


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

学　校	上海大学
参赛队号	23102800086
队员姓名	1.张尊帅
	2.周生华
	3.郎志远

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： 出血性脑卒中临床智能诊疗建模

摘 要：

出血性脑卒中是一种病因复杂的脑血管疾病，该病起病急、病情恶化迅速、预后效果较差并且病死率较高，而且患者常常会遗留神经功能障碍。本文分析了出血性脑卒中血肿扩张风险相关因素，研究了血肿体积、水肿体积及治疗方法之间的关系，并进行了预后预测及关键因素探索。

针对问题一，本文对题目提供的数据进行了数据整合和特征提取，通过患者进行影像检查时的流水检查号确定对应的检查时间，根据首次影像检查时间以及发病到首次影像检查时间间隔确定患者的发病时间，计算后续影像检查与首次检查时血肿绝对体积变化与相对体积变化判断患者是否在 48 小时内发生了血肿扩张。基于血肿扩张事件所涉及到的多个特征利用利用多层感知机进行建模，通过迭代训练集确定损失较小且准确率较高的模型。

针对问题二，由于数据涉及到的特征较多我们使用 Z-Score 对数据集进行清洗，并对治疗方案进行了特征重编码以有效地表示多个治疗方案的组合。在进行曲线拟合时，为了得到更好地拟合曲线，我们分别采用多项式曲线拟合和高斯曲线拟合两种拟合方法通过比较拟合曲线的 R^2和 RMSE 我们选择了高斯曲线拟合。在对患者聚类时，我们采用了 GMM 和 K-Means 两种聚类方法，通过比较 K-Means 方法聚类效果更好，接着分别比较聚类簇数为 3、4、5 时的聚类效果，选择把患者划分为 4 个亚组。最后通过比较不同组合治疗方案下患者水肿、血肿的体积进展并观察血肿与水肿体积分布分析了血肿体积、水肿体积及治疗方法三者之间的关系。

针对问题三，我们使用多层感知机构建了一个多分类模型，在该模型中使用 ReLU 作为激活函数帮助模型学习非线性函数以避免梯度消失问题，并在最后一个全连接层使用 softmax 激活函数用于多分类任务，将原始的输出转化为各个类别的概率分布。接着根据患者的 90 天 mRS 评分个人史、疾病史、首次和后续随访期间的影像特征，构建一个预测模型用于预测出血性脑卒中患者的治疗方案，通过分析得出这些特征之间的关联关系并为临床相关决策提出了建议。

关键词： 多层感知机 高斯曲线拟合 K-Means 聚类 回归预测 关联分析

目录

一、 问题重述

1.1 问题背景

出血性脑卒中是指脑实质内血管非外伤性破裂导致的脑出血，占所有脑卒中病例的 10-15%。这种疾病的病因非常复杂，通常由于脑动脉瘤破裂、脑动脉异常等原因导致血液涌入脑组织，使脑部受到机械性损伤并引发一系列生理病理反应。出血性脑卒中脑具有突然发病、病情加重迅速、预后不佳等临床特征，急性期病死率高达 45-50%，并且患者会遭受严重神经功能障碍等后遗症。出血性脑卒中不仅让患者承受死亡的风险，还给社会与患者家庭带来沉重的健康和经济负担。因此，深入研究出血性脑卒中的发病风险，整合影像学数据换着临床信息及临床诊疗方案，使患者的预后预测更加精准，并以此为依据针对临床决策进行优化，具有重要的临床价值。

在出血性脑卒中后，血肿范围扩大是预后不良的重要危险因素之一。短时间内，由于脑组织受损和炎症反应等因素，血肿范围可能逐渐扩大，导致颅内压急剧升高，进一步损害神经功能，甚至威胁患者生命。因此，需要重点检测和控制血肿的扩张情况。另外，血肿周围的水肿在近年来引起广泛关注，作为脑出血后继发性损害的标志，血肿周围水肿可能导致脑组织受到压迫，进而影响神经元功能，加重脑损伤，更进一步损害患者的神经功能。所以对血肿扩展和血肿周围水肿的早期识别和预测有着非常重要的价值。

医学影像技术能够帮助我们动态监测出血性脑卒中后脑组织损伤，同时在医学领域我们可以使用人工智能技术对大量影像数据进行数据挖掘与智能分析。助脑出血患者的影像信息，结合患者的个人信息、治疗方案和预后数据，我们可以构建智能诊疗模型，明确导致出血性脑卒中预后不良的危险因素，实现精确的个性化治疗效果评估和预后预测。

1.2 问题提出

问题一：出血性脑卒中患者血肿扩张风险相关因素探索建模。

a) 根据患者的临床信息、患者血肿体积及位置影像信息判断患者发病后 48 小时内是否发生血肿扩张事件。

b) 以是否发生血肿扩张事件为目标变量基于患者的个人史，疾病史，治疗方案以及患者血肿的首次影像检查结果构建模型，预测所有患者发生血肿扩张的概率。

问题二：对出血性脑卒中患者血肿周围水肿的发生及进展建模，并探索治疗干预和水肿进展的关联关系。

a) 根据患者的水肿体积和重复检查时间点，构建一条全体患者水肿体积随发病至影像检查时间进展曲线，并计算真实值与拟合曲线之间存在的残差。

b)对患者进行分组，针对不同人群的构建水肿体积随时间进展曲线并计算真实值和拟合曲线之间的残差。

c) 研究不同治疗方法对水肿体积进展模式的影响。

d) 分析治疗方法、血肿体积、水肿体积三者之间的存在的关系。

问题三：对出血性脑卒中患者预后预测及关键因素探索。

a) 利用患者的个人史，疾病史，治疗方案和首次影像检查结果等数据构建预测模型，预测患者的 90 天 mRS 评分。

b) 根据患者的临床、治疗、所有影像结果预测所有含随访影像检查的患者 90 天 mRS 评分。

c) 分析出血性脑卒中患者的预后和个人史、疾病史、治疗方法及影像特征等关联关系，基于这些关系为临床相关决策提出建议。

二、 问题分析

2.1 问题一分析

a) 根据问题描述，需要根据"表 1"和"表 2"中的数据来判断患者 sub001 至 sub100 发病后 48 小时内是否发生血肿扩张事件。如果发生血肿扩张事件，还需要记录血肿扩张发生时间。首先，我们需要对题目提供的数据进行预处理，包括处理缺失值和异常值。接下来，根据问题要求，我们提取入院首次影像检查流水号、发病到首次影像检查时间间隔以及各时间点流水号及对应的 HM_volume 等特征。此外，我们还需要计算血肿体积前后的绝对体积变化和相对体积变化，以便判断是否发生血肿扩张事件。

b) 对于该问题，需要以是否发生血肿扩张事件为目标变量，使用"表 1"前 100 例患者（sub001 至 sub100）的个人史、疾病史、发病相关字段（字段 E 至 W）、"表 2"中的影像检查结果（字段 C 至 X）以及"表 3"中的影像检查结果（字段 C 至 AG）等特征来建模。首先，进行数据预处理，包括处理异常值和缺失值，并按时间顺序排序数据。接下来，提取个人史、疾病史、发病相关和影像检查结果等特征。然后，为了对血肿扩张事件进行预测，我们可以使用回归预测模型如多层感知机来建立模型，以得到每个患者发生血肿扩张的概率。在建模完成后使用模型损失率和精准率来判断模型是否合理。

