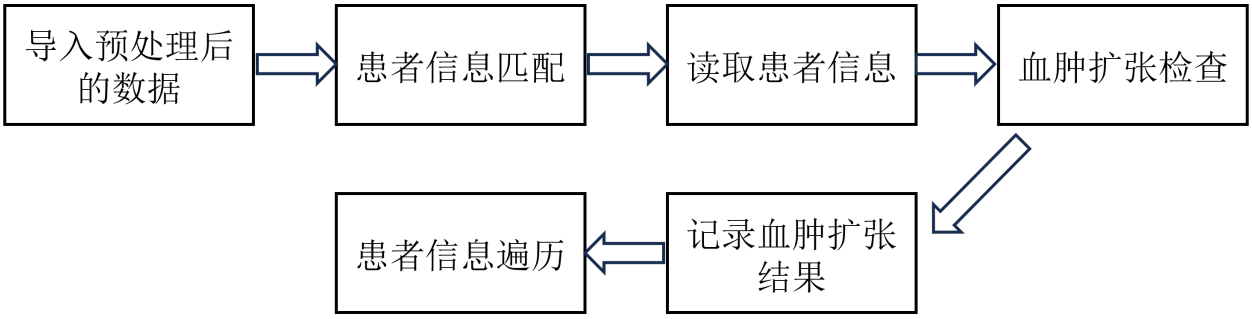

图 4.1　血肿扩张检测流程

至此，我们通过整合和分析患者的临床信息和影像数据，成功地完成了预测患者是否在发病后 48 小时内发生血肿扩张事件的任务，并记录了相关的结果，为临床问题的解决提供了有力支持。

4.4 第二问模型建立与求解

在本问中，我们的目标是基于前 100 例患者的个人史、疾病史、发病及治疗相关特征、首次影像检查结果等变量，构建一个模型，以预测所有患者（sub001 至 sub160）是否会发生血肿扩张事件，并输出患者发生血肿扩张事件的概率。

4.4.1 模型介绍

LightGBM 模型

Light Gradient Boosting Machine (LightGBM) 是一种梯度提升树（Gradient Boosting Tree）算法的改进版本，它具有高效的训练速度和优秀的性能。与传统的梯度提升树不同，LightGBM 引入了创新性的设计和优化，以提高模型的效率和泛化能力 [6]，其基本原理为：

1. Leaf-wise 生长策略：使用以叶子数为导向（带深度限制的 Leaf-wise 算法）的决策树建立算法而不是大多数 GBDT 工具使用的决策树深度导向（按层生长 (level-wise)）的生长策略，可以降低误差，得到更好的精度。
2. 分类特征的编码方式的优化：LightGBM 使用直方图来加速特征的分裂计算。它将数据按照特征进行分桶，然后在分桶上进行分裂选择，避免了对所有数据点的排序操作，提高了训练速度。

LightGBM 通过不断增加扩充树模型，根据数据特征来分裂生长树，并拟合上次的预测残差，最后累加每颗树所得的预测值，即得到最后的预测结果：

$$\hat{y}_i = \sum_{i=1} f_k(x_i), f_k \in F \tag{4.1}$$

式中，F 表示回归树的空间，每个 f_k 分别代表相互独立的不同结构的树。训练的目标函数如下式所示，其中 y_i 为标签真实值，$\hat{y}_i^{k-1}$ 为第 $K-1$ 次学习结果，c^{K-1} 为前 $K-1$ 次棵树的正则化项和，目标函数的含义为寻找一颗合适的数 f_k 使得函数的值最小。

$$\text{obj}^K = \sum_i l\left(y_i, \hat{y}_i^K\right) + \Omega(f_k) + c^{K-1} = \sum_i l\left(y_i, \hat{y}_i^{K-1} + f_K(x_i)\right) + \Omega(f_k) + c^{K-1} \tag{4.2}$$

其中，l 表示损失函数，表示预测值与真实值之间的残差，损失函数则是将每一棵树的残差相加，Ω 为惩罚函数。对于损失函数，可以拆分为每棵树的预测值与真实值的差值，假定第 0 课树为：$\hat{y}_i^0 = 0$。那么在构造第一颗树的时候，可以看成第 0 棵树的残差与第一颗树的残差相加，可以表示为：

$$\hat{y}_i^1 = \hat{y}_i^0 + f_1(x_i) \tag{4.3}$$

因此构造第 K 棵树的时候，预测值可以表示为：

$$\hat{y}_i^k = \hat{y}_i^{k-1} + f_k(x_i) \tag{4.4}$$

最终的损失函数可以表示为：

$$\sum_i l\left(y_i, \hat{y}_i^k\right) = \sum_i l\left(y_i, \hat{y}_i^{k-1} + f_k(x_i)\right) \tag{4.5}$$

接下来运用泰勒展开，此时选择二阶展开式结果比较好，可以在保证精度的同时简化计算，损失函数的泰勒二阶展开为：

$$\sum_i l\left(y_i, \hat{y}_i^{K-1} + f_K(x_i)\right) = \sum_i \left[l\left(y_i, \hat{y}_i^{K-1}\right) + g_i f_K(x_i) + \frac{1}{2} h_i f_K^2(x_i)\right] \tag{4.6}$$

g_i 和 h_i 是与变量无关的常数，分别表示第 i 个样本损失函数的一阶偏导和二阶偏导：

$$g_i = l'\left(y_i, \hat{y}_i^{K-1}\right), h_i = l''\left(y_i, \hat{y}_i^{K-1}\right) \tag{4.7}$$

简化后的目标函数可表示为：

$$\text{obj}^K = \sum_i \left[L\left(y_i, \hat{y}_i^{K-1}\right) + g_i f_K(x_i) + \frac{1}{2} h_i f_K^2(x_i)\right] + \Omega(f_K) + c \tag{4.8}$$

随机森林模型

随机森林是一种特殊的 Bagging 方法，其中决策树用作基本模型 [7]。其工作原理如下：首先，使用 Bootstrap 方法生成 m 个训练集。接下来，针对每个训练集，构建一棵决策树。在构建过程中，每当需要选择一个特征进行节点分裂时，随机从特征集中选取一部分特征进行考虑，然后从这些特征中选择最优的特征应用于节点分裂。每棵决策树都按照下列规则生成：

1. 如果训练集大小为 N，对于每棵树而言，随机且有放回地从训练集中的抽取 N 个训练样本，作为该树的训练集；
2. 如果每个样本的特征维度为 M，指定一个常数 m«M，随机地从 M 个特征中选取 m 个特征子集，每次树进行分裂时，从这 m 个特征中选择最优特征；
3. 每棵树都尽最大程度的生长，并且没有剪枝过程。

随机森林模型做分类和预测的原理如图4.2所示。

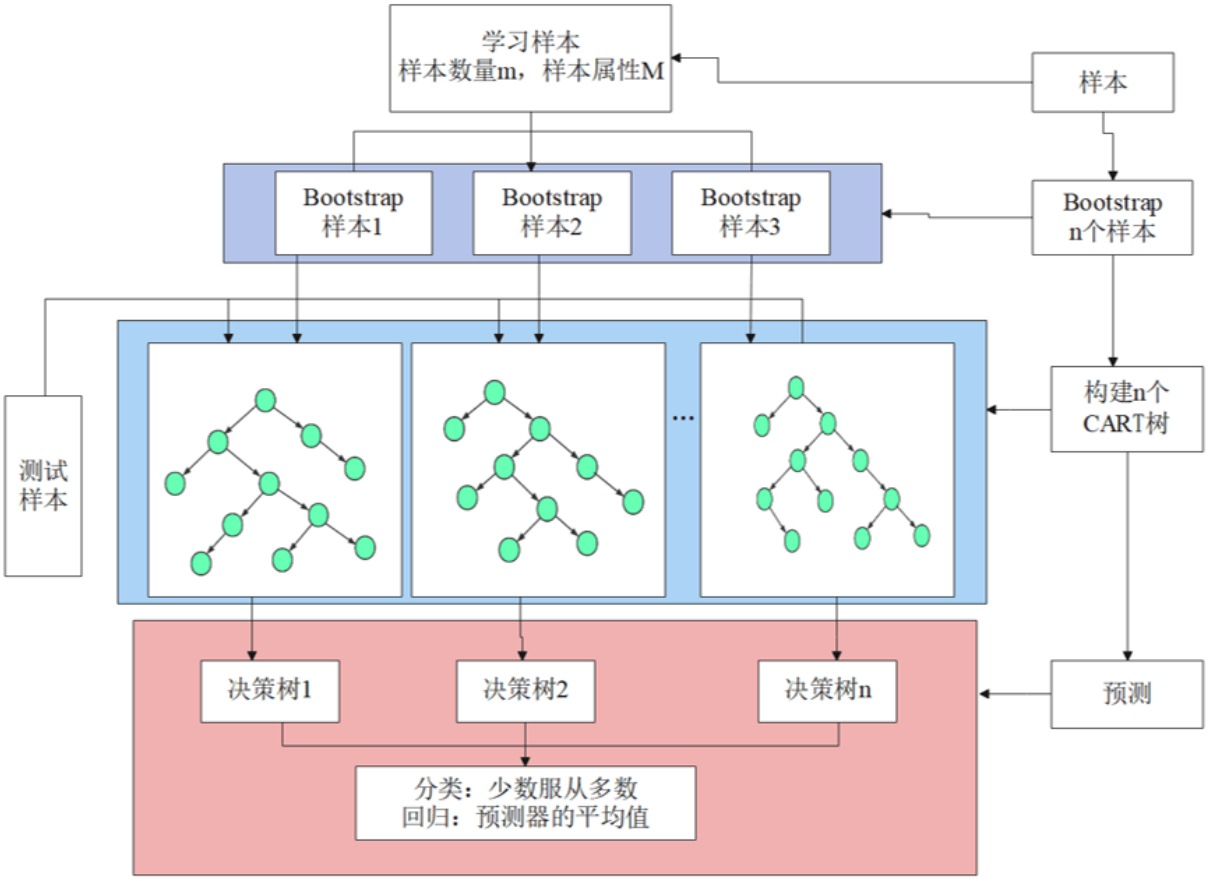

图 4.2　随机森林模型原理

假设 X 与 Y 分别为随机森林模型的输入和输出变量, 其中 Y 为连续变量, 给定训练数

据集:

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\} \tag{4.9}$$

可以通过遍历训练集数据的每个特征变量及其对应的特征值，将其定义为不同的区域。假设正在进行区域划分的切分变量为第 i 个变量，对应切分特征值为 j，则：

$$\begin{cases} a_1(i,j) = \{x \mid x^i \leq j\} \\ a_2(i,j) = \{x \mid x^i > j\} \end{cases} \tag{4.10}$$

通过不断的遍历划分，输入数据及被划分为 L 个子空间 $a_1, a_2, \cdots, a_L$，且每一个子空间 a_1 都包含其对应的样本数据和输出值 β_1，则此时模型的解为：

$$f(x) = \sum_{l=1}^{L} \beta_l I \quad (x \in \alpha_l) \tag{4.11}$$

采用和方差度量，度量目标是对于划分特征 i，对应划分点 j 两边的数据集 D1 和 D2。使 D1 和 D2 各自集合的均方差最小，同时 D1 和 D2 的均方差之和最小。其表达式为：

$$\min_{i,j}\left[\min_{\beta_1} \sum_{x_i \in \alpha_1(i,j)} (y_l - \beta_1)^2 + \min_{\beta_2} \sum_{x_l \in \alpha_2(i,j)} (y_l - \beta_2)^2\right] \tag{4.12}$$

4.4.2 模型建立与求解

初步建模

对于血肿扩张判定模型来说，其输入变量 X 为表 1 中患者的个人史，疾病史，发病相关（字段 E 至 W）、“表 2”中其影像检查结果（字段 C 至 X）及“表 3”其影像检查结果（字段 C 至 AG）等信息；输出变量 Y 为病人发生血肿扩张事件的概率。由于题目中并没有限定只对病人的血肿疾病信息进行判定分析，因此，我们模型的输入变量除去病人的血肿疾病信息，还包含了病人的水肿疾病信息。

考虑到第一问结果中，发生 48 小时内血肿扩张事件的病人仅有 23 名，仅占被研究患者总数的 23%。这种不平衡的数据可能导致模型为了追求精度，将占比少的数据信息权重调低，使得模型脱离了现实意义。因此，我们选择对血肿扩张患者的信息进行重采样，并对其各信息均加入白噪声，以提高模型的鲁棒性，降低数据的不平衡性。

根据输入变量的连续性、离散型以及类别性质，首先对输入变量进行初步处理，将输入变量转译为 85 个 float 型变量、8 个 int 型变量和 15 个 bool 型变量。然后，通过粒子群算法优化 LightGBM 和随机森林模型的超参数，确定二者的超参数如表4.2所示。

训练模型，得到 LightGBM 和随机森林模型对患者发生血肿扩张概率的预测准确率分别达到了 96.6667% 和 96.6666667%，具体训练结果如表4.3所示。

Nelder-Mead & 集成学习调优

表 4.2　LightGBM& 随机森林超参数设置

LightGBM		**随机森林**	
学习率	0.03	树的数量	300
叶子节点数	128	最大叶子节点数	15000
特征抽样比例	0.9	随机种子	0
叶子节点最小数据量	5	Bootstrap 抽样	TRUE

表 4.3　LightGBM & 随机森林结果展示

模型	LightGBM	随机森林
score_val	0.966667	0.966666667
pred_time_val	0.000998	0.025336027
fit_time	0.228427	0.30355835
pred_time_val_marginal	0.000998	0.025336027
fit_time_marginal	0.228427	0.30355835
stack_level	1	1
can_infer	TRUE	TRUE
fit_order	13	5

为了进一步提高模型对血肿扩张概率预测的准确度，我们决定使用 Nelder-Mead 和集成学习结合的方法，对两模型进一步调优并加权集成。

Nelder-Mead 是一种用于单目标优化的迭代数值优化算法，通常用于寻找函数的最小值。其主要原理是主要原理是通过不断寻找单个函数的最优参数值，以使该函数的输出值最小化或最大化。