2.2 问题二分析

a) 针对问题一中的数据，首先进行数据预处理，包括筛选不合适的数据并提取所需特征，如水肿体积（ED volume）和重复检查时间点等。将这些特征按时间顺序排序，并使用高斯曲线拟合、多项式拟合等曲线拟合算法拟合数据。然后，通过比较真实曲线数据与拟合曲线数据来计算残差。

b) 类似于问题二(a)，首先进行数据预处理，根据问题可以使用 K-Means、高斯混合模型等聚类算法对不同人群进行聚类。然后，绘制不同人群的水肿体积随时间进展的曲线，并使用多项式拟合等方法拟合这些曲线。最后，计算前 100 个患者（sub001 至 sub100）真实值与拟合曲线之间的残差。

c) 根据问题描述，需要分析不同治疗方法对水肿体积进展模式的影响。首先，提取水肿体积（ED volume）和不同治疗方法（"表 1"字段 Q 至 W）等特征数据。然后，根据这些数据判断不同治疗方式下的水肿体积变化情况等相关指标，进行统计分析和可视化分析。

d) 针对问题描述，需要分析血肿体积、水肿体积以及治疗方法（"表 1"字段 Q 至 W）三者之间的关系。首先，提取血肿体积、水肿体积和治疗方法的相关数据。然后，参考问题 c）使用进行统计分析和可视化分析，并总结三类特征之间的关系。

2.3 问题三分析

a) 根据问题描述，首先提取个人史、疾病史、发病相关和首次影像结果等特征数据。然后，使用多层感知机等预测算法进行建模，并预测患者的 90 天 mRS 评分。

b) 类似于问题三(a)，基于在三(a)中提取的特征数据增加随访时的影响结果特征，同样可以使用多层感知机等方法进行建模和预测，在建模完成后可以预测患者的 90 天 mRS 评分。

c) 利用问题三(b)所提取的数据，我们可以根据前 100 位患者（sub001 至 sub100）的 90 天 mRS 评分个人史、疾病史、发病相关信息、首次和后续随访期间的影像特征，构建一个预测模型用于预测出血性脑卒中患者的治疗方案，基于该模型为出血性脑卒中患者提供更精准的临床诊断策略。

三、 模型假设与符号说明

3.1 模型基本假设

(1) 患者在出血性脑卒中发病后没有发生其他严重疾病或并发症

(2) 出血性脑卒中患者的病情特征，如水肿和血肿体积，只受到已提供的病史、治疗方案等内部特征的影响，而不受其他外界因素的干扰

(3) 可以假设患者的临床评估（例如 90 天的 mRS 评分）是可靠的衡量指标，能够反映患者的预后情况。

(4) 提供的影像数据包括血肿和水肿的体积、位置以及形状特征，是经过高质量标定和处理的，不存在显著的测量误差。

3.2 符号说明

表 1 符号说明

符号	含义
ΔV	以首次检查为基准后续检查时血肿体积的变化
X_i	数据点 i 的特征向量
μ	数据集平均值
δ	数据集标准差
C_i	聚类时的第 i 个簇

四、 模型建立与求解

4.1 问题一模型建立与求解

4.1.1 是否发生血肿扩张判定

（1）问题分析

判定是否发生血肿扩张任务涉及到“表 1”、“表 2”和“附表 1”三个数据表格，这些表格包含了入院首次影像检查信息、发病到首次影像检查时间间隔、各时间点流水号和对应的 HM_volume 数据、以及流水号与时间的对应关系。我们的目标是判断患者是否在发病后 48 小时内发生了血肿扩张事件，并记录事件发生的时间。血肿扩张的定义是后续检查比首次检查绝对体积增加≥6 mL 或相对体积增加≥33%。

为了判断血肿扩张事件，我们需要将不同数据表格中的信息关联起来。具体来说，我们需要使用流水号从“附表 1”中查询相应影像检查时间点，并结合发病到首次影像时间间隔以及后续影像检查时间间隔，来判断当前影像检查是否在发病 48 小时内。

（2）数据整合与问题求解

为了解决问题，首先需要将来自不同数据表格的信息整合在一起，以建立一个综合数据集，便于后续的分析和决策。具体来说，从表 1 中提取到入院首次影像检查流水号和发病到首次影像检查时间间隔两个字段，从表 2 中提取到各个时间点的流水号和对应的 HM_volume（体积）数据几个字段，从附表 1 中提取到流水号、影像检查时间点和发病到首次影像检查时间间隔三个字段。使用流水号将上述三个数据结构关联在一起。通过流水号，我们可以将不同数据表格中的信息连接起来，以便更准确地判断是否发生血肿扩张事件。

相对时间计算

利用发病到首次影像检查时间间隔和影像检查时间点，计算相对时间。相对时间表示了从发病到首次影像检查时间间隔加上随后的影像检查时间点距离入院首次检查的时间差。这个相对时间将在后续步骤中用于判断每次影像检查是否在发病后 48 小时内。

血肿扩张事件判断

根据问题描述，血肿扩张事件的定义包括两个条件：

- 后续检查比首次检查绝对体积增加≥6 mL
- 或相对体积增加≥33%

我们将对每一次影像检查数据进行分析，计算体积变化，并根据上述条件来判断是否发生了血肿扩张事件。其中计算体积变化和相对体积增加百分比的公式如下：

$$\Delta V = V_{current} - V_{initial}$$

$$\text{Relarive Increase} = \frac{\Delta V}{V_{initial}} \times 100\%$$

根据上述判断方式，我们得到患者中发生血肿扩张与未发生血肿扩张的比例如图 1 所示

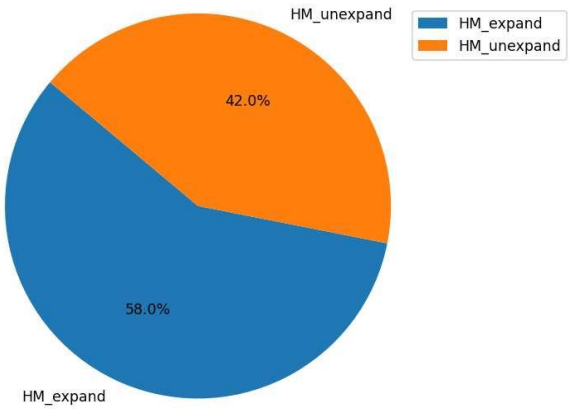

图 1 患者中是否发生血肿扩张人数比例

4.1.2 血肿扩张事件预测

血肿扩张事件预测任务是基于患者的个人史、疾病史、发病相关信息、治疗相关特征以及首次影像检查结果等变量，构建一个二分类模型，用于预测患者是否会在未来发生血肿扩张事件。

（1）模型数据准备

数据提取与筛选

首先从表 1、表 2 和表 3 中读取患者的临床信息、影像信息和形状及灰度分布信息，选择前 100 例患者数据作为训练数据。通过匹配患者流水号，从表 3 中筛选出与表 1 中的首次影像检查匹配的数据，这些数据将用于模型训练和预测。

异常值处理

表 1 中的血压数据和性别数据无法直接送入模型进行预测，我们进行了处理：对于血压数据，将其拆分成高压和低压两个部分，并分别存储。原始的血压数据列被移除，只保留高压和低压数据。对性别数据进行编码，将“男”编码为 1，将“女”编码为 0，以便模型能够处理性别信息。

归一化处理

由于不同特征的取值范围相差很大，这种尺度差异可能会导致模型在训练过程中对某些特征更加敏感，而对其他特征不敏感。此外数据归一化还可以带来加速收敛、提高模型稳定性和泛化能力以及改善可解释性等益处。因此对于取值不在[0,1]区间内的特征，我们进行了归一化操作。