集成学习是将多个不同的机器学习模型结合在一起，以提高预测的准确性和鲁棒性的策略。通过采用不同的算法和数据特征，集成模型可以降低过拟合风险，提高模型的泛化能力，并通过投票策略获得更可靠的预测结果。这种方法能够最大程度地利用各个模型的优势，使整体模型更强大。

对 LightGBM 和随机森林做集成学习的具体流程如图4.3所示。

为了实现对 LightGBM 和随机森林的进一步调优和集成，首先将 Nelder-Mead 的目标函数 $target$ 设置为 LightGBM 和随机森林的损失函数 l，并对其做最小值寻优。然后使用 VotingClassifier 集成学习器对二者进行权重分配并整合拟合结果。最后，使用优化好的 LightGBM 和随机森林对患者发生血肿扩张事件的概率再一次进行预测，并使用 Voting-Classifier 确立的权重对结果进行加权求和。经检验，集成后的模型对患者发生血肿扩张事件概率的预测精度达到了 99%，相较集成之前，误差下降了 70%。这为我们更准确地识别潜在高风险患者，采取及时的干预措施，提供了有力的保障。

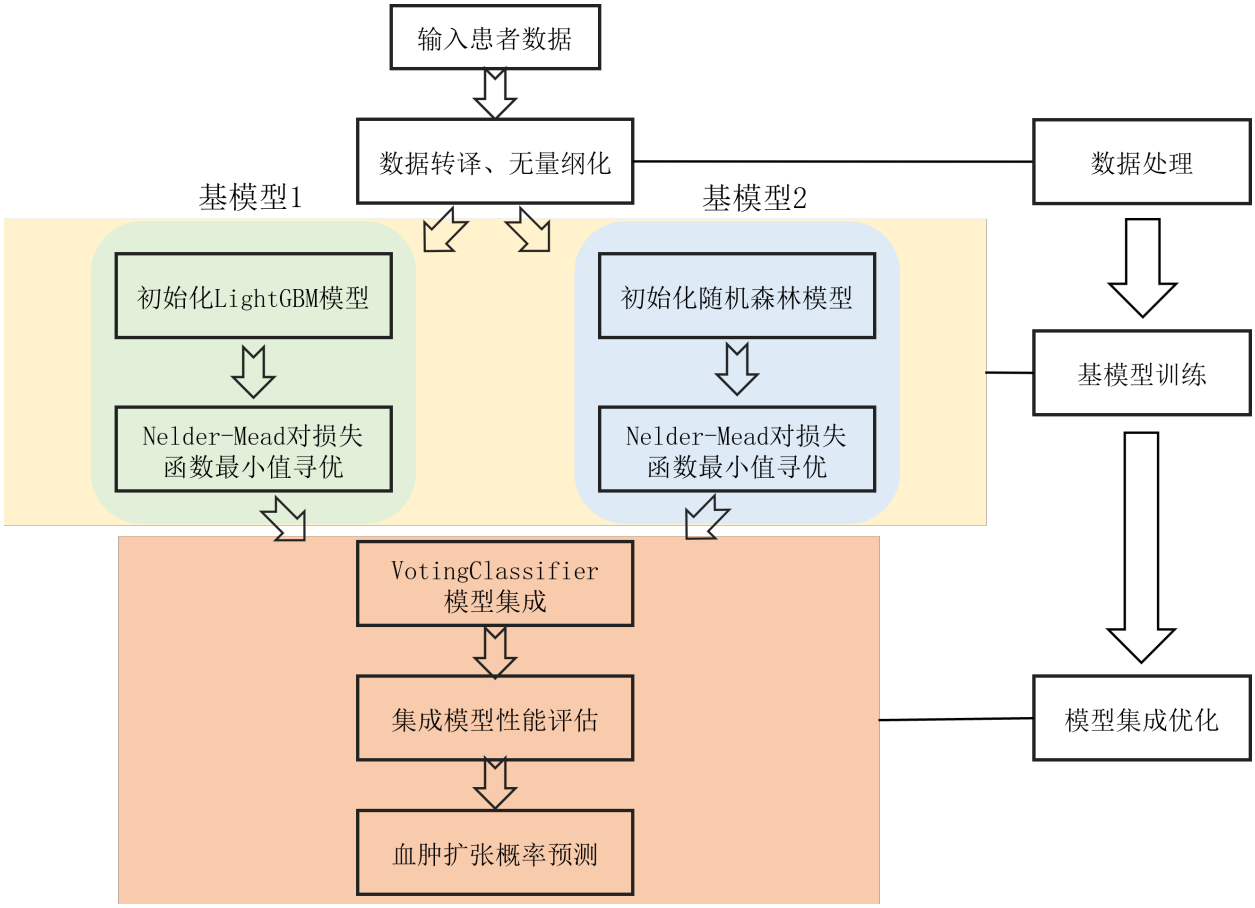

图 4.3　集成流程

5 问题二建模与求解

5.1 问题分析

为了研究血肿周围水肿的发生及进展，并探索治疗干预和水肿进展的关联关系。在这一环节中，我们将建立新的、更加高效的模型，为血肿、水肿患者的临床治疗提供可信的信息预测。

在本问中，我们一共要完成四项任务：

1. 根据“表 2”前 100 个患者的水肿体积和重复检查时间点，建立适当的数学模型，从而构建全体患者水肿体积随时间进展曲线，并计算前 100 个患者的真实值和所拟合曲线之间的残差；
2. 将患者根据某些特征（例如，性别、年龄、疾病类型等）分为不同的亚组。然后，针对每个亚组构建不同人群的水肿体积随时间进展曲线，并计算前 100 个患者的真实值和曲线之间的残差；
3. 建立合适的数学模型，剖析不同治疗方法对水肿体积进展模式的影响；
4. 构建血肿体积、水肿体积和治疗方法之间的散点图或其他合适的可视化模型，分析血

肿体积、水肿体积及治疗方法三者之间的关系。

5.2 第一问模型的建立与求解

针对第一问，我们需要根据“表 2”中的数据，构建一个水肿体积随时间进展的曲线，研究患者水肿体积的大小与发病时间的关系，并计算前 100 个患者水肿体积真实值和所拟合曲线之间存在的残差。具体来说，我们可以使用曲线拟合的方法，来计算真实值和所拟合曲线之间的残差，本文选用混合线性预测和 BiLSTM 相结合的方法，对患者水肿体积大小随时间的发展情况进行拟合。

5.2.1 混合线性预测模型

混合线性模型（Mixed Linear Model）是一种常用的统计学模型 [8]，用于分析多个因素对某个因变量的影响。其中，混合线性模型结合了线性模型和随机效应模型，可以同时考虑固定效应和随机效应的影响。

在混合线性模型中，我们假设存在一个固定效应和一个随机效应。固定效应表示某个因素对因变量的影响，而随机效应表示个体之间的差异。因此，混合线性模型可以同时考虑这些因素的影响，并且通过线性模型来拟合这些效应。

具体来说，混合线性模型的公式如下：

$$y = X\beta + z\gamma + e \tag{5.13}$$

其中，y 是因变量，X 是固定效应的系数矩阵，β 是固定效应的系数向量，z 是随机效应的系数矩阵，γ 是随机效应的系数向量，e 是误差项。

混合线性模型中的随机效应可以表示为：

$$z = Z\theta + R(u) \tag{5.14}$$

其中，Z 是随机效应的系数矩阵，θ 是随机效应的系数向量，$R(u)$ 是一个随机过程，表示个体之间的差异。因此，混合线性模型的完整公式如下：

$$y = X\beta + z\gamma + e = X\beta + Z\theta + R(u) + e \tag{5.15}$$

其中，$X\beta$ 是固定效应的影响，$Z\theta$ 是随机效应的影响，$R(u)$ 是随机误差项的影响。

5.2.2 BiLSTM 模型

BiLSTM（Bidirectional LSTM）是一种循环神经网络（RNN）模型，它结合了前向和后向的 LSTM 单元，能够处理序列数据 [9]。BiLSTM 由两个 LSTM 单元组成，一个用于正向传播，另一个用于反向传播，从而能够更好地捕捉序列数据中的信息。

LSTM 的主要组成结构为输入门 i_t、遗忘门 f_t 和输出门 o_t，同时在门结构中以 C_t 表示细胞单元的当前状态，h_t 表示隐藏层状态，x_t 为输入数据 [10]。先在输入层中，定义

原始故障时间序列为 $F_0 = \{f_1, f_2, \cdots, f_n\}$，则划分的训练集和测试集可以表示为 $F_{\text{tr}} = \{f_1, f_2, \cdots, f_m\}$ 和 $F_{\text{te}} = \{f_m + 1, f_m + 2, \cdots, f_n\}$，满足约束条件 $m < n$ 和 $m, n \in N$。然后对训练集中的元素 f_t 进行标准化，标准化后的训练集为 $F'_{\text{tr}} = \{f'_1, f'_2, \cdots, f'_m\}$。

为了适应隐藏层输入的特点，应用数据分割的方法对 F'_{tr} 进行处理，设定分割窗口长度取值为 L，则分割后的模型输入为 $\mathbf{X} = \{X_1, X_2, \cdots, X_L\}$。则对应的理论输出为 $\mathbf{Y} = \{Y_1, Y_2, \cdots, Y_L\}$。将 X 输入隐藏层，其包含 L 个按前后时刻连接的同构 LSTM 细胞。设定细胞状态向量大小为 S_{state}，则 C_{P-1} 和 H_{p-1} 两个向量的大小均为 S_{state}。应用训练好的 LSTM 网络 (表示为 $LSTM^*_{net}$) 进行预测。预测过程采用迭代的方法。首先理论输出 Y 的最后一行数据为 $F_f = \{f'm - L + 1, f'm - L + 2, \cdots, f'm\}$, 将 F_f 输入 $LSTM^*_{net}$ 输出结果，然后将 Y_f 的最后 $L - 1$ 个数据点和 $P_{\text{m+1}}$ 合并为新的一行数据：

$$Y_{f+1} = \{f'm - L + 2, f'm - L + 3, \cdots, f'm\} \tag{5.16}$$

将 Y_{f+1} 输入 $LSTM^*_{net}$ 则 $\mathrm{m} + 2$ 时刻的预测值为 P_{m+2}。依次类推，得到的预测序列为 $P_0 = \{P_{m+1}, P_{m+2}, \cdots, P_n\}$ 接下来对 P_0 进行 $z-$ score 反标准化 (表示为 de_zscore)，得到最终的与测试集 F_{te} 对应的预测序列为 $P_{te} = de\_z\,\text{corere}\,(P_0) = \left\{P^*_{m+1}, P_{m+2}, \cdots P^*_n\right\}$，其中：

$$p^*_k = p_k \sqrt{\sum_{t=1}^{n} \left(f_t - \sum_{t=1}^{n} f_t/n\right)^2 /n} + \sum_{t=1}^{n} f_t/n \quad m+1 \le k \le n, k \in N \tag{5.17}$$

类似地，将 X 的每行作为模型输入可以得到与训练集 F_{tr} 对应的拟合序列 P_{tr}，最后通过计算 F_{tr} 和 P_{tr} 以及 F_{te} 和 P_{te} 的偏差定量地给出模型的拟合和预测精度。

在 BiLSTM 中，输入序列首先通过一个 LSTM 单元进行前向传播，然后将输出通过另一个 LSTM 单元进行反向传播。这个过程在每个时间步重复，直到达到序列的末尾。

BiLSTM 的公式如下：

$$h_t = f_1\left(h_{t-1}, x_t\right) \tag{5.18}$$

$$h_t = g_1\left(h_{t-1}, x_t\right) \tag{5.19}$$

其中，h_t 是当前时间步的输出，h_{t-1} 是前一个时间步的输出，x_t 是当前时间步的输入。f_1 和 g_1 是 LSTM 单元的激活函数，用于计算当前时间步的输出。

BiLSTM 模型的原理如图5.4所示。

5.2.3 CEEMD 模型

CEEMD 模型是一种用于分析非线性系统的方法 [11]，它可以分解出系统的各个模态，这也是混合线性预测模型和 BiLSTM 模型结合的基础。通过 CEEMD 模型，我们可以将原始数据分为分别适合与混合线性预测模型和 BiLSTM 模型的模态，并进行分别拟合，从而实现二者相结合的目的，提高模型的拟合精度。CEEMD 的原理如图5.5所示。

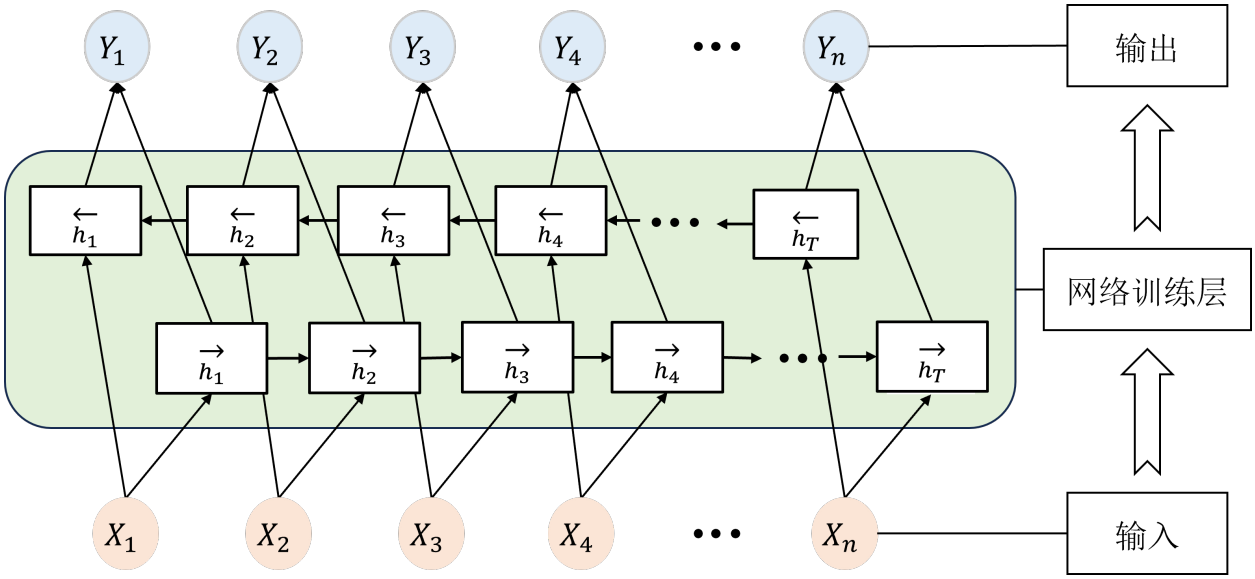

图 5.4　BiLSTM 原理

5.2.4 模型参数的确立与求解

为了研究全体患者水肿体积随时间进展的变化趋势，我们首先将前一百名患者的水肿体积信息以发病时间为初始节点，按照间隔时间进行排序。接着，将每名患者的水肿体积数据与发病时的初始体积作比值，得到每名患者水肿体积的相对大小随时间的变化表格。然后使用混合线性预测模型与 BiLSTM 模型结合的预测方式，对患者水肿体积和发病间隔时间进行拟合。拟合结果分为两块：对每名患者水肿体积初始大小的预测；对患者水肿体积的相对大小随时间变化的预测。