数据集创建

我们将从表 1、表 2 和表 3 中提取筛选以及处理和归一化的数据作为特征，将任务 a)求得的是否发生血肿事件作为目标值来创建数据集。

（2）模型建立原理

由于预测血肿扩张事件涉及个人史、疾病史、发病相关及治疗相关等多种特征，我们选择了 MLP（多层感知器）作为模型架构，MLP 有助于处理多特征、非线性关系、数据归一化和概率预测等任务需求，使模型能够更好地适应和解决这个具体的预测任务。模型包含三个全连接层，用于学习输入特征之间的复杂关系，在每个全连接层之间，使用 ReLU 激活函数，它引入了非线性性质，有助于模型学习非线性函数，在最后一个全连接层之后，使用 softmax 来计算概率，由于是二分类任务，输出层有两个神经元。模型架构如图 2 所示：

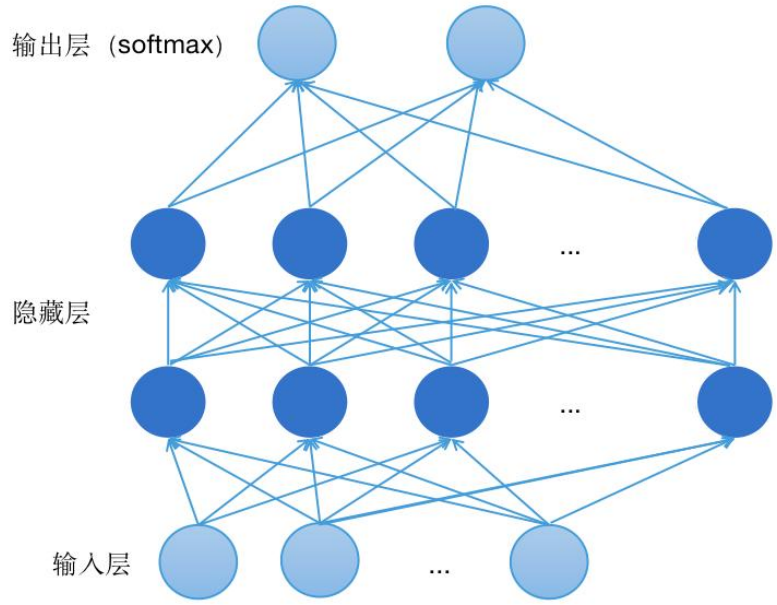

图 2 MLP 模型架构

我们对全连接层的权重进行 Xavier 初始化，Xavier 初始化有助于解决神经网络训练过程中的梯度问题，破坏对称性，保持激活函数的输出范围合理，并提高模型的训练速度和泛化能力。

模型使用交叉熵损失函数（CrossEntropyLoss）来度量模型输出与真实标签之间的差异。优化器选择了 Adam 优化器，用于自适应地调整模型参数以最小化损失函数。

模型通过迭代训练数据集来学习特征之间的关系和类别之间的决策边界。在每个迭代周期（epoch）中，模型计算损失，使用反向传播算法更新权重，然后继续下一轮迭代。这个过程会不断优化模型，使其逐渐提高分类性能。

此外，我们训练神经网络时加入了随机噪声，因为随机噪声可以提供正则化效应、增加模型的鲁棒性、帮助避免局部极小值、增加探索性，并提高模型的泛化能力。

（3）模型评估分析

模型的损失情况如图 3 所示：

图 3 模型损失情况

模型的准确率如图 4 所示：

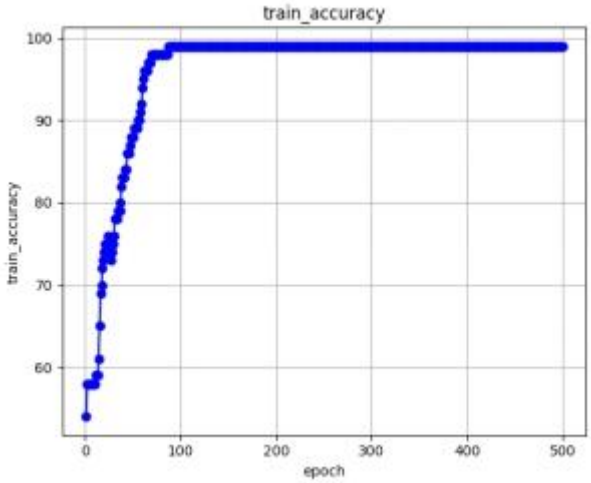

图 4 模型准确率

由图可以得知，我们建立的 MLP 模型可以迅速收敛，准确率稳定在 99%。

保存准确率最高模型训练参数后，使用该参数对目标值血肿是否扩张事件进行预测，预测值填在了表 4 的相应位置。

4.2 问题二模型建立与求解

4.2.1 数据预处理

一、异常数据处理：

根据题目描述，在患者在重复进行影像检查不能保证每次影像检查的数据均能反映水肿、血肿体积等相关特征的实际情况，因此数据中可能同时存在真实数据和具有干扰信息的异常数据，针对具有干扰信息的异常数据，我们基于 Z-Score（Z 值）对异常数据进行清洗。

Z-Score（Z 值）是一种常用于检测和清洗异常数据的统计方法。Z-Score 的计算可以帮助我们理解数据点相对于整个数据集的位置，以及它们的离散程度，它通过测量数据点与其均值的偏差来标准化数据，并将其表示为标准差的倍数。Z-Score 的计算公式如下：

$$Z = \frac{X - \mu}{\delta}$$

其中 Z 表示数据点的 Z-Score，X表示数据点的值，μ表示数据集的平均值，δ表示数据集的标准差。

我们首先计算每个数据点的 Z-Score，将 Z-Score 的绝对值大于某个阈值的数据点定义为异常值，根据阈值来清洗异常数据，可以选择将异常值从数据集中删除。

数据清洗前后患者水肿体积随时间变化的离散图如图 5 所示

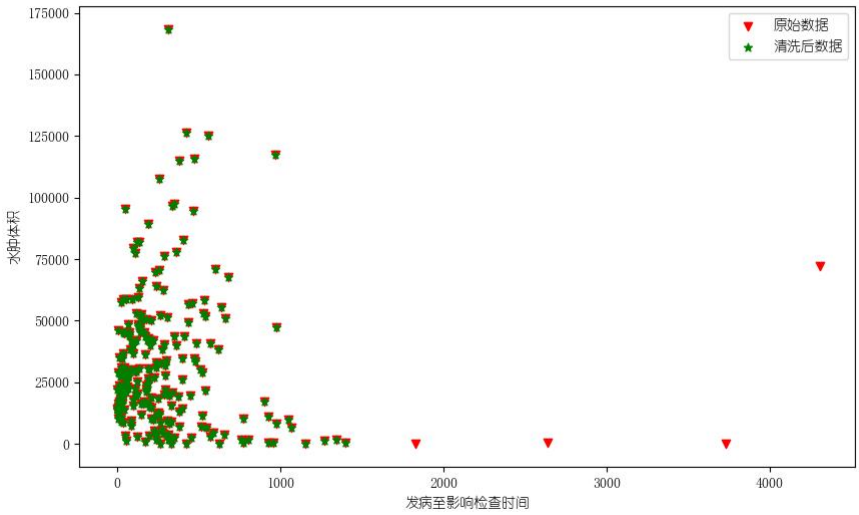

图 5 数据清洗前后患者水肿体积随时间变化对照

二、特征重编码

在表一中不同的治疗方案对应了数据集的多个特征，考虑到治疗方案之间的组合方式我们可以将治疗方案进行重编码，将其由多个特征组合为一个特征。本文采用二进制编码的方式将多个治疗方案特征重新进行编码，使用二进制编码中，每个方案特征都会被映射到一个二进制编码，其中每个二进制位代用 1 和 0 分别代表采用与不采用该种治疗方案，使用一串二进制编码可以有效地表示多个治疗方案的组合，并且不引入治疗方案之间的顺序关系。

4.2.2 曲线拟合

模型一：多项式曲线拟合

多项式曲线拟合是一种用多项式函数来逼近或拟合一组离散数据点的方法。通过选择合适的多项式阶数和系数，使得多项式曲线能够最好地拟合给定的数据点。

多项式函数通常表示为如下形式：

$$f(\mathrm{x}) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

其中，$f(\mathrm{x})$是多项式函数，x 是自变量，$a_0, a_1, a_2, \ldots, a_n$ 是多项式的系数，n 是多项式的阶数。每个系数对应于多项式中的一个幂次。

为了拟合多项式曲线，可以将问题转化为一个线性方程系统。考虑有 m 个数据点(x_i, y_i),需要找到多项式的系数$(a_0, a_1, a_2, \ldots, a_n)$使得多项式曲线$f(\mathrm{x})$尽可能地接近这些数据点。为每一个数据点$(x_i, y_i)$构建一个线性方程：

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n$$

这样就得到了一个包含 m 个线性方程的系统。将这些线性方程组合成矩阵形式：Y=XA。在此式中 Y 为包含所有y_i的列向量，X 为 m×(n+1)的矩阵，每一行对应于一个数据点，每一列包含对应x_i的不同幂次的项，A 是包含多项式系数的列向量。

为了找到最佳的多项式系数 A，可以使用最小二乘法来求解。最小二乘法的目标是最小化拟合误差，即数据点与多项式曲线之间的垂直距离的平方和。损失函数表示为：

$$L(A)=\sum_{i=1}^{m}(y_i-f(\mathrm{x}_i))^2$$

通过求解 $X^TXA=X^TY$ 就能够得到最佳的系数 A。

模型二：高斯曲线拟合

高斯曲线拟合是一种常用于数据分析和模型建立的方法，它基于正态分布（也称为高斯分布）的数学性质，通过找到一条或多条高斯曲线，最佳地拟合给定的数据分布。对一组数据(x,y)(i=1,2,3…)使用高斯函数进行描述：

$$y_i=y_{max}\times e^{-\frac{(x_i-x_{max})^2}{S}} \qquad (1)$$

(1)式中待估计参数y_{max}对应高斯曲线的峰值，x_{max}对应高斯曲线的峰值位置，S 表现出高斯曲线的半宽度信息。

(1)式两边同时取对数可化为：

$$\ln y_i=\ln y_{max}-\frac{(x_i-x_{max}^2)^2}{S}=\left(\ln y_{max}-\frac{x_{max}^2}{S}\right)+\frac{2x_ix_{max}}{S}-\frac{x_i^2}{S} \qquad (2)$$

令：

$$\ln y=z,\ \ln y_{max}-\frac{x_{max}^2}{S}=b,\frac{2x_{max}}{S}=b_1,-\frac{1}{S}=b_2 \qquad (3)$$

对于全部实验数据，以矩阵的形式可以将(2)式表示为：

$$\begin{bmatrix}z_1\\z_2\\z_3\\\cdots\\z_n\end{bmatrix}=\begin{bmatrix}1&x_1&x_1^2\\1&x_2&x_2^2\\1&x_3&x_3^2\\\cdots&\cdots&\cdots\\1&x_n&x_n^2\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\end{bmatrix} \qquad (4)$$

记为：Z = XB

构成矩阵 B 的广义最小二乘解为：

$$B=(X^TX)^{-1}X^TZ$$

根据（3）式求出待估参数y_{max}、x_{max}、S，得到高斯函数的特征参数

采用多项式曲线拟合我们得到拟合曲线的表达式为

$$f(x)=-3.096x+3.295\times10^4$$

拟合效果如图 6 所示

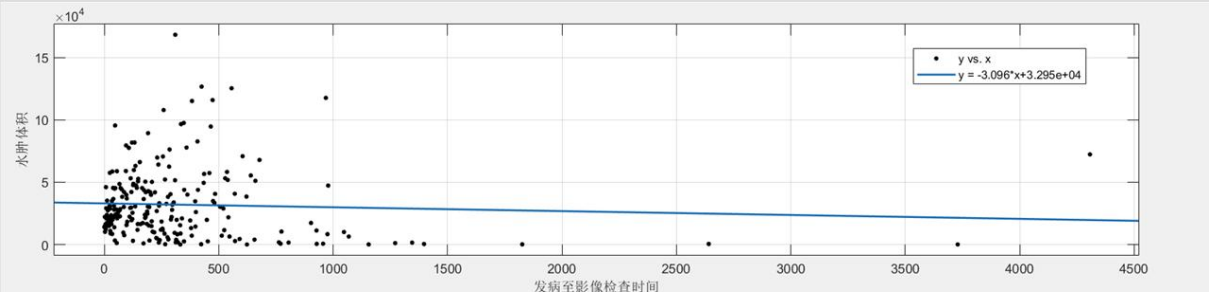

图 6 多项式曲线拟合结果

采用高斯曲线拟合我们得到拟合曲线的表达式为

$$f(x)=3.918\times10^4\times e^{-(\frac{x-412.8}{639.3})^2}$$

拟合效果如图 7 所示

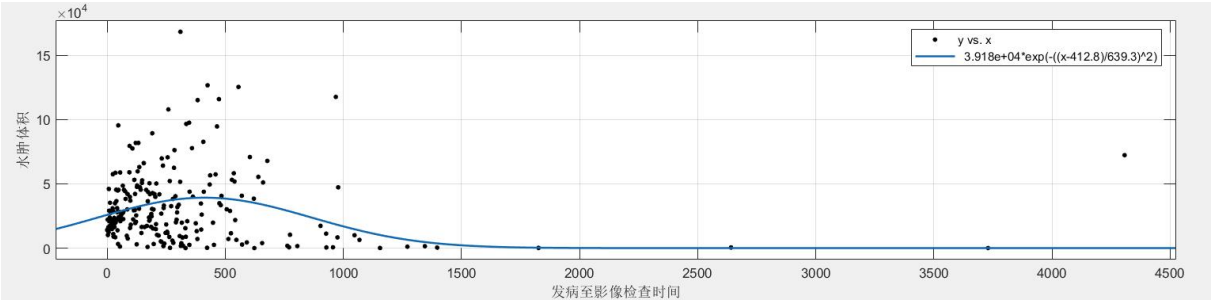

图 7 高斯曲线拟合结果

使用多项式曲线拟合时，拟合曲线的 R^2为 0.002796，RMSE 为 2.78×10^4；使用高斯曲线拟合时，拟合曲线的 R^2为 0.04927，RMSE 为 2.721×10^4，高斯曲线拟合的效果更好。此时前 100 个患者真实值和所拟合曲线之间的残差如图 8 所示