在确定模型的超参数之前，先对患者水肿体积大小和发病时间间隔的趋势进行初步分析，患者水肿体积大小随发病时间间隔的趋势如图5.6示。

除此之外，考虑到数据样本集合的大小，以及输出变量和输入变量的维度，我们需要选择合适的参数，用以优化模型的拟合效果。

混合线性预测模型超参数确立：对于此次任务，混合线性预测模型的主要超参数分别为 intercept、time_gap 和 Group Var。其中，intercept 是常数项，用于平移模型输出的截距，为了对每名患者水肿体积的相对大小进行拟合，我们选用各患者水肿体积初始大小的平均值 26020.755312 作为 intercept 的取值；time_gap 表示时间间隔，用于控制时间序列模型的建模时间和步长，考虑到患者多次复查之间的时间间隔，以及模型的整体拟合效果，将该时间步长取值为-3.29294；Group Var 表示组方差，用于体现不同组别之间的差异，从而对不同患者样本进行区分，避免模型过于拟合某一患者样本，对该超参数的值取 0.648374。

BiLSTM 模型超参数确立：BiLSTM 是由两个传播方向相反的 LSTM 模型组成的，因此只需确定 LSTM 模型的超参数即可。LSTM 模型的结构示意如图5.7所示。

LSTM 的主要模型超参数有 num_epochs、batch_size、learning_rate 和 dropout。

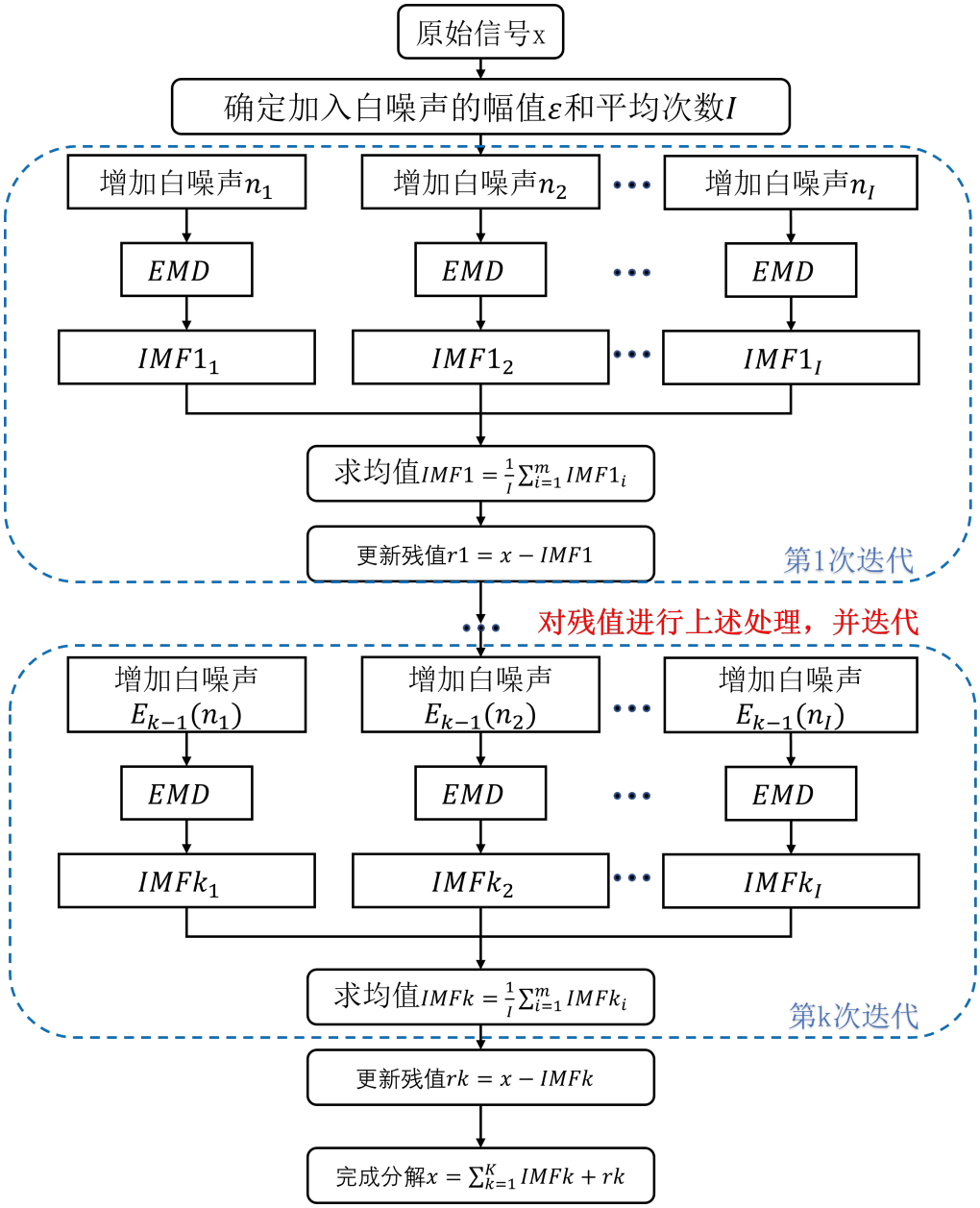

图 5.5　CEEMD 原理

1. num_epochs 指定模型训练的迭代次数，考虑到患者样本数据集并不大，不会对模型训练参生较大的负荷，因此为了提高模型的训练精度，我们将 num_epochs 设置为 100。
2. batch_size 表示每次训练时使用的样本数量，当该参数较大时，模型的训练速度和收敛速度会提高，但是同样会造成计算成本的上升，考虑到二者的综合博弈，将 batch_size 设置为 32。
3. learning_rate 为模型更新参数时的学习率，学习率越大，模型更新参数的速度越快，但可能会造成过拟合；学习率越小，模型更新参数的速度越慢，但可以减少过拟合的风险。为了防止过拟合，并尽可能的提高模型的训练精度，将 learning_rate 设置为 0.001。

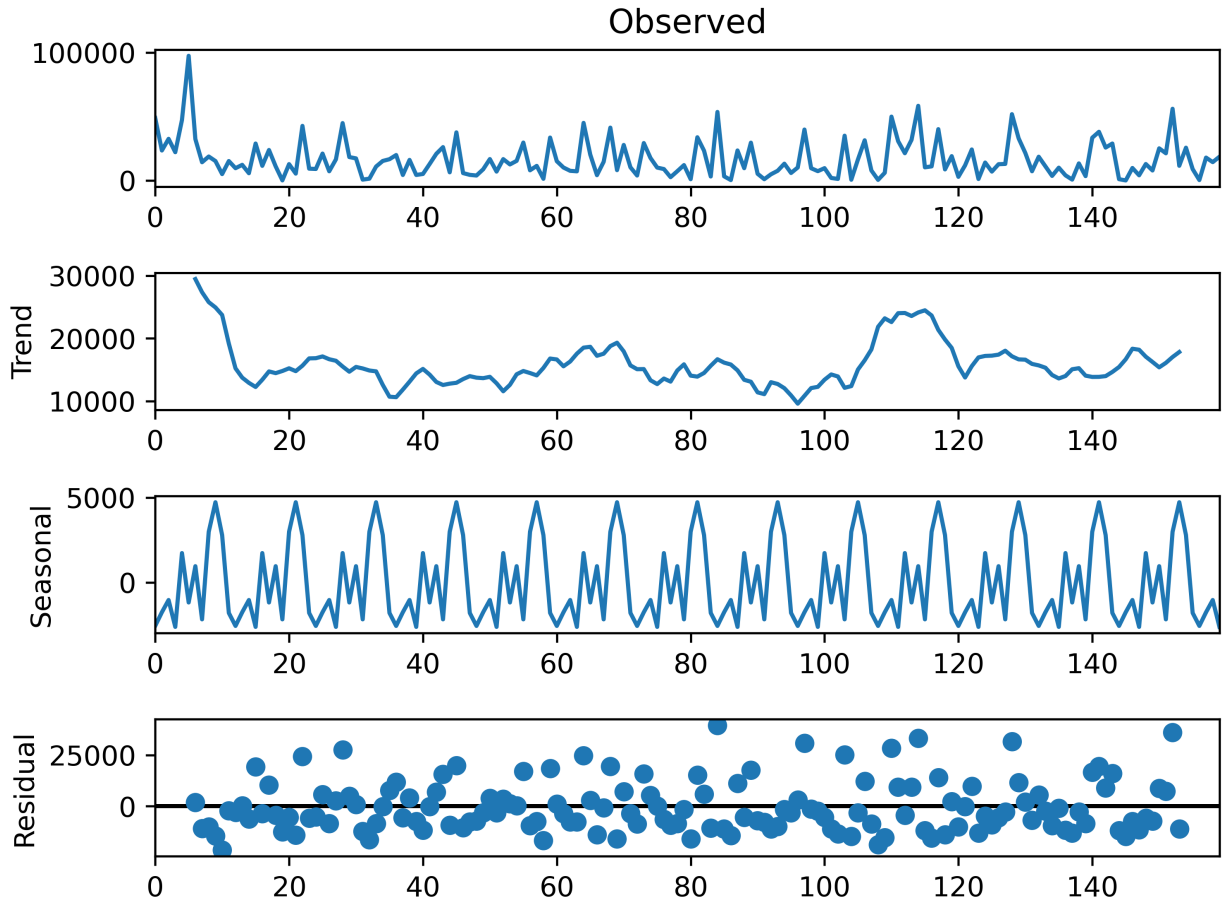

图 5.6　水肿体积大小随发病时间间隔的趋势

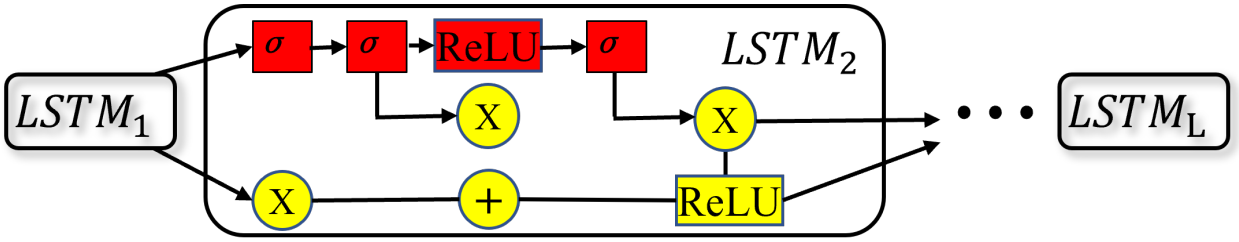

图 5.7　LSTM 结构示意图

4. dropout 为训练过程中丢弃神经元（即随机抑制神经元的活动）的概率。丢弃神经元可以防止过拟合，减少模型对训练数据的依赖性。为了控制模型的复杂度，提高模型的泛化能力，将 dropout 设置为 0.5。