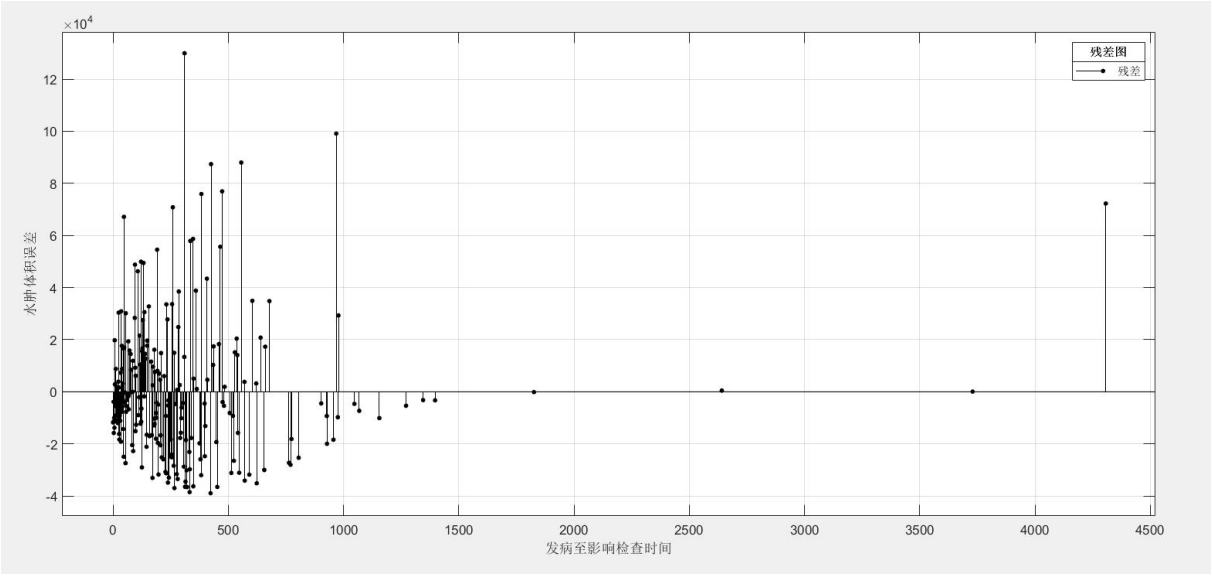

图 8 真实值与拟合曲线间的残差分布

4.2.3 聚类划分亚组

模型一：高斯混合聚类模型

高斯混合模型（Gaussian Mixture Model，GMM）是一种基于概率分布的聚类方法，通过用多个高斯分布函数去近似任意形状的概率分布，假设数据是由多个高斯分布组成的混合体。每个高斯分布成为一个“分量”，然后找到这些高斯分布的参数以及每个数据点属于每个分量的概率。

GMM 假设数据是由多个高斯分布组成的混合体，分量数表示高斯分布的个数。这个数量通常需要根据数据的性质来选择，可以是事先确定的，也可以通过模型选择方法估计得出，每个高斯分布都有其均值和协方差矩阵作为参数，用于描述分布的形状和位置使用混合系数代表每个高斯分布的权重，表示每个分量在整个数据集中的相对重要性，这些权重之和等于 1。

GMM 的聚类过程通常包括以下步骤：

1.初始化：随机初始化每个高斯分布的均值、协方差矩阵和混合系数。

2.使用 Expectation-Maximization(EM)算法：来迭代地估计模型参数。EM 算法包括两个步骤：

a.Expectation 步骤：使用贝叶斯定理和高斯分布的概率密度函数来计算每个数据点属于每个分量的后验概率，即计算每个数据点在每个高斯分布下的概率权重。

b.Maximization 步骤：在这一步中，根据 Expectation 步骤得到的后验概率，更新每个高斯分布的均值、协方差矩阵和混合系数，以最大化似然函数。

3.收敛判定：通常设置参数变化的阈值或最大迭代次数为收敛条件，根据收敛条件检查模型参数是否已经收敛。

4.结果输出：一旦模型收敛，输出每个高斯分布的均值、协方差矩阵和混合系数，以及每个数据点属于哪个分量的后验概率，根据后验概率将数据点分配到对应的簇。

图 9 为利用 PCA 对数据进行降维后使用 GMM 模型将患者聚类为 4 个亚组的结果：

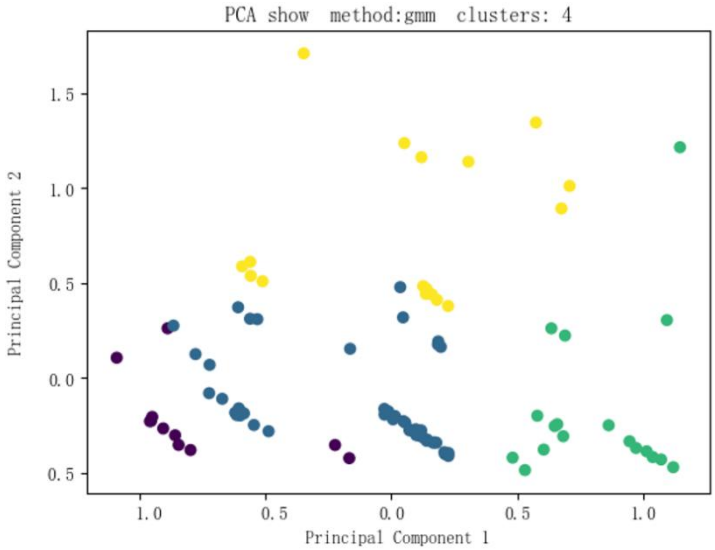

图 9 PCA 降维后 GMM 聚类结果

图 10 为利用 tsne 对数据进行降维后使用 GMM 模型将患者聚类为 4 个亚组的结果

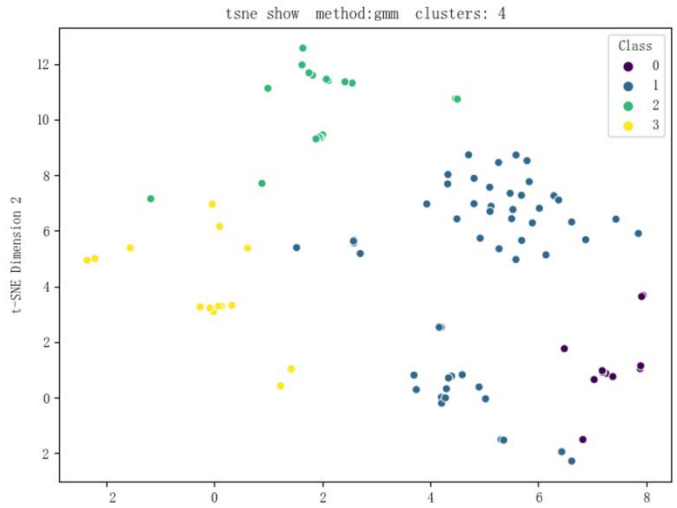

图 10 tsne 降维后 GMM、聚类结果

模型二：K-Means 聚类模型

K-Means 聚类（K-Means Clustering）是一种常用的无监督学习算法，用于将数据集中的观测点划分为 K 个不同的簇（cluster），每个簇代表一个数据群体。该算法的目标是最小化数据点与其所属簇内的聚类中心之间的距离，从而实现簇内数据的相似性最大化，簇间数据的相似性最小化。K-Means 聚类是一种迭代算法，通过不断地分配数据点到最近的聚类中心并更新聚类中心，最终实现数据集的聚类分析。

对于有一个包含 N 个数据点的数据集，其中每个数据点表示为 $x_1, x_2, ..., x^N$，每个数据点属于 K 个簇之一，使用 K-Means 方法进行聚类的步骤如下：

1. 初始化：选择 K 个聚类中心，通常表示为 $\mu_1, \mu_2, ..., \mu_k$。这些聚类中心是从数据集中随机选择的，或者可以通过其他初始化方法得到。

2. 分配数据点：对于每个数据点 x_i，计算其与 K 个聚类中心的距离，通常使用欧氏距离或其他距离度量方式。将数据点 x_i分配到距离最近的聚类中心所在的簇中。这可以表示为：

$$c^{(x_i)} = argmin\left(||x_i - \mu_j||^2\right), \text{其中 j} = 1,2, ...K$$

$c(x_i)$表示数据点 x_i所属的簇的索引，argmin 表示找到最小值的索引

3. 更新聚类中心：对于每个簇 j，重新计算其聚类中心 μ_j，这个聚类中心通常是该簇内所有数据点的均值：

$$\mu_j = \frac{1}{|C_j|}\sum_{x \in C_j} x$$

其中$|C_j|$表示簇$|C_j|$中数据点的数量。

4. 重复迭代：重复执行步骤 2 和步骤 3，直到满足停止条件，通常是达到最大迭代次数或聚类中心不再发生显著变化。

5. 输出结果：最终，每个数据点 x_i将分配到一个簇中，同时 K 个聚类中心 $\mu_1, \mu_2, ..., \mu_k$将作为聚类的结果输出。

在使用 K-Means 聚类时选择合适的 K 值直接影响到聚类的结果和质量，通常我们可以使用手肘法选择合适的 K 值。手肘法基于一个经验法则：在合适的 K 值处，簇内平方和或簇间距离的下降速度会减小，这对应着数据点更加紧密地分布在簇内。因此，选择 K 值就是为了找到这个拐点，这个拐点通常对应着聚类的自然分界点，所以我们通常选择拐点处的 K 值作为聚类的簇数。

图 11 为利用 PCA 对数据进行降维后使用 K-Means 聚类模型将患者聚类为 4 个亚组的具体效果：

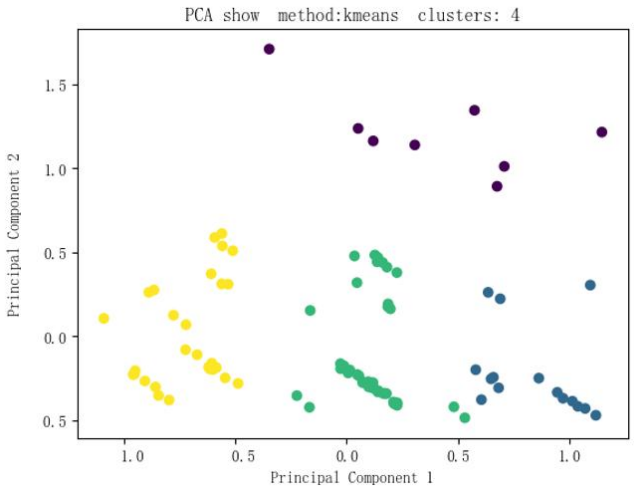

图 11 PCA 降维后 K-Means 聚类结果

图 12 为利用 tsne 对数据进行降维后使用 K-Means 聚类模型将患者聚类为 4 个亚组的具体效果：

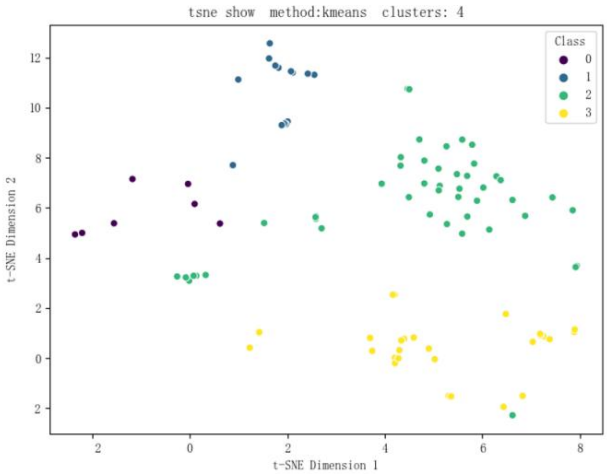

图 12 tsne 降维后 K-Means 聚类结果

对比之下我们发现在使用 K-Means 聚类模型是所划分的亚组在空间中更为集中，联系更加紧密，聚类效果要比 GMM 模型更好。接下来我们采用 K-Means 聚类模型分别将患者聚类为 3 个亚组和 5 个亚组。

将患者聚类为 3 个亚组时，聚类效果如图 13 所示：

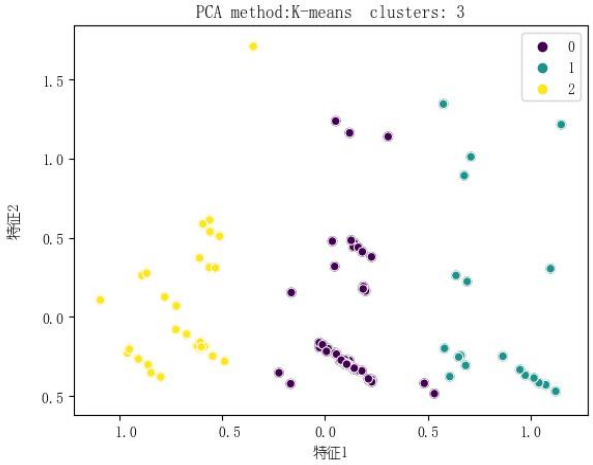

图 13 用 K-Means 将患者聚类为 3 个亚组

将患者聚类为 5 个亚组时，聚类效果如图 14 所示：

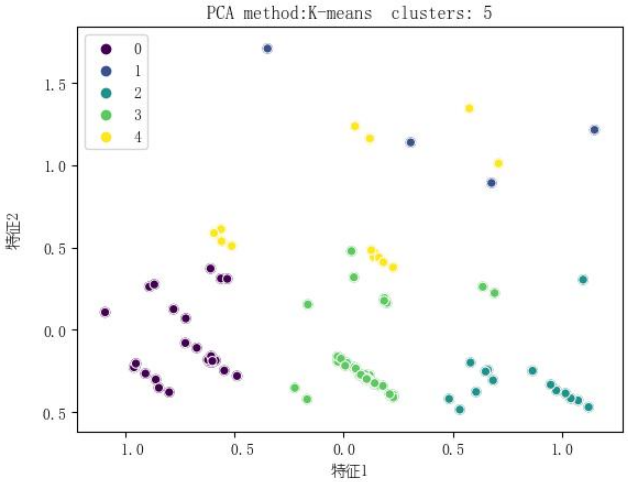

图 14 用 K-Means 将患者聚类为 5 个亚组

在将患者分为 3 个亚组时，每组会有部分患者特征差异较大，这在图 13 中表现为每类中有少数点距离其他点明显较远；在将患者分为 5 个亚组时，会出现不同组中的患者数量非常少的情况，这在图 14 中表现为有一类点明显比其他类的点更少。相比之下将患者聚类为 4 个亚组时效果最好，因此我们选择将患者聚类分为 4 个亚组。