模型结果展示

将患者样本的水肿体积数据以及发病间隔时间带入模型，并使用 CEEMD 模型对初始数据做模态分解，分解结果如图5.8所示。

对两组不同模态的数据分别采用混合线性预测模型和 BiLSTM 模型进行拟合，然后

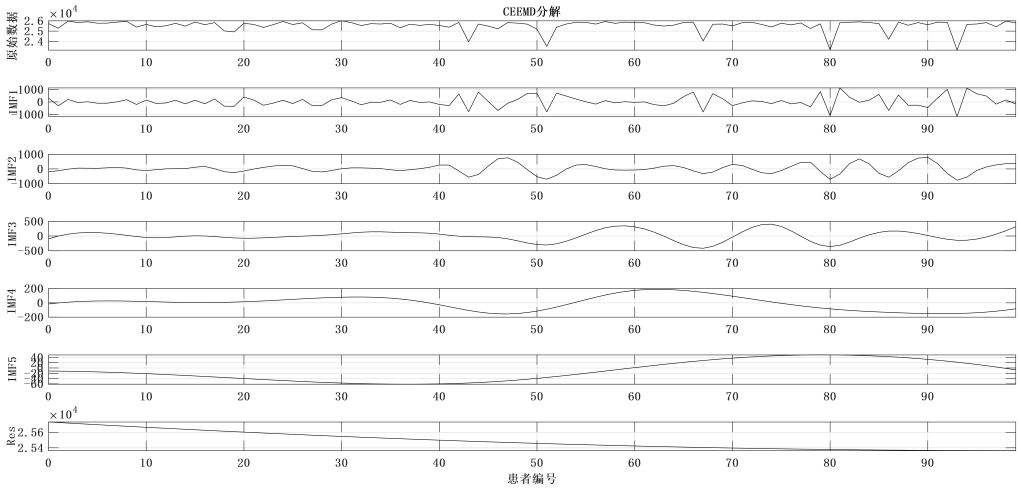

图 5.8　模态分解结果

对拟合结果进行加权求和，最后将同一患者样本的拟合误差取平均值，模型拟合效果如图5.9所示。

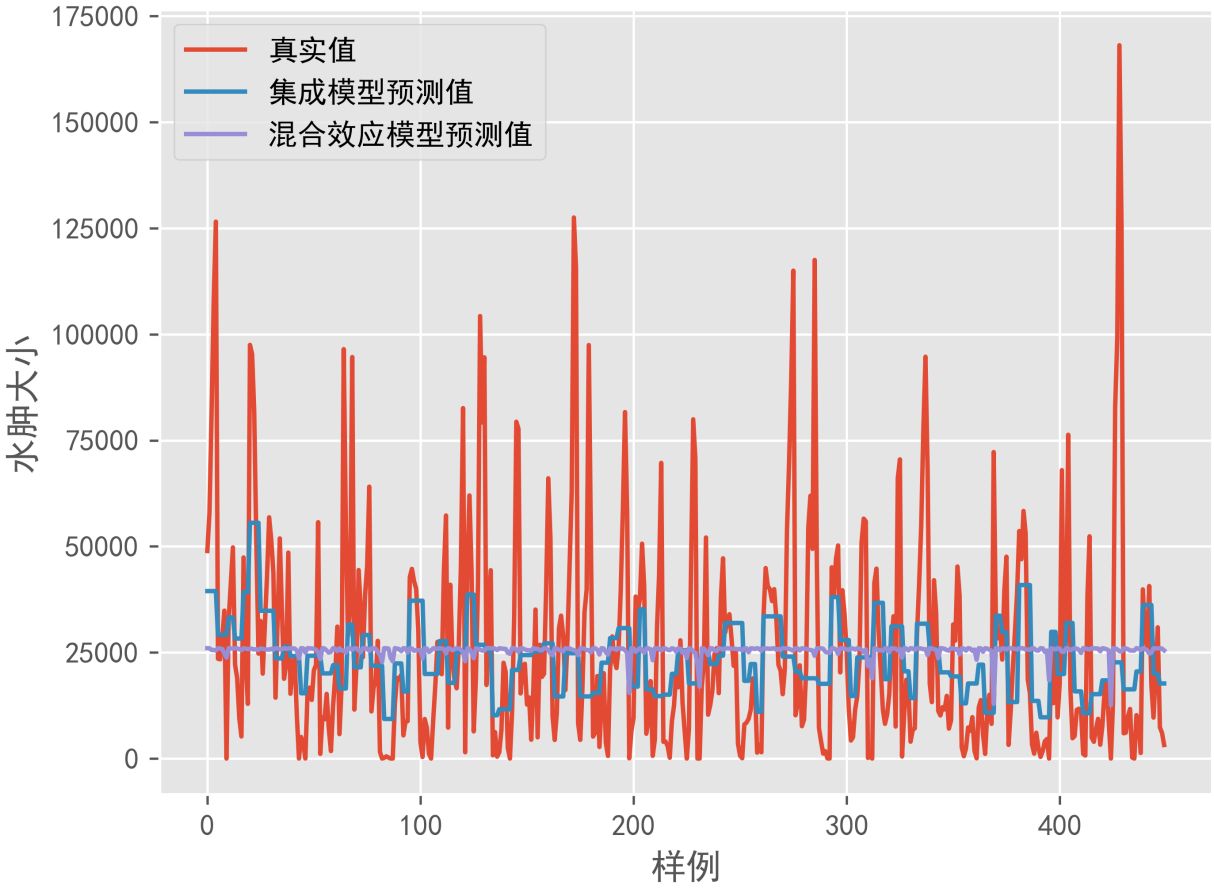

图 5.9　模型拟合曲线

患者水肿体积大小随时间进展的拟合结果如表5.4所示。

表 5.4　拟合结果展示

患者 ID	水肿体积预测值	水肿体积实际值	预测误差
sub001	39472.44922	84583	-45110.55078
sub002	29165.27344	22039.8	7125.473438
sub003	33256.30859	41226.66667	-7970.358073
sub004	28283.60742	13972.75	14310.85742
...	...	...	...
sub099	20033.80469	20034.33333	-0.528645833
sub100	17693.62109	5530.333333	12163.28776

分析拟合结果可知，模型对患者水肿体积大小和发病时间间隔拟合曲线的 RMSE 为 25670.456。这是因为，相同的时间间隔对应的患者样本个体不同，在这样但输入的限制下，输入数据的信息可能无法充分地传递给多个输出，导致输出结果的不准确。在对患者样本进行进一步聚类处理后，模型效果会有明显的提升。

5.3 第二问模型的建立与求解

为了研究患者水肿体积随时间进展模式的个体差异，并构建不同人群的水肿体积随时间进展曲线。我们首先需要对患者样本进行聚类，将其按照不同的特征化成 3-5 个亚组。然后，在使用调优后的混合预测模型对各自亚组下的患者样本水肿体积随时间的变化趋势做进一步拟合。

5.3.1 K-means 聚类

K-means 聚类是一种无监督学习算法，用于将数据集分为 K 个不同的簇 [12]。其基本思想是将数据集中的每个数据点分配到距离其最近的簇中心，然后不断迭代更新簇中心和数据点的簇分配，直到收敛。K-means 聚类的具体流程如下所示：

1. 首先，随机选择 K 个数据点作为初始的簇中心。
2. 对于每个数据点，计算其到 K 个簇中心的距离，并将其分配到距离最近的簇中。
3. 对于每个簇，重新计算其簇中心，即计算簇中所有数据点的均值。
4. 重复步骤 2 和 3，直到簇中心不再改变或达到最大迭代次数。

在计算每个数据点到簇中心的距离时，可以采用欧几里得距离，其计算公式如下：

$$D(i,j)=\sqrt{(x_{1j}-x_{1i})^2+(x_{2j}-x_{2i})^2+\cdots+(x_{nj}-x_{ni})^2} \tag{5.20}$$

其中，$D(i,j)$ 表示第 i 个点和第 j 个点之间的欧几里得距离，x_{ni} 和 x_{nj} 分别表示第 i 个点和第 j 个点的第 n 个坐标值。

表 5.5　feature importance 展示

特征名称	importance
HM_volume	4751.719707
delta_ED_volume	1941.639098
ED_ACA_R_Ratio	1718.325745
检查时间	941.2162692
发病到首次影像检查时间间隔 _y	768.8101595
...	**...**
饮酒史	-0.951681477
止吐护胃	-1.023995937

特征选取：在进行 k-means 聚类分析时，特征选择至关重要。需要关注以下方面：首先，确保所选特征彼此无关，避免对模型产生不利影响；其次，确保特征数量合理，避免过多或过少；第三，了解每个特征对聚类结果的影响，使用评估指标来评估特征的重要性。

为了选取合适的特征，保证聚类结果的合理性，我们使用问题一种建立的 LightGBM 和随机森林结合的拟合模型对患者水肿体积大小进行拟合，并借助 feature importance 对影响患者水肿体积的各个特征进行分析，得到各特征的 feature importance 如表5.5所示。

考虑到特征数量的合理性以及各个特征对患者水肿体积影响的重要性，我们挑选了对患者水肿体积影响最大的十个特征，并分别进行了 3-5 个 cluster 的聚类。计算每种设置下的 RMSE 值，发现四聚类的效果最好，在该种聚类下，其 RMSE 值为 16058.834。聚类结果的四个聚类中心如表5.6所示。

表 5.6　聚类中心展示

HM_volume	delta_ED_volume	ED_ACA_R_Ratio	发病到首次影像检查时间间隔 _y	...	低压
75451.78571	0.260466285	0.170601	3.428571	...	92.07143
16215.3617	0.068643194	0.036738	2.955745	...	90.87234
179631.5	-0.198321164	0.295934	13.25	...	77
40348.56757	0.329963231	0.041451	3.099189	...	96.67568

5.3.2 拟合求解

将聚类后的亚组带入第一小问中建立的混合预测模型，并对损失函数 loss 进行最小值寻优，得到寻优后的损失函数 loss 值为 0.0147，模型对患者水肿体积大小随时间进展的拟合残差 RMSE 为 15740.167。聚类后模型的预测结果与患者真实数据的散点图如图5.10所示。

分析数据易知，相较于聚类前模型对患者水肿体积大小的拟合曲线，聚类后曲线的拟

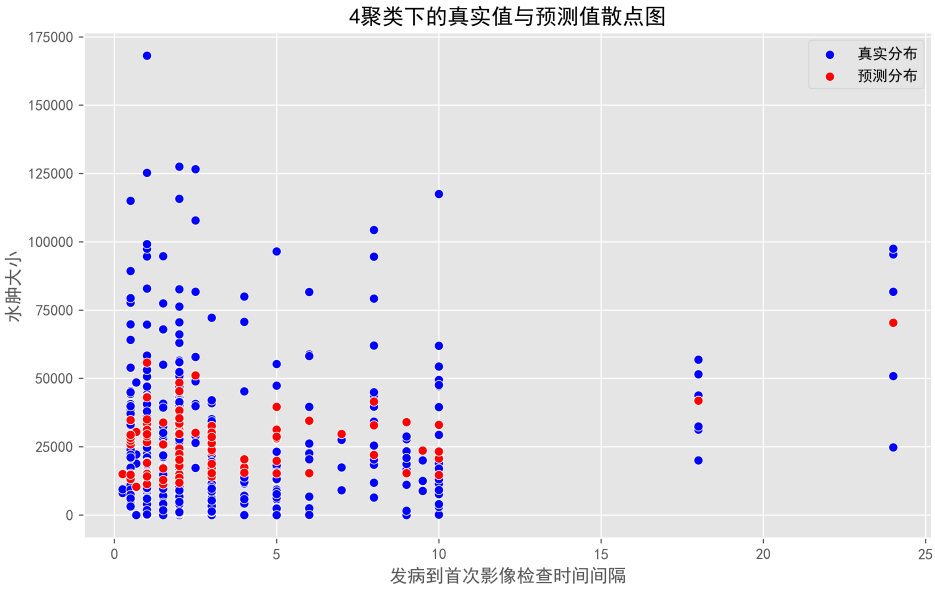

图 5.10　四聚类后的拟合散点图

合残差 RMSE 由 25670.456 降至了 15740.167，即模型效果提升了 36.6%。

5.4 第三问模型的建立与求解

从数学模型的角度来看，分析不同治疗方法对水肿体积进展模式的影响，属于分析不同因素对输出的影响。因此，我们可以建立不同因素与输出的关系的数学模型来分析它们的影响。

5.4.1 因果推断法

在医学研究中，我们经常需要确定某种治疗方法是否真的能够有效地减少水肿体积的进展 [13]。为了解决这个问题，我们可以使用因果推断的方法。因果推断是一种数学方法，用于确定某个事件是否导致了另一个事件。在我们的研究中，我们可以将治疗方法作为因，将水肿体积的进展作为果，然后使用因果推断的方法来确定不同治疗方法对于水肿体积进展的影响效果。

因果推断法的原理如下：

因果推断法原理基于一个原则，即一个变量的变化（假设为原因）会导致另一个变量的变化（假设为结果）。因果推断法的公式如下：

$$Y = \alpha + \beta X + \varepsilon \tag{5.21}$$

其中，Y 是因变量，X 是自变量，α 是常数项，β 表示自变量对因变量的影响程度，ε 是误差项。

5.4.2 建模与求解

在对患者水肿体积和不同治疗方法做因果推断之前，需要先对患者的基本信息、不同治疗措施应用与否等特征以及病情进展结果 (如水肿变化率) 做预处理。

连续特征标准化：为了减少患者信息中连续特征的数量级差异，使特征具有可比性，首先对患者的连续特征进行标准化处理。这里我们使用 z-score 标准化来消除连续特征间的量纲差异，z-score 标准化的计算公式如下：

$$z - score = \frac{(x - \mu)}{\sigma} \tag{5.22}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.23}$$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{5.24}$$

其中，n 表示数据的样本大小，x 表示原始数据；μ 表示表示数据的平均值，即样本均值；σ 表示数据的标准差，即样本标准差。**线性回归：**建立线性回归模型，以病情进展结果（水肿体积变化率）为因变量，治疗措施为自变量，控制其他特征（年龄、性别等）。使用这些特征来拟合治疗效应，并计算 RMSE 评估模型拟合优度。**两独立样本 t 检验：**两独立样本均数差异的 t 检验（two-sample t-test）是一种常用的假设检验方法，用于比较两个样本的均值是否具有显著差异。因此，可以根据平均值差异的大小和 P 值来判断治疗措施的效果。

t 检验的基本原理是假设两个样本来自具有相同均值的正态分布，并检验样本均值是否存在显著差异。如果样本均值存在显著差异，则可以拒绝原假设，即认为两个样本之间存在显著差异。t 检验的公式如下：

在两独立样本 t 检验中，我们需要比较两个样本的均值是否相等。假设样本 2 的观测值为 y，样本 2 的均值 $_2$，样本 2 的标准差 $sigama_2$，则两样本 t 检验的原理和公式如下：

$$t = \frac{\bar{x} - \bar{y}}{s} \tag{5.25}$$

$$s^2 = \frac{(\bar{x} - \bar{y})^2}{n_1 - 1} + \frac{(\bar{x} - \bar{y})^2}{n_2 - 1} \tag{5.26}$$

$$df = n_1 + n_2 - 2 \tag{5.27}$$

$$p = P(t > |t|) \tag{5.28}$$

其中，$\bar{x}$ 和 $\bar{y}$ 分别是样本 1 和样本 2 的均值，s 是两个样本的标准差之和，df 是自由度，p 是双尾检验的概率值。

结果展示：在通过两独立样本 t 检验计算出两组样本间的差异值之后，计算两组样本间的平均值差异，并根据平均值差异的大小和 P 值来判断治疗措施的效果。检验结果如表5.7所示。

表 5.7　两独立样本 t 检验结果

治疗方法	t 值	p 值	RMSE
脑室引流	-3.996477521	9.83845E-05	10.85092637
止血治疗	2.215551266	0.028152135	-3.283633053
降颅压治疗	-0.631849939	0.5283981	0.974667795
降压治疗	-1.127355326	0.261301413	2.393760777
镇静、镇痛治疗	-1.539470021	0.125689914	2.200534668
止吐护胃	0.076988611	0.93873001	-0.243611368
营养神经	-0.32141552	0.748320317	1.016726442

在我们的研究中，我们发现脑室引流组和对照组之间存在显著差异，p 值非常小（0.00025），T 统计量为负值（-3.7664），平均数差为正值（11.9008）。这些数据表明，脑室引流组的意识障碍程度显著低于对照组。因此，脑室引流治疗可以有效降低意识障碍程度。

相比之下，止血治疗可能会导致轻微的增加意识障碍程度，p 值较小（0.00527），T 统计量为正值（2.83855），平均数差为负值（-5.46171）。虽然这种差异可能看起来很小，但我们需要考虑到 p 值较小可能是由于样本大小不足导致的。因此，我们需要进一步研究以确定止血治疗的实际效果。

其他治疗方法（如降颅压、降压、镇静镇痛等）没有显示出明显的疗效，p 值较大，一般大于 0.05。这表明这些治疗方法在治疗组和对照组之间没有显著的效应。

5.5 第四问模型的建立与求解

研究血肿体积、水肿体积及治疗方法三者之间的关系有助于深入理解脑损伤的病理生理过程和治疗效果，通过了解血肿体积与水肿体积之间的关系、血肿体积与治疗方法之间的关系以及水肿体积与治疗方法之间的关系，可以为临床实践提供有价值的指导，帮助医生为患者制定个体化的治疗方案。

为了进一步分析研究血肿体积、水肿体积及治疗方法三者之间的关系，我们分别选择使用时间序列相关分析法、两独立样本 t 检验法对三者间的关系一一分析。

5.5.1 时间序列相关性分析

时间序列相关性分析是研究时间序列数据中各数据点与时间之间的依赖关系，即数据点随时间变化的规律 [14]。在时间序列相关性分析中，常用的公式是自相关函数（Autocorrelation Function，ACF）和偏自相关函数（Partial Autocorrelation Function，PACF）[15]。

自相关函数描述了时间序列中前一时间点与当前时间点之间的相关性，其公式为：

$$ACF(k) = E[(Y_t - Y_m)(Y_{t+k} - Y_m)] \tag{5.29}$$

其中，Y_t 表示时间序列中第 t 个观测值，Y_m 表示时间序列中第 m 个观测值，E 表示期望值。k 表示自相关函数的阶数，即需要计算的时间点之间的距离。ACF(0) 表示时间序列自身与当前时间点之间的相关性，ACF(1) 表示时间序列自身与下一个时间点之间的相关性，以此类推。

偏自相关函数描述了时间序列中除了前一时间点与当前时间点之间的相关性外，还存在滞后一阶的偏差影响。其公式为：

$$PACF(k) = E[(Y_t - Y_m)(Y_{t+k} - Y_{m-k})] - E[(Y_t - Y_m)(Y_{t+k-1} - Y_{m-1})] \tag{5.30}$$

其中，$m-k$ 表示偏自相关函数的阶数，即需要计算的时间点之间的距离。PACF(0) 表示时间序列自身与当前时间点之间的相关性，PACF(1) 表示时间序列自身与下一个时间点之间的相关性，以此类推。

通过自相关函数和偏自相关函数可以分析时间序列的长期依赖关系和周期性变化规律，为时间序列数据的分析和预测提供依据。

5.5.2 血肿、水肿相关性分析

在对患者血肿体积和水肿体积的关系进行分析时，我们沿用第二小问中对全体患者聚类的亚组结果。并分别对四个亚组的血肿体积、水肿体积关系做时间序列相关性分析。将四个亚组的患者信息分别带入时间序列相关性分析模型，得到各组患者血肿体积和水肿体积的相关系数如表5.8所示。

表 5.8　各亚组血肿体积和水肿体积的相关系数表

亚组类别	HM_volume 和 ED_volume 的相关系数
0	0.256879203
1	0.121319768
2	0.126008688
3	0.225407361

通过对四类患者的血肿体积和水肿体积进行相关性分析，得出每类患者的相关系数。这些系数用于衡量两个变量之间的线性关系强度和方向。

对于第 0 类患者，HM_volume 和 ED_volume 的相关系数为 0.2568792029800091，属于中等强度正相关，表明血肿体积增加通常会导致水肿体积增加。

对于第 1 类患者，相关系数为 0.12131976804052762，表明血肿体积和水肿体积之间存在弱的负相关关系。这可能是因为该类患者的血肿体积较小，对水肿体积的影响较小。

对于第 2 类和第 3 类患者，相关系数分别为 0.126008687633567 和 0.22540736135529715，均属于较弱相关，表明血肿体积和水肿体积之间存在中等强度的正相关关系。

5.5.3 水肿、治疗方法的因果推断分析

在第三小问种，我们研究了不同治疗方法对水肿体积进展模式的影响。在这一环节中，我们将进一步的研究不同治疗方法，对不同亚组患者水肿体积进展模式的影响。同时在研究方法上，我们沿用第三小问建立的因果推断模型做细致研究。

将四个亚组患者预处理过后的信息分别带入因果推断模型，得到各组别患者治疗方法和水肿体积的检验结果如表5.9所示。

表 5.9　各亚组不同治疗方法对水肿体积进展模式的影响

治疗方法	亚族类别	t 值	p 值	平均数差
脑室引流:	0	0.264948318	0.79554551	-0.220757479
镇静、镇痛治疗:	0	-1.476222738	0.165637734	1.330724984
...	...	...	...	...
降压治疗	1	-0.827261589	0.412454283	2.664417911
...	...	...	...	...
镇静、镇痛治疗:	2	NAN	NAN	0.179997761
...	...	...	...	...
营养神经:	3	-0.023796308	0.981150241	0.160024917

经过两独立样本 t 检验的研究，对于患者采用的不同治疗方法与水肿体积大小之间的关系得到了以下结论：

1. 对于编号为 0、1、2 的亚组而言，所采用的治疗方法均没有显示出明显的疗效，p 值较大，大于 0.05。这表明这些治疗方法在治疗组和对照组之间没有显著的效应；
2. 对于编号为 3 的亚组而言，止血治疗可能会导致轻微的增加意识障碍程度，p 值较小（0.0146），T 统计量为正值（2.56），平均数差为负值（-11.735），即止血治疗可以对患者水肿体积的减小有着不可忽视的抑制效果。

综上所述，不同治疗方法对患者的水肿体积大小有显著影响。对于采用止血治疗的患者，水肿体积大小存在统计学差异；而对于采用降颅压治疗的患者，水肿体积大小没有显著差异。因此，在治疗过程中需要根据患者的具体情况选择合适的治疗方法，以降低水肿对患者的影响。

5.5.4 血肿、治疗方法的因果推断分析

在这一环节，我们需要研究患者血肿体积和不同治疗方法之间的关系，和研究患者水肿体积和不同治疗方法之间关系的目标类似。两者均是以研究多组别自变量和单组别因变量之间关系为任务，因此在对患者血肿体积和不同治疗方法之间的关系进行研究时，我们依然沿用前文中所建立的因果推断模型。同时，在对患者信息进行分析时，沿用第二小问

中对患者的亚分组结果，对不同亚组患者血肿体积和不同治疗方法之间的关系做细致化的研究。

将四个亚组患者预处理过后的信息分别带入因果推断模型，得到各组别患者治疗方法和血肿体积的检验结果如表5.10所示。

表 5.10　各亚组不同治疗方法对血肿体积进展模式的影响

治疗方法	亚族类别	t 值	p 值	平均数差
脑室引流:	0	-0.20787	0.838817	0.050949
...	...	...	...	...
止血治疗	1	1.125985	0.266141	-0.35201
降颅压治疗	1	1.450096	0.153966	-0.40649
...	...	...	...	...
镇静、镇痛治疗:	2	NAN	NAN	0.294945
...	...	...	...	...
止吐护胃	3	-0.16928	0.866548	0.11999
营养神经	3	-0.79125	0.434126	0.460704

经过两独立样本 t 检验的研究，对于患者采用的不同治疗方法与血肿体积大小之间的关系得到了以下结论：

1. 对于编号为 0 的亚组而言，镇静、镇痛治疗可能会导致轻微的降低意识障碍程度，p 值相对较小（0.0813668971466179）。由于患者采用的治疗方法较少，数据样本集不足以支持检验结果的产生，因此，需要更多的数据来进一步研究其它治疗方法对该亚组患者血肿体积大小的影响；
2. 对于编号为 1 的亚组而言，所采用的治疗方法均没有显示出明显的疗效，p 值较大，大于 0.05。这表明这些治疗方法在治疗组和对照组之间没有显著的效应；
3. 对于编号为 2、3 的亚组而言，所采用的治疗方法均没有显示出明显的疗效，p 值较大，大于 0.05。这表明这些治疗方法在治疗组和对照组之间同样没有显著的效应。

综上所述，不同治疗方法对患者的血肿体积大小有一定影响。对于采用止血治疗的患者，血肿体积大小存在统计学差异。因此，在治疗过程中需要根据患者的具体情况选择合适的治疗方法，以降低血肿对患者的影响。

6 问题三建模与求解

6.1 问题分析

出血性脑卒中患者的预后预测及关键因素探索有助于提高患者的生存率和生活质量，为临床相关决策提供依据。通过探索患者的个人史、疾病史、治疗方法及影像特征等关键因素，可以为患者提供更精准、个性化的治疗方案和护理措施。

在本问中，我们一共要完成三项任务：

1. 对前一百名患者的信息进行匹配，然后根据患者的个人史、疾病史、发病相关信息和首次影像结果，构建预测模型，预测患者 90 天 mRS 评分；
2. 根据前 100 个患者的所有已知临床、治疗信息和影像（首次和随访）结果，预测所有含随访影像检查的患者 90 天 mRS 评分；
3. 建立合适的数学模型，分析出血性脑卒中患者的预后（90 天 mRS）和个人史、疾病史、治疗方法及影像特征等关联关系，并依此为临床相关决策提供建议。

6.2 第一问模型的建立与求解

在本项任务中，我们需要通过患者的个人史、疾病史、发病相关（“表 1” 字段 E 至 W）及首次影像结果（表 2，表 3 中相关字段）构建预测模型，预测患者（sub001 至 sub160）90 天 mRS 评分。我们采用深度森林对患者 90 天 mRS 评分进行预测。首先我们先对数据进行了处理，对分类特征进行 one-hot 编码，连续特征归一化到 [0，1] 范围。

6.2.1 深度森林

深度森林是一种集成学习方法，它组合了多个决策树来进行预测 [16]。在这里，我们将使用深度森林模型进行预测。每个决策树将根据患者的个人史、疾病史、发病相关特征以及首次影像结果来进行预测。首先，深度森林模型的整体预测可以表示为：

$$Y_i = F(X_i) \tag{6.31}$$

其中,Y_i 是对第 i 个患者的 90 天 mRS 评分的预测值。X_i 是包含第 i 个患者的特征集，包括个人史、疾病史、发病相关特征以及首次影像结果。F 是深度森林模型，它是多个决策树的集成。深度森林模型中的每个决策树的预测可以表示为：

$$F_k(X_i) = \sum_{j=1}^{J_k} c_{kj} \cdot I(X_i \epsilon R_{kj}) \tag{6.32}$$

其中 $F_k(X_i)$ 是第 k 棵树的测值。J_k 是第 k 棵树的叶节点数量。c_{kj} 是第 k 棵树的第 j 个叶节点的值。$I(X_i \epsilon R_{kj})$ 是一个指示函数，如果第 i 个患者的特征集 X_i 位于第 k 棵树的第 j

个叶节点 R_{kj} 内，则值为 1，否则为 0。最终的预测值是所有决策树的预测值的平均值:

$$Y_i = \frac{1}{K}\sum_{k=1}^{K} F_k(X_i) \tag{6.33}$$

其中，K 是深度森林中决策树的数量。

6.2.2 模型的求解

如上文所述数据预处理方案，本文采用深度森林方式对首次影响检查数据和 90 天 mRS 评分进行建模，同样，我们首先将所有特征作为自变量，将 90 天 mRS 作为因变量进行拟合，通过该模型求解得到变量的重要程度，并选取了前十个重要的变量进行模型的二次拟合，我们的实验结果如表6.11所示。其中训练集和测试集为前 100 个病人 ID 按照 8:2 的比例进行随机采样（我们的具体结果详见补充材料中的表 4）。

表 6.11　问题三第一小问求解结果

	一次拟合		二次拟合	
	训练精度	测试精度	训练精度	测试精度
DeepForest (Ours)	0.85	0.45	0.87	0.55
LightGBMXT	0.67	0.35	0.70	0.40

从表中可以看出，我们采用的方案取得了最优的性能，同时我们发现，进行特征筛选后进行训练，模型的训练精度差异不大，但是在测试集上的精度有了较大的提升，这说明过多的特征容易造成模型的过拟合，导致在测试集上的精度下降较多。因此进行二次的特征筛选是很有必要的。

6.3 第二问模型的建立与求解

在本问中我们将根据前 100 个患者（sub001 至 sub100）所有临床、治疗（表 1 字段 E 到 W）、表 2 以及表 3 的影像（首次 + 随访）结果，去预测所有含随访影像检查的患者（sub001 至 sub100,sub131 至 sub160）90 天 mRS 评分。本问多引入了随访影像结果这一因素，其与首次影像结果存在时间层面的关联，因此再数据处理阶段我们添加了时间间断字段来引入了时间相关联的因素。我们通过建模 LSTM 网络对所有含随访影像检查的患者 90 天 mRS 进行预测。

6.3.1 LSTM 模型

LSTM 网络是一种常用于处理序列数据的循环神经网络（RNN）变种。LSTM 网络的关键在于它的记忆单元和门控机制，使其能够更好的捕捉和处理长期依赖性，适用于建立各种序列建模任务。LSTM 的主要组成结构为输入门 i_t、遗忘门 f_t 和输出门 o_t。LSTM 的

核心是细胞状态，它可以被看作是网络的” 记忆”。细胞状态可以在长期时间内保持信息，允许网络存储和访问过去的信息。

1. LSTM 单元的输入门（input gate）输入门决定要更新细胞状态的程度。它的计算通常由以下公式表示：$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$。
2. LSTM 单元的遗忘门: 遗忘门决定要从细胞状态中忘记多少信息。它的计算通常由以下公式表示:$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$。
3. 更新细胞状态: 细胞状态的更新通过以下公式完成：$C_t = C_t + C_t$。
4. LSTM 单元的输出门: 输出门决定要输出多少细胞状态的信息。它的计算通常由以下公式表示：$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$。
5. 隐藏状态的更新：最后，使用输出门和更新后的细胞状态来计算当前时间步的隐藏状态：$h_t = o_t \cdot \tanh(C_t)$。