聚类完成后不同人群的水肿体积随时间进展曲线如图 15 所示：

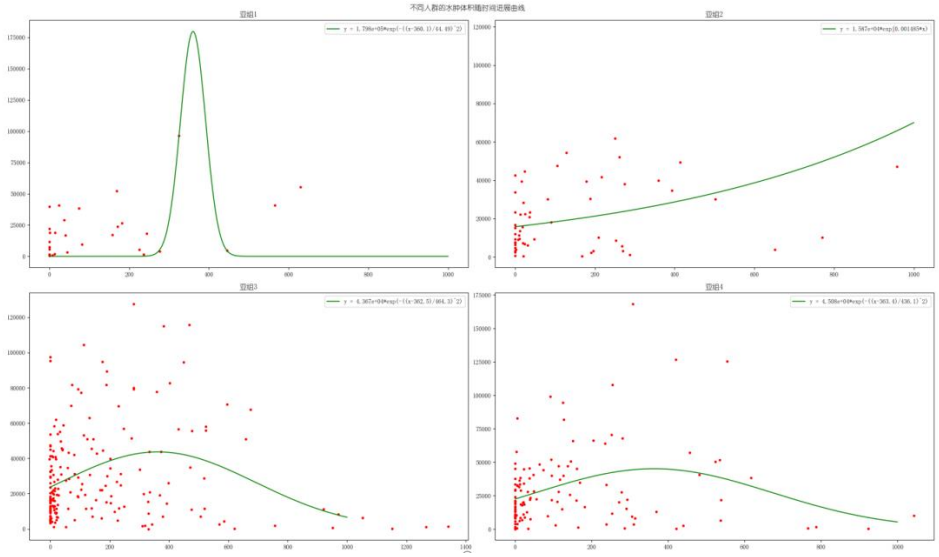

图 15 不同人群的水肿体积随时间进展曲线

4.2.4 治疗方法与血肿、水肿体积之间的关系

对 7 种治疗方案进行二进制编码后，每个患者采用的治疗方案组合都是一个 7 位的 01 串，从高位到低位分别表示脑室引流、止血治疗、降颅压治疗、降压治疗、镇静与镇痛治疗、止吐护胃、营养神经。采用不同组合治疗方法的患者水肿体积进展模式如图 16 所示：

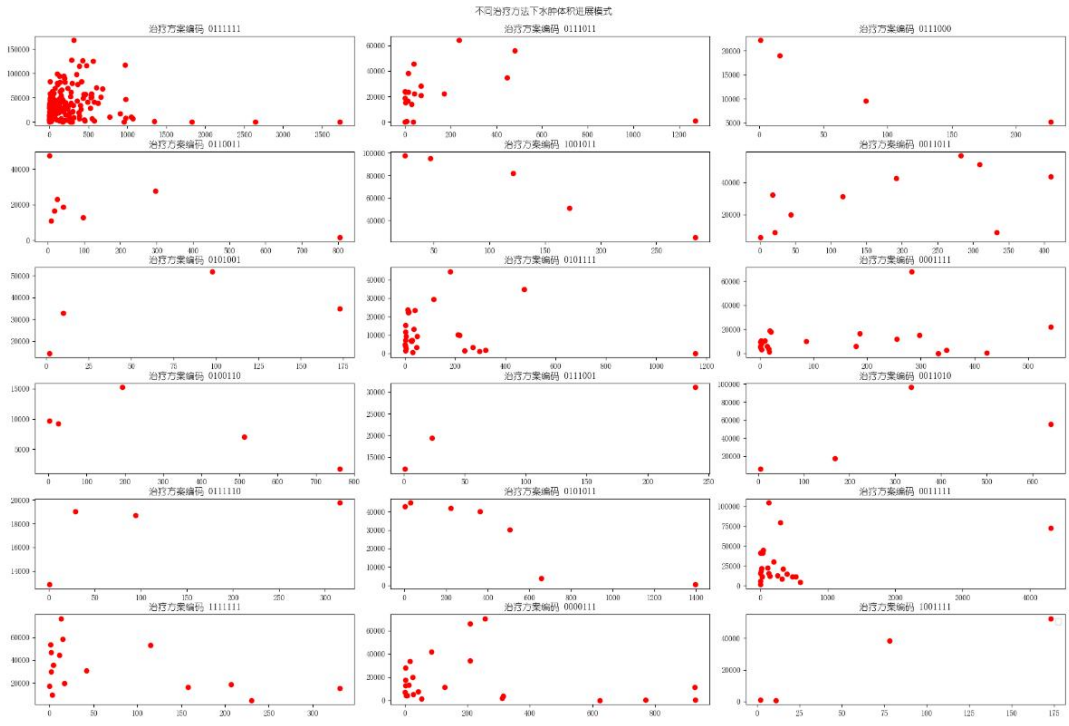

图 16 不同治疗方法下水肿体积进展模式

由于部分组合方案所治疗的患者人数过少，这里我们针对样本数多于 10 人的不同治疗方法对水肿体积进展模式的影响进行分析：

方案 0111111 采用止血治疗、降颅压治疗、降压治疗、镇静与镇痛治疗、止吐护胃、营养神经。使用这种组合方案进行治疗的患者最多，随着时间推移患者的水肿体积整体呈下降趋势，表示这种组合治疗方案效果较好。

方案 0111011 采用止血治疗、降颅压治疗、降压治、止吐护胃、营养神经。使用这种组合治疗方法的患者水肿体积大多数时间都呈增长趋势，其治疗效果明显不好。

方案 0110011 采用止血治疗、降颅压治疗、止吐护胃、营养神经。使用这种组合治疗方法的患者水肿体积略有起伏，但总体呈下降趋势。

方案 0011011 采用降颅压治疗、降压治、止吐护胃、营养神经。使用这种组合治疗方式的患者水肿体积在不断增大，不利于患者的治疗

方案 0101111 采用止血治疗、降压治疗、镇静与镇痛治疗、止吐护胃、营养神经。使用这种组合方式的患者水肿体积在不同时间点总有大有小，这种方法可能对患者水肿体积影响较小。

方案 0101001 采用止血治疗、降颅压治疗、营养神经方案的患者的水肿体积整体呈上升趋势，这种方案治疗效果较差。

方案 0001111 采用降颅压治疗、镇静与镇痛治疗、止吐护胃、营养神经方案的患者的水肿体积一直保持在相对较低的程度，这种方案比较有利于稳定患者的病情。

方案 0011111 采用降颅压治疗、降压治疗、镇静与镇痛治疗、止吐护胃、营养神经。采用这种方法的患者主要水肿体积在前期呈下降趋势。

方案 1111111 采用脑室引流、止血治疗、降颅压治疗、降压治疗、镇静与镇痛治疗、止吐护胃、营养神经。使用这种组合治疗方法的患者水肿体积呈下降趋势，但效果不如方案 0111111，表示脑室引流这种治疗方法一直水肿扩张的效果不好。