6.3.2 模型的求解

由于本问使用了额外多次的随访结果，随访之间存在着时间关联性，因此，本问采用更加适合时序建模的 LSTM 模型对本问进行建模，和上一小问同样的，我们采用了二次拟合的方式进行建模，首先对所有特征进行拟合，在筛选出影响程度较大的十个变量，通过这十个变量进行二次拟合，我们的模型结果如表6.12所示。

表 6.12　问题三第二小问求解结果

	一次拟合		二次拟合	
	训练精度	测试精度	训练精度	测试精度
LSTM (Ours)	0.968	0.840	0.995	0.975
LightGBMXT	0.923	0.815	0.985	0.951

我们发现，在采用多次随访结果进行拟合后，测试精度和训练精度都达到了一个很高的水平，这表明对于水肿和血肿的建模来说，长时序的特征更加有利于对疾病程度的判断，相较于仅仅凭借首次影像的诊断，多次诊断结果精度提升了将近一倍。因此，病患有必要进行定期的检查一遍医生做出正确的决断。

6.4 第三问模型的建立与求解

在这任务中我们需要通过分析出血性患者的预后与其个人史、疾病史、治疗方法以及影像特征之间的关系来去更好的提供临床方面决策的建议，以此来帮助医疗的专业人员能更加了解影响预后的种种因素，并在为患者提供针对性的治疗方案。我们将通过两种方案来分析 mRS 影响因素。

6.4.1 相关性分析

我们通过计算病患个人信息、临床指标和 CT 影像信息与病患预后的 Pearson 相关系数来进行分析。对于每个特征 x_i，计算其与目标变量 y(即 90 天 mRS 评分) 的 Pearson 相关系数:

$$r_{x_i y} = \frac{\sum_{j=1}^{N}(x_{ij} - \bar{x_i})(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{N}(x_{ij} - \bar{x_i})^2}\sqrt{\sum_{j=1}^{N}(y_j - \bar{y})^2}} \tag{6.34}$$

其中,$\bar{x_i}$,$\bar{y}$ 分别为 x_i,y 的平均值。N 为样本数量。

我们选取了前 10 个相关性系数高的特征对其进行分析。从表6.13可以看出病患的血肿指标很大程度影响着 90 天 mRS。血肿的表面面积、体积、最大二维直径、轴长等，这些与病灶大小、扩展范围相关，与预后评分正相关。糖尿病史和冠心病史与预后也存在正相关, 可能是由于基础疾病会对神经系统造成影响。综合来看, 病灶图像特征是影响预后最关键的因素之一, 但不能忽视基础疾病和健康状况的综合影响。

表 6.13　影响病患预后的特征（TOP10）

Feature	Correlation
original_shape_SurfaceArea_Hemo	0.39930563864819446
HM_volume	0.3855776316216518
original_shape_Maximum2DDiameterColumn_Hemo	0.38484300696888557
original_shape_LeastAxisLength_Hemo	0.3726962035382604
original_shape_VoxelVolume_Hemo	0.3561706236369382
original_shape_MeshVolume_Hemo	0.3561098037359607
original_shape_MajorAxisLength_Hemo	0.34009470239168677
糖尿病史	0.3133283408213977
冠心病史	0.3131265691541535
NCCT_original_firstorder_Range_ED	0.30870457107435795

6.4.2 特征重要性

我们使用 LightGBM 等机器学习模型可以输出特征重要度 [17]，这些特征重要度指标通常用于衡量特征对目标变量（在这种情况下是 90 天 mRS 评分）的贡献程度。一般来说，特征重要度分数越高，表明该特征对预测目标变量的贡献越大。以下是一种数学建模的方式来描述特征重要性的计算过程：我们通过训练好的模型计算特征重要度分数。对于一个特征 X_i，它的特征重要度分数 FI 可以表示为:

$$FI(X_i) = \frac{\sum_j Gain(X_i, j) \cdot I(X_i, j)}{\sum_j Gain(X_i, j)} \tag{6.35}$$

其中，FI(X_i) 表示特征 X_i 的特征重要度分数。Gain(X_i, j) 表示特征 X_i 在模型的节点 j 分裂时带来的增益，通常可以通过基尼不纯度或均方误差的减小来衡量。$I(X_i, j)$ 是一个指示函数，如果特征 X_i 在节点 j 分裂中使用了，则为 1，否则为 0。N 是总的节点数。最后，我们根据特征重要度分数对所有特征进行排序，选择前几名的特征，这些特征被认为对于预测 90 天 mRS 评分的贡献最高。我们也根据特征重要度分数对影响患者预后因素进行了进一步分析。

表6.14给出了模型训练好后所得到的重要性前十得特征，我们可以得出以下几点：

1. 最重要的特征是糖尿病史, 提示糖尿病作为基础疾病会对预后产生重要影响。糖尿病会降低病患的免疫力，疾病所造成的损害不能及时的进行自我修复，导致病情恢复缓慢甚至变得更为严重。
2. 右前动脉血肿分布比例 (HM_ACA_R_Ratio) 也较重要, 血肿所在位置与严重程度相关。
3. 发病到影像时间间隔排第三, 发病越慢救治及时程度较好。
4. 年龄、低血压、吸烟史等个人信息特征位列前列。

表 6.14　模型特征重要性 (Top10)

feature	importance	stddev	p_value	n	p99_high	p99_low
糖尿病史 _1	0.03	0.00	0.00	5	0.04	0.02
HM_ACA_R_Ratio	0.03	0.00	0.00	5	0.03	0.02
发病到首次影像检查时间间隔	0.02	0.01	0.00	5	0.04	0.01
房颤史 _1	0.02	0.00	0.00	5	0.03	0.01
吸烟史 _1	0.02	0.00	0.00	5	0.03	0.01
低压	0.02	0.00	0.00	5	0.02	0.01
年龄	0.01	0.00	0.00	5	0.02	0.01
止吐护胃 _1	0.01	0	0.50	5	0.01	0.01
original_shape_Flatness_y	0.01	0.01	0.00	5	0.03	0.00
original_shape_Elongation_y	0.01	0.01	0.00	5	0.03	0.00

6.4.3 临床决策的建议

分析上述 feature importance 结果，影响血性脑卒中患者预后 90 天 mRS 评分的重要特征包括糖尿病史、右前动脉血肿分布比例、发病到影像时间间隔、年龄、低血压和吸烟史等个人信息特征。

为了保证患者的身体健康情况，临床决策应该综合考虑以下几点：

1. 针对有糖尿病史的患者，应加强对其糖尿病的管理和控制，包括合理控制血糖水平、定期检查、接受规范治疗等，以提高免疫功能和促进病情恢复；

2. 对于血肿分布在右前动脉的患者，需要密切监测其病情变化，及时采取适当的治疗措施，包括手术、药物治疗等，以减轻血肿对脑组织的损害；
3. 在治疗过程中，需要密切监测患者的病情变化，提供及时的治疗干预，包括药物治疗、康复训练等，以促进患者的康复；
4. 在治疗过程中，需要根据患者的年龄、血压情况和吸烟史等个人信息特征进行个体化的治疗决策。年龄较大、低血压患者需要密切监测血压和相关生命体征，并给予适当的支持治疗；吸烟者应引导他们戒烟，以减少吸烟对血管造成的损害。

换言之，临床决策应综合考虑患者的个人疾病史、血肿分布位置、治疗及时程度、年龄、血压、吸烟史等因素，制定个体化的治疗方案，以促进患者的康复和预防二次脑卒中的发生。

7 模型评价

7.1 灵敏度分析

本文涉及的大部分模型都不依赖于精细的调参，采用默认参数就能取得比较好的性能。然而，所提出的模型仍有部分需要人为调节，例如，亚类人群划分的聚类数，以及模型特征数量的选取等。因此，在这一部分，我们对这些设置进行灵敏度分析。

7.1.1 聚类数目的灵敏度分析

在第二问中，涉及到聚类数目的选择，本文的实验证明了在对患病人群进行聚类后，预测模型的指标会得到一定的提升，然而，聚类数量的选择对最终性能的增益仍然有一定的探索空间 [18]，因此，在这一部分我们划分了 3-8 个聚类数目，并对最终的预测结果进行评价。评价指标分成预测的性能和聚类的性能，其中预测的性能使用 RMSE 指标进行衡量，聚类的性能采用轮廓系数进行衡量，轮廓分数是一种聚类效果评估指标，用于衡量聚类的紧密度和分离度。它的计算方法如下：

- 对于每个样本 i，计算其与同一簇内所有其他样本的平均距离，记为 a(i)。
- 对于样本 i，计算其与最近不同簇内所有样本的平均距离，记为 b(i)。
- 轮廓系数 s(i) 定义为：s(i) = (b(i) - a(i)) / maxa(i), b(i)

最终的轮廓分数是所有样本的轮廓系数的平均值。轮廓分数的取值范围在-1 到 1 之间：如果 s(i) 接近 1，表示样本 i 聚类得很好，距离其同一簇内的其他样本较近，与不同簇的样本距离较远。我们的计算结果如表7.15和图7.11所示，

表 7.15　聚类数量灵敏度分析，**黑体**表示最佳结果，<u>下划线</u> 表示次优结果

聚类数	2	3	4	5	6	7	8
聚类得分	**0.67**	0.56	<u>0.62</u>	0.56	0.57	0.53	0.54
RMSE	16828.78	16544.75	**15740.17**	16460.35	16197.40	16547.04	<u>16101.55</u>

从表中的结果可以看出，我们的方案对于聚类数量的敏感度不高，调节聚类的数量无论是聚类的得分还是 RMSE 的预测指标变化都不太大，在预测水肿体积时表现了较强的鲁棒性。其中 4 聚类取得了次优的聚类得分和最优的预测得分，这说明聚类的好坏在一定程度上会影响到最后预测的 RMSE 值。

7.1.2 模型特征数量的灵敏度分析

在本文构建的模型中，对数据进行了降维，降维的方式通过计算特征重要性，然后选取前 k 个重要特种进行模型的进一步拟合，然后 k 作为一个超参数，需要进行人工调节，为此，和上一部分保持一致，我们仍采用第二问中的预测对 k 进行灵敏度分析，调节 k 的

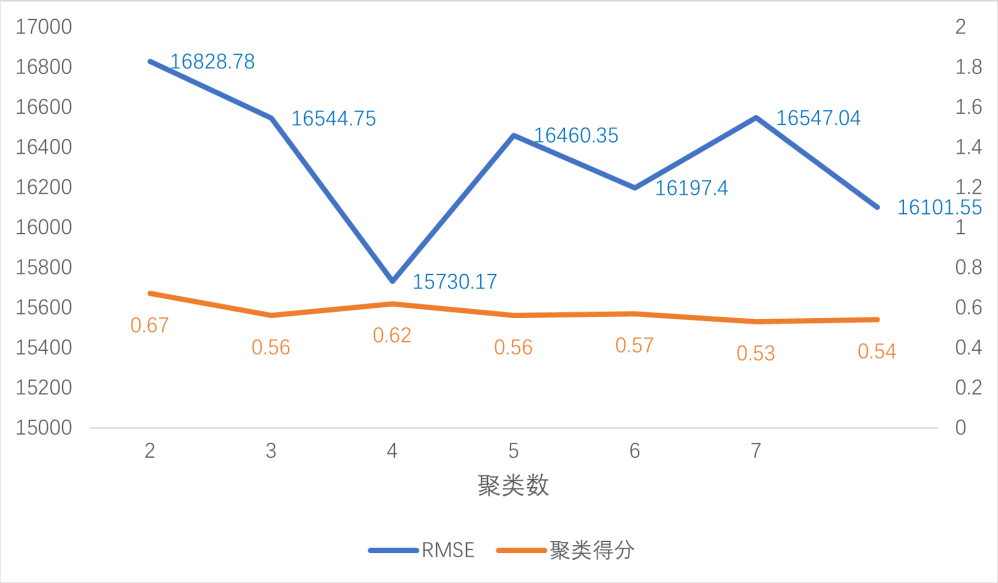

图 7.11　聚类数量灵敏度分析

范围为 5-19，和上一小节保持一致，我们采用聚类得分和预测得分衡量模型的综合性能，我们的灵敏度分析结果如表7.16和图7.12所示。

表 7.16　特征数量灵敏度分析，**黑体**表示最佳结果，下划线 表示次优结果

特征数量	5	7	9	11	13	15	17	19
聚类得分	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62
RMSE	16235.80	16876.47	15783.28	16641.38	15911.26	**15636.39**	17012.92	16197.91

从我们的结果中可以看出，特征数量 k 的选择对于聚类得分并没有明显的影响，但是对于模型预测能力有一定的影响，本文选择的特征数为 k=10，处于一个较优的配置，这也说明了本文参数选择的正确性，带来了比较好的模型性能。

7.2 模型优缺点分析

7.2.1 模型优点

1. 本文在问题一中首先将血肿扩散的条件进行符号化，根据条件构建了一个优化后的集成模型进行病人血肿扩散风险的预测，同时考虑了变量之间的冗余性，针对影响较高的变量进行建模，高效精准地完成了风险地评估。相较于单一的模型具有更好的精度和更强的泛化能力，达到了 99% 的预测精度。
2. 本文在问题二中，对水肿发生时间和水肿体积大小进行建模。考虑到同一个发生时间可能会对应多个水肿体积大小，本文同时考虑了绝对时间（当前的治疗时间）和相对时间（发病到就医的时间）。并采用机理建模（混合精度模型）和网络建模（双向 LSTM

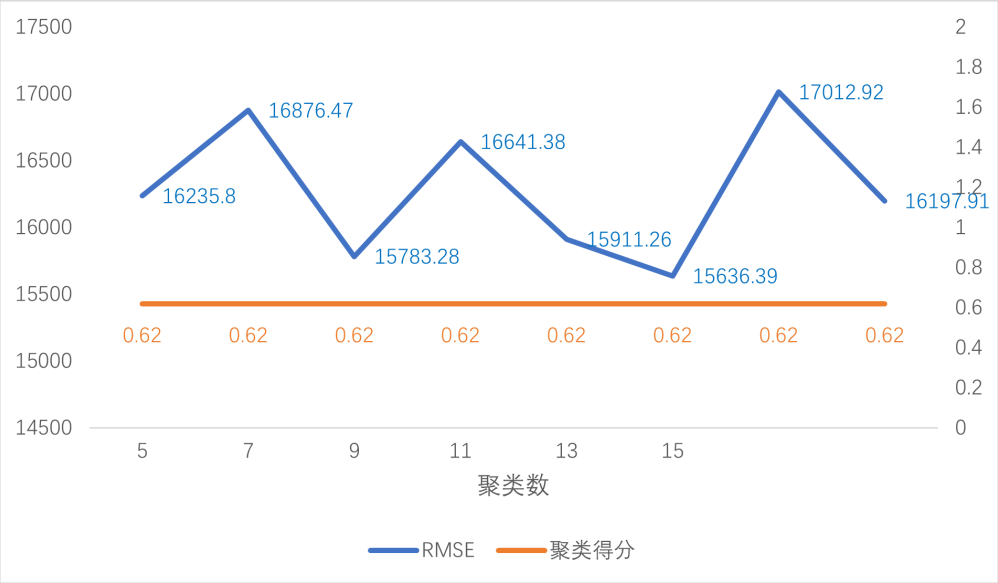

图 7.12　特征数量灵敏度

模型）相结合的方式，对水肿体积和残差进行建模，结果表明，这一建模方式能够有效应对一对多的数据分布问题。采用聚类的方法细分病患群体进一步提升了模型的预测性能，灵敏度分析的结果也表明本文的模型和超参数选择是鲁棒且高效的。

3. 本文在问题三中对首次就医和多次就医数据的 90 天 mRS 进行建模。同时考虑了水肿和血肿的影像数据，并结合病人的群体特征进行建模，首次和多次数据的拟合精度在测试集上达到了 55% 和 97.5%，在训练集上达到了 87% 和 99%，并根据我们的模型结果对相应的群体提供了接受治疗的参考意见。

7.2.2 模型缺点

1. 本文在问题二中，受限于因变量的选取，尽管采用绝对和相对时间变量，模型预测的 RMSE 仍然较大，还存在着数据一对多的问题。
2. 本文在问题三中，仅使用首次就医数据对于 90 天 mRS 的性能仍然有一定的提升空间。可以在未来继续集成或者设计针对性的模型。

参考文献

[1] 李立新, 李琳琳, 缺血性和出血性脑卒中危险因素分析, 郑州大学学报: 医学版(2): 313-315, 2010.

[2] 黄广苏, 邱小鹰, 陈汉明, 等, 脑出血早期血肿扩大的研究进展, 中国脑血管病杂志, 6 (1):54-56, 2009.

[3] 吴智平, 蒲传强, 脑出血后血肿扩大, 国外医学: 脑血管疾病分册, 12(9):672-674, 2004.

[4] 徐凤霞, 刘斌, 吴长鸿, 医学影像新技术在脑血管病诊断中的应用, 影像技术(3):44-46, 1996.

[5] 李如画, 医学影像在诊断过程中的作用, 影像研究与医学应用(1):69-70, 2018.

[6] Ke G, Meng Q, Finley T, et al., Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems, 30, 2017.

[7] 方匡南, 吴见彬, 朱建平, 等, 随机森林方法研究综述, 统计与信息论坛, 26(3):32-38, 2011.

[8] Zhang Z, Ersoz E, Lai C Q, et al., Mixed linear model approach adapted for genome-wide association studies, Nature genetics, 42(4):355-360, 2010.

[9] Siami-Namini S, Tavakoli N, Namin A S, The performance of LSTM and BiLSTM in forecasting time series, 2019 IEEE International conference on big data (Big Data), IEEE, 3285-3292, 2019.

[10] Yu Y, Si X, Hu C, et al., A review of recurrent neural networks: LSTM cells and network architectures, Neural computation, 31(7):1235-1270, 2019.

[11] Yan B, Aasma M, et al., A novel deep learning framework: Prediction and analysis of financial time series using CEEMD and LSTM, Expert systems with applications, 159:113609, 2020.

[12] Likas A, Vlassis N, Verbeek J J, The global k-means clustering algorithm, Pattern recognition, 36(2):451-461, 2003.

[13] 苗旺, 刘春辰, 耿直, 因果推断的统计方法, 中国科学: 数学, 48(12):1753-1778, 2018.

[14] 丁明, 张立军, 吴义纯, 基于时间序列分析的风电场风速预测模型, 电力自动化设备, 25 (8):32-34, 2005.

[15] Liu H, Tian H q, Li Y f, Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction, Applied Energy, 98:415-424, 2012.

[16] Zhou Z H, Feng J, Deep forest, National science review, 6(1):74-86, 2019.

[17] 徐少成, 李东喜, 基于随机森林的加权特征选择算法, 统计与决策(18):25-28, 2018.

[18] Hamerly G, Elkan C, Learning the k in k-means, Advances in neural information processing systems, 16, 2003.

附录 A 附录

附录中主要包含了核心的可执行的 python 代码，处理后的数据和所有代码（包括灵敏度分析和绘图代码）打包在补充材料中。

1.1 问题一第一小问

```python
import pandas as pd

data = pd.read_excel(r'\处理后的表2.xlsx')
time_data = pd.read_excel(r'\数据\竞赛发布数据\表1-患者列表及临床信息.xlsx')
df_result = pd.read_excel(r'\数据\竞赛发布数据\表4-答案文件.xlsx')
# time_data['发病到首次影像检查时间间隔'] = pd.to_timedelta(time_data['发病到首次影像检查时间间隔'], unit='h')

def query_database(patient_id, data, time_data = time_data):

  volumes_q = data.loc[data['ID'] == patient_id, 'HM_volume']
  times_q = data.loc[data['ID'] == patient_id, '检查时间']
  time_add = time_data.loc[time_data['Unnamed: 0'] == patient_id, '发病到首次影像检查时间间隔']

  return volumes_q, times_q, time_add.values

def get_volumes(patient_id, data):
    # 根据patient_id从数据库中查询影像体积数据
    volumes, times, time_add = query_database(patient_id, data)
    return volumes, times, time_add

def check_expanded(volumes_c, times_c, time_add, first):
    first_vol = volumes_c[first]
    first_time = times_c[first]
```

```
    len_ = len(volumes_c)

    expanded = 0
    time = []
    delta_volume = []

    for i in range(first + 1,first + len_):
        if ((volumes_c[i] - first_vol)/first_vol >= 0.33 or
           volumes_c[i] - first_vol >= 6000) and ((times_c[i
           ] - first_time).total_seconds() + time_add *
           3600) / 3600 <= 48:
             expanded = 1
             time.append(((times_c[i] - first_time).
                total_seconds() + time_add * 3600) / 3600)
             delta_volume.append(volumes_c[i] - first_vol)
    if expanded:
        max_volume_index = delta_volume.index(max(
           delta_volume))
        final_time = time[max_volume_index][0]
    else:
        final_time = 0
    return expanded, final_time, first + len_

expands = []
times_ = []
first = 0
for index, row in df_result.iterrows():
    if index in range(2,102):
        patient_id = row.iloc[0]
        volumes, times, time_add = get_volumes(patient_id,
           data)
        expand, time, first= check_expanded(volumes, times,
           time_add, first)
        df_result.iloc[index, 2] = expand
```

```
        df_result.iloc[index, 3] = time

df_result.to_excel(r'\数据\竞赛发布数据\表4-答案文件更新后.xlsx', index=False)
```

1.2 问题一第二小问

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from autogluon.tabular import TabularDataset, TabularPredictor
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import random

def load_data():
    train_data = pd.read_excel('q1_concat1.xlsx')
    train_label = pd.read_excel(r'数据\竞赛发布数据\表4-答案文件更新后.xlsx', skiprows=2)
    label = train_label['1是，0否']
    label_train = label[:100]
    label_test = label[100:]
    feature = train_data.iloc[:, 4:]
    feature_train = feature[:100]
    feature_test = feature[100:]
    return feature_train, label_train, feature_test, label_test

def preprocess_data(feature_train, label_train):
    train_set = pd.concat([feature_train, label_train], axis=1)
    label_name = train_set.columns[-1]
```

```
    raw_train = train_set.rename(columns={label_name: 'class'})
    train_set_1 = raw_train[raw_train['class'] == 1]
    train_set = pd.concat([raw_train, train_set_1, train_set_1], axis=0)
    train_set = train_set.sample(frac=1)
    return train_set

def train_autoML(train_data, label = 'class'):
    train_data = TabularDataset(train_data)
    predictor = TabularPredictor(label=label).fit(train_data)
    return predictor

def evaluate_model(predictor, raw_train, label = 'class'):
    acc = accuracy_score(predictor.predict(raw_train), raw_train[label])
    return acc

def main():
    feature_train, label_train, feature_test, label_test = load_data()
    train_set = preprocess_data(feature_train, label_train)

    # AutoML training
    predictor = train_autoML(train_set)

    leaderboard = predictor.leaderboard(silent=True)
    leaderboard.to_excel('q1_b_leaderboard.xlsx')

    total_feature = pd.read_excel('q1_concat1.xlsx')
    prob = predictor.predict_proba(total_feature)
    prob.to_excel('q1_b_prob.xlsx')

```

```
    feature_importance = predictor.feature_importance(
        train_set)
    feature_importance.to_excel('q1_b_feature_importance.
        xlsx')

    acc = evaluate_model(predictor, train_set)
    print(f'Accuracy of the train set is {acc}')

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

1.3 问题二第二小问

```
# kmeans
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.tsa.seasonal as seasonal
import matplotlib.pyplot as plt
from q2_a import *
from q1_b_model import train_autoML

def concat_feature(first_data_path, feature_path):
    first_data = pd.read_excel(first_data_path)
    feature = pd.read_excel(feature_path)
    id_col = feature.columns[0]
    concat_data = pd.merge(first_data, feature, right_on=
        id_col, left_on='ID', how='inner')
    concat_data['高压'] = concat_data['血压'].apply(lambda x
        : int(x.split('/')[0]))
    concat_data['低压'] = concat_data['血压'].apply(lambda x
```

```
        : int(x.split('/')[1]))
    concat_data.drop(['血压','Unnamed: 0_x','Unnamed: 0_y',
        '发病到首次影像检查时间间隔_x', 'time_gap',\
                        'time_from_start','数据集划分','入院首
                            次影像检查流水号'], axis=1, inplace
                            =True)
    start_index = concat_data.columns.tolist().index('检查时
        间')
    print(f'concat_data shape is {concat_data.shape}')
    return concat_data, start_index

def kmeans_cluster(data_in,choose_feature, n_clusters=3):
    data = data_in[choose_feature]
    data = data.dropna()
    data = data.drop(['检查时间'], axis=1)
    # scaler = MinMaxScaler()
    # data = scaler.fit_transform(data)

    # kmeans聚类
    kmeans = KMeans(n_clusters=n_clusters, random_state=0).
        fit(data)
    labels = kmeans.labels_
    data_in['label'] = labels
    data_in['label'].value_counts()
    # 评价聚类效果
    score = silhouette_score(data, labels)
    print(f'kmeans score is {score}')
    id_col = data_in['ID']
    label_col = data_in['label']
    id2label = dict(zip(id_col, label_col))
    cluster_center = kmeans.cluster_centers_
    cluster_center = pd.DataFrame(cluster_center, columns=
        data.columns)
    return data_in, id2label, cluster_center, score
```

```
48
49
50
51 if __name__ == '__main__':
52     import os
53     save_dir = 'q2_b_results'
54     os.makedirs(save_dir,exist_ok=True)
55     feature_path = r'\数据\竞赛发布数据\表1-患者列表及临床信
       息.xlsx'
56     first_data_path = r'\first_data.xlsx'
57     concat_data, start_index = concat_feature(
       first_data_path, feature_path)
58     train_data = concat_data.iloc[:, 2:]
59     # train_data = concat_data.iloc[:, start_index:]
60     # train_data = pd.concat([concat_data['ED_volume'],
       train_data], axis=1)
61     label = 'ED_volume'
62     predictor = train_autoML(train_data, label)
63     truth_value = train_data[label]
64     pred_value = predictor.predict(train_data)
65     rmse = np.sqrt(mean_squared_error(truth_value,
       pred_value))
66     print(f'RMSE␣of␣the␣train␣set␣is␣{rmse}')
67     feature_importance = predictor.feature_importance(
       train_data)
68     feature_importance.to_excel(os.path.join(save_dir,'
       q2_b_feature_importance.xlsx'))
69     top_10_feature = feature_importance.iloc[:10, 0].index.
       tolist()
70     device = torch.device('cuda' if torch.cuda.is_available
       () else 'cpu')
71     best_mse = int(1e6)
72     save_dict = {}
73     cluster_num = []
```

```
    cluster_score = []
    predict_score = []
    for cluster in range(2,9):
        cluster_num.append(cluster)
        data_in, id2label, cluster_center, score =
           kmeans_cluster(concat_data, top_10_feature,
           n_clusters=cluster)
        cluster_score.append(score)
        data_in.to_excel(os.path.join(save_dir,f'
           q2_b_cluster_{cluster}.xlsx'), index=False)
        cluster_center.to_excel(os.path.join(save_dir,f'
           q2_b_cluster_{cluster}_center.xlsx'), index=False
           )
        print(f'cluster␣{cluster}␣finished')
        data_train = pd.read_excel(r'\q2_total_data.xlsx')
        data_train['label'] = data_train['ID'].map(id2label)
        mse_error = []
        need_col = ['ID', 'label', 'ED_volume', '
           time_from_start', 'time_gap', 'y_pred', 'y_truth'
           , 'diff']
        record_dataframe = pd.DataFrame(columns=need_col)
        for i in range(cluster):
            temp_data = data_train[data_train['label'] == i]
            # x =
            X = temp_data['time_gap'].values
            y = temp_data['ED_volume'].values
            x_from_start = temp_data['first_ED_volume'].
               values
            scaler_X = MinMaxScaler()
            scaler_y = MinMaxScaler()
            scaler_x2 = MinMaxScaler()
            X = scaler_X.fit_transform(X.reshape(-1, 1))
            y = scaler_y.fit_transform(y.reshape(-1, 1))
            x_from_start = scaler_x2.fit_transform(
```

```
            x_from_start.reshape(-1, 1))
100         X = np.concatenate((X, x_from_start), axis=1)
101
102         X = torch.tensor(X, dtype=torch.float32).reshape
                (-1, 2)
103         y = torch.tensor(y, dtype=torch.float32).reshape
                (-1, 1)
104         model = train_bilstm_model(X, y, num_epochs=100)
105         y_pred, y_truth = evaluate_bilstm_model(model, X
                , y, scaler_y)
106         mse_error.extend(np.sqrt((y_pred - y_truth)**2).
                tolist())
107         temp_dict = {'ID': temp_data['ID'].values, '
                label': temp_data['label'].values, 'ED_volume
                ': temp_data['ED_volume'].values, \
108                      'time_from_start': temp_data['
                         time_from_start'].values, '
                         time_gap': temp_data['
                         time_gap'].values, \
109                        'y_pred': y_pred.reshape(-1)
                           , 'y_truth': y_truth.
                           reshape(-1), 'diff': (
                           y_pred.reshape(-1) -
                           y_truth.reshape(-1)).
                           reshape(-1)}
110         temp_data = pd.DataFrame(temp_dict)
111         record_dataframe = pd.concat([record_dataframe,
                temp_data], axis=0)
112
113
114
115     mse_error = np.array(mse_error).mean()
116     predict_score.append(mse_error)
117     if mse_error < best_mse:
```

```
118             best_mse = mse_error
119             best_cluster = cluster
120             print(f'best␣cluster␣is␣{best_cluster},␣best␣mse
                    ␣is␣{best_mse}')
121
122
123         else:
124             print(f'not␣best␣cluster,␣best␣cluster␣is␣{
                    best_cluster},␣best␣mse␣is␣{best_mse},␣
                    current␣cluster␣is␣{cluster},␣current␣mse␣is␣
                    {mse_error}')
125         record_dataframe.sort_values(by='ID', inplace=True)
126         record_dataframe.to_excel(os.path.join(save_dir,f'
                q2_b_cluster_{cluster}_record.xlsx'), index=False
                )
127         x = record_dataframe['time_gap'].values
128         plt.figure(figsize=(10, 6))
129         plt.rcParams['font.sans-serif'] = ['SimHei']
130         plt.rcParams['axes.unicode_minus'] = False
131         plt.style.use('ggplot')
132         # sns.scatterplot(x=x, y=record_dataframe['y_truth
                '].values, label='真实分布',hue=record_dataframe
                ['label'].values, palette='Set2')
133         sns.scatterplot(x=x, y=record_dataframe['y_truth'].
                values, color='blue', label='真实分布')
134         sns.scatterplot(x=x, y=record_dataframe['y_pred'].
                values, color='red', label='预测分布')
135         plt.xlabel('time_gap')
136         plt.ylabel('水肿大小')
137         plt.title(f'{cluster}聚类下的真实值与预测值散点图')
138         plt.savefig(f'q2_b_cluster_{cluster}_scatter.png',
                dpi=480, bbox_inches='tight')
139         plt.show()
140         plt.close()
```

```
    save_dict = {'cluster_num':cluster_num, 'cluster_score':
        cluster_score, 'prediction score':predict_score}
    save_df = pd.DataFrame(save_dict)
    save_df.to_excel(os.path.join(save_dir,'聚类灵敏度分析.
      xlsx'))
```

1.4 问题二第三小问

```

import numpy as np
import matplotlib.pyplot as plt
# from causalnex.match import Match
from causalnex.structure.notears import from_pandas
from causalnex.inference import InferenceEngine
from causalnex.network import BayesianNetwork
from causalnex.plots import plot_structure, NODE_STYLE,
   EDGE_STYLE
from causalnex.structure import StructureModel
import os
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from scipy.stats import ttest_ind
from sklearn.preprocessing import StandardScaler

def extract_feature(data, start_name, end_name):
    start_index = data.columns.tolist().index(start_name)
    end_index = data.columns.tolist().index(end_name)
    feature = data.iloc[:, start_index:end_index+1]
    id2feature = {}
    for i in range(feature.shape[1]):
        id = data['Unnamed: 0'].values
        temp_feature = data.iloc[:, start_index + i].values
        temp_feature_name = data.columns.tolist()[
           start_index + i]
```

```
26          id2feature[temp_feature_name] = dict(zip(id,
                temp_feature))
27      return feature, id2feature
28
29
30  def causal_analysis(data, start_name, end_name, save_path=
        None, y_name='ED_change_rate'):
31      feature = data.loc[:, start_name:end_name]
32      treat_ment_name = feature.columns.tolist()
33      y_name = y_name
34      donot_need_name = ['ED_change_diff','首次检查流水号','ID
            ','delta_ED_volume']
35      other_name = [i for i in data.columns.tolist() if i not
            in treat_ment_name and i != y_name and i not in
            donot_need_name]
36      # scaler = StandardScaler()
37      # data[other_name] = scaler.fit_transform(data[
            other_name])
38      # # data = data.dropna()
39
40
41      # X = data[treat_ment_name + other_name]
42      # y = data[y_name]
43      # reg = LinearRegression().fit(X, y)
44      # r2 = r2_score(y, reg.predict(X))
45      # print(f'r2 is {r2}')
46      if save_path is not None:
47          os.makedirs(save_path, exist_ok=True)
48          save_path = os.path.join(save_path, 'causal_analysis
                .txt')
49          with open(save_path, 'a') as f:
50              for i in range(len(treat_ment_name)):
51                  control = data[data[treat_ment_name[i]] ==
                        0]
```

```
52                  treatment = data[data[treat_ment_name[i]] ==
                        1]
53                  if control.shape[0] > 0 and treatment.shape
                        [0] > 0:
54                      trt = treat_ment_name[i]
55                      t, p = ttest_ind(control[y_name],
                            treatment[y_name])
56                      print(f'{treat_ment_name[i]}: {trt}, t
                            is {t}, p is {p}')
57                      print(f'mean diff is {np.mean(treatment[
                            y_name]) - np.mean(control[y_name])}'
                            )
58                      f.write(f'{treat_ment_name[i]}: {trt}, t
                            is {t}, p is {p}, mean diff is {np.
                            mean(treatment[y_name]) - np.mean(
                            control[y_name])}\n')
59
60                  else:
61                      print(f'{treat_ment_name[i]}: {trt}, no
                            control or treatment')
62                      f.write(f'{treat_ment_name[i]}: {trt},
                            no control or treatment\n')
63
64
65
66  if __name__ == '__main__':
67      q2_data = pd.read_excel('q2_total_data_all.xlsx')
68      feature_data = pd.read_excel(r'\数据\竞赛发布数据\表1-患
            者列表及临床信息.xlsx')
69      start_name = '脑室引流'
70      end_name = '营养神经'
71      feature, id2feature = extract_feature(feature_data,
            start_name, end_name)
72      for key in id2feature.keys():
```

```
73          q2_data[key] = q2_data['ID'].map(id2feature[key])
74      q3_data_first = q2_data.groupby('ID').first().
          reset_index()
75      q3_data_last = q2_data.groupby('ID').last().reset_index
          ()
76      ED_change_diff = q3_data_last['ED_volume'] -
          q3_data_first['ED_volume']
77      q3_data_first['ED_change_diff'] = ED_change_diff
78      ED_change_rate = q3_data_last['ED_volume'] /
          q3_data_first['ED_volume']
79      q3_data_first['ED_change_rate'] = ED_change_rate
80      feature_data = pd.concat([feature_data['Unnamed:␣0'],
          feature_data.loc[:, '年龄':'血压']], axis=1)
81      q3_data_first.drop(['Unnamed:␣0'], axis=1, inplace=True)
82      q3_data_first = pd.merge(q3_data_first, feature_data,
          left_on='ID', right_on='Unnamed:␣0', how='inner')
83      q3_data_first['高压'] = q3_data_first['血压'].apply(
          lambda x: int(x.split('/')[0]))
84      q3_data_first['低压'] = q3_data_first['血压'].apply(
          lambda x: int(x.split('/')[1]))
85      q3_data_first.drop(['血压'], axis=1, inplace=True)
86
87      # table2 = pd.read_excel(r'\处理后的表2.xlsx')
88      # table2 = table2.sort_values(by=['ID', '检查时间'])
89      # table2_first = table2.groupby('ID').first().
          reset_index()
90      # table2_first = pd.concat([table2_first['ID'],
          table2_first.loc[:, 'HM_volume':'
          ED_Cerebellum_L_Ratio']], axis=1)
91      # q3_data_first = pd.merge(q3_data_first, table2_first,
          on='ID', how='inner')
92      # q3_data_first.drop_duplicates(subset=['ID'], keep='
          first', inplace=True)
93      q3_data_first.drop(['检查时间','发病到首次影像检查时间间
```

```
        隔_x','time_from_start','Unnamed:␣0',\
94                          'time_gap'], axis=1, inplace=True)
95      sex_map = {'男': 1, '女': 0}
96      q3_data_first['性别'] = q3_data_first['性别'].apply(
          lambda x: int(sex_map[x]))
97
98      # sm.to_excel('q3_causal.xlsx')
99
100     q3_data_first.to_excel('q3_data_first.xlsx', index=False
          )
101
102     causal_analysis(q3_data_first, start_name, end_name,
          save_path='q3_causal_analysis')
103     # q2_data.to_excel('q3_total_data.xlsx', index=False)
```

1.5 问题三第二小问

```
1  import pandas as pd
2  import numpy as np
3  import torch
4  import torch.nn as nn
5  import torch.optim as optim
6  from sklearn.model_selection import train_test_split
7  from torch.utils.data import DataLoader, TensorDataset
8  from tensorflow.keras.preprocessing.sequence import
     pad_sequences
9  from torch.optim.lr_scheduler import CosineAnnealingLR
10 from sklearn.decomposition import PCA
11 from sklearn.preprocessing import PolynomialFeatures
12
13 # 检查GPU的可用性
14 device = torch.device("cuda" if torch.cuda.is_available()
     else "cpu")
15
16 data = pd.read_excel('/braindat/lab/wuxl/code/shumo/
```

```
    jianmo2023/竞赛发布数据/第三问b_2.xlsx')

# 创建一个空的时间序列数据集
time_series_dataset = []

# 初始化变量以跟踪当前ID和连续计数
current_id = None
count = 0
time_series = []

# 遍历数据
for index, row in data.iterrows():
    # 获取当前行的ID
    current_row_id = row['ID']

    # 如果ID与上一个数据点相同，将其添加到当前时间序列
    if current_row_id == current_id:
        current_data = row.drop(['ID'])  # 去除ID列
        time_series.append(current_data.values)
        count += 1
    else:
        # 如果ID与上一个数据点不同，检查时间序列的长度
        if count >= 1:  # 只添加包含1到9个数据点的时间序列
            time_series_dataset.append(time_series.copy())
        # 重置时间序列和计数
        time_series = [row.drop(['ID']).values]
        count = 1
        current_id = current_row_id

# 添加最后一个时间序列（如果有的话）
if count >= 1:
    time_series_dataset.append(time_series.copy())

time_series_data = np.array(time_series_dataset)
```

```
50
51 # 找到最大的时间步长
52 max_length = max(len(seq) for seq in time_series_data)
53
54 # 使用pad_sequences填充数据
55 padded_data = pad_sequences(time_series_data, maxlen=5,
       dtype='float32', padding='post', truncating='post')
56
57
58 # 准备输入数据和目标数据
59 X = padded_data[:, :, :-1]  # 输入数据，排除'90天mRS'字段
60 y = padded_data[:, 0, -1]   # 目标数据，仅包含'90天mRS'字段
61
62 X_2d = X.reshape(X.shape[0], -1)
63 pca = PCA(n_components=30)  # 选择要保留的主成分数量
64 X_pca = pca.fit_transform(X_2d)
65
66 # 将数据拆分为训练集和测试集
67 X_train, X_test, y_train, y_test = train_test_split(X_pca, y
       , test_size=0.2, random_state=42)
68
69 # 将数据转换为PyTorch张量
70 X_train = torch.Tensor(X_train).to(device)
71 y_train = torch.Tensor(y_train).to(device)
72 X_test = torch.Tensor(X_test).to(device)
73 y_test = torch.Tensor(y_test).to(device)
74
75
76 # 创建数据加载器
77 train_dataset = TensorDataset(X_train, y_train)
78 train_loader = DataLoader(train_dataset, batch_size=100,
       shuffle=True)
79
80 # 定义LSTM模型
```

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers=1, dropout_prob = 0.2):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, 7)
        self.dropout = nn.Dropout(dropout_prob)

    def forward(self, x):
        out, _ = self.lstm(x)
        # out = self.fc(out[:,-1. :])  # 获取LSTM的最后一个时间步的输出
        out = self.fc(out[:, :])  # 获取LSTM的最后一个时间步的输出
        return out

# 初始化模型
input_size = X_train.shape[1]
hidden_size = 2048
num_layers = 1

model = LSTMModel(input_size, hidden_size, num_layers).to(device)

# 定义损失函数和优化器
criterion = nn.CrossEntropyLoss()  # 使用交叉熵损失
optimizer = optim.AdamW(model.parameters(), lr=0.001)

# 设置余弦学习率调度器
num_epochs = 100  # 根据需要调整总的训练周期数
scheduler = CosineAnnealingLR(optimizer, T_max=num_epochs)

# 训练模型
```

```
for epoch in range(num_epochs):
    total_loss = 0.0  # 定义总损失
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs.squeeze(), targets.long())
        loss.backward()
        optimizer.step()
        total_loss += loss.item()  # 累积损失
    scheduler.step()  # 更新学习率
    print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {
        total_loss/(num_epochs*100):.10f}')

from sklearn.metrics import accuracy_score
# 在测试集上进行预测
model.eval()
with torch.no_grad():
    test_outputs = model(X_test)
    y_pred = test_outputs.argmax(dim=1).cpu().numpy()  # 根
        据输出的概率值取最大的类别作为预测类别

y_test_cpu = y_test.cpu().numpy()
accuracy = accuracy_score(y_test_cpu, y_pred)

print(f"Accuracy: {accuracy}")
```