方案 0000111 采用镇静与镇痛治疗、止吐护胃、营养神经方案的患者的水肿体积在上升后呈下降趋势，表示这种方案风险较高，不适合在发病时水肿体积较大的患者。

采用不同组合治疗方法的患者血肿体积进展如图 17 所示：

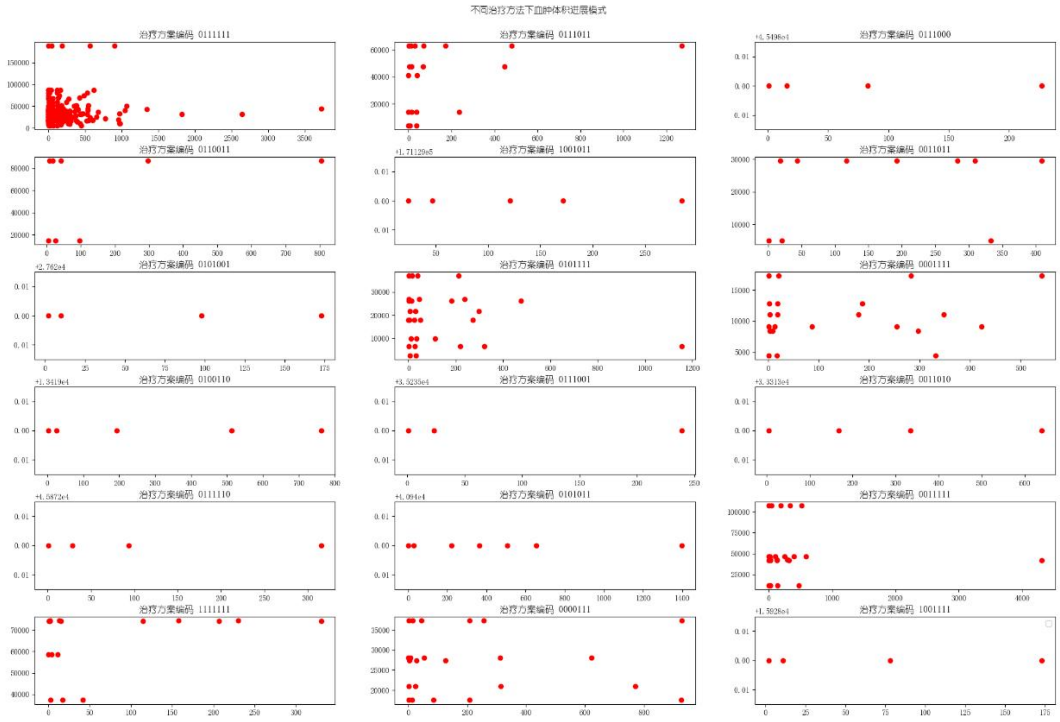

图 17 不同治疗方法下患者血肿体积进展模式

血肿体积随时间呈下降趋势的有方案 0111111；呈上升趋势的有方案 0111011、1111111；随时间分布较为均衡的有 0101111、000111、0011010、0000111。这表示脑室引流和镇静、镇痛治疗不能有效抑制血肿扩张。

血肿与水肿体积之间的关系如图 18 所示：

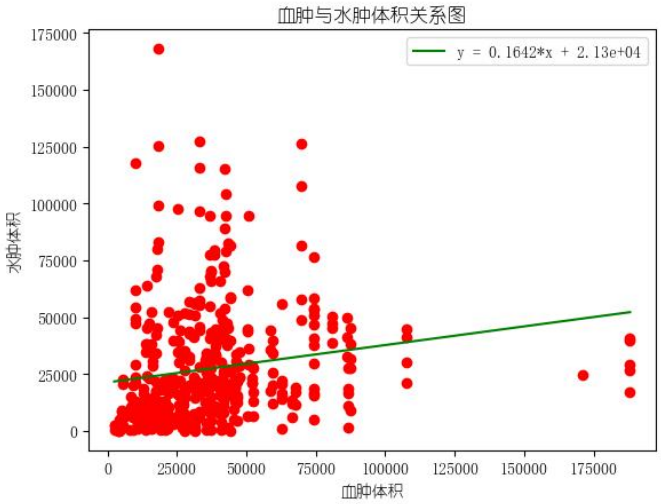

图 18 血肿与水肿体积关系图

可以看出血肿体积与水肿体积是呈正相关的。

综上所述，不同治疗方法可以对出血性脑卒中患者的水肿体积和血肿进展模式产生不同的影响。血肿体积与水肿体积成正相关，在不采用脑室引流方法治疗，同时采用剩余其他治疗方法时对血肿体积扩张和水肿体积扩张的抑制效果较好。

4.3 问题三模型建立与求解

4.3.1 基于患者个人史、疾病史、发病相关及影像结果预测 mRS

根据前 100 位患者的个人史、疾病史、发病相关信息以及首次影像结果，我们构建一个预测模型，用于预测所有患者在 90 天后的 mRS 评分。首先我们只考虑首次影像检查结果中的特征，构建一个回归任务，预测取值范围在 0 到 6 之间的有序等级变量，代表不同的 mRS 评分等级。在模型构建完成后，用类似的构建方法我们再加入随访时的影像检查结果中的特征，预测患者的 mRS 评分。

（1）模型数据准备

数据提取与筛选

首先，从表 1、表 2 和表 3 中提取患者的临床信息、影像信息、以及形状和灰度分布信息。然后，为了构建训练数据集，我们选择了前 100 位患者（sub001 至 sub100）的数据。这些数据将用于模型的训练。对于任务 a）通过匹配流水号，我们从表 3 中筛选出与表 1 中的首次影像检查匹配的数据，对于任务 b)还要加上所有患者随访的结果，这些数据将用于模型的训练和预测。

特殊值处理

对于表 1 中的血压数据，我们进行了处理：将血压数据拆分成高压和低压两个部分，并将原始的血压数据列移除，只保留高压和低压数据。此外，为了更好地处理性别信息，我们对性别数据进行了编码，将“男”编码为 1，将“女”编码为 0，以便模型能够处理性别信息。任务 b)中患者的随访次数不同，我们取前 7 次的随访结果，空缺值用零值填充。

归一化处理

鉴于不同特征的取值范围差异较大，我们进行了数据归一化处理，以确保这些特征具有相似的尺度。这一步骤旨在帮助模型更有效地学习特征之间的关系。确保所有特征的取值范围都在[0, 1]之间，以提高模型的性能和稳定性。

数据集创建

我们将从表 1、表 2 和表 3 中提取筛选以及处理和归一化的数据作为特征，将表 1 患者的 90 天 mRS 评分作为目标值来创建数据集。

（2）模型建立原理

为了有效地处理包含多种特征、非线性关系、数据归一化以及概率预测的任务需求，我们选择了多层感知器（MLP）作为模型架构。以下是详细的模型建立原理：

MLP 架构选择

我们采用了多层感知器（MLP）作为模型架构。MLP 是一种适用于多特征、多类别分类任务的神经网络结构。它由多个全连接层组成，具备处理复杂非线性关系的能力。

层次结构

我们的 MLP 模型包含三个全连接层。全连接层是神经网络的核心组件，每个神经元与前一层的所有神经元相连接，从而能够学习输入特征之间的复杂关系。

激活函数： 在每个全连接层之间，我们使用 ReLU（Rectified Linear Unit）作为激活函数。ReLU 引入了非线性性质，有助于模型学习非线性函数，并避免梯度消失问题。

输出层： 最后一个全连接层使用 softmax 激活函数，用于多类别分类任务。在这个任务中，输出层有 7 个神经元，每个神经元对应一个 mRS 评分类别（0 到 6）。softmax 函数将模型的原始输出转化为各个类别的概率分布。

权重初始化

我们对全连接层的权重进行 Xavier 初始化。Xavier 初始化有助于避免梯度消失问题，破坏权重的对称性，以及提高模型的训练速度和泛化能力。

损失函数和优化器

模型的损失函数选择了交叉熵损失函数（CrossEntropyLoss），这是多类别分类任务中常用的损失函数。我们选择了 Adam 优化器，它是一种自适应学习率算法，能够自动调整学习率以最小化损失函数。

训练过程

模型通过迭代训练数据集来学习特征之间的关系和类别之间的决策边界。在每个迭代周期（epoch）中，计算损失并使用反向传播算法来更新权重，以不断优化模型，提高分类性能。

正则化

为了提高模型的泛化能力和鲁棒性，我们引入了随机噪声进行训练。随机噪声可以提供正则化效应，帮助避免过拟合，并增加模型的探索性，以更好地泛化到未见过的数据。

模型评估分析

基于首次影像结果构建的模型的损失情况如图 19 所示：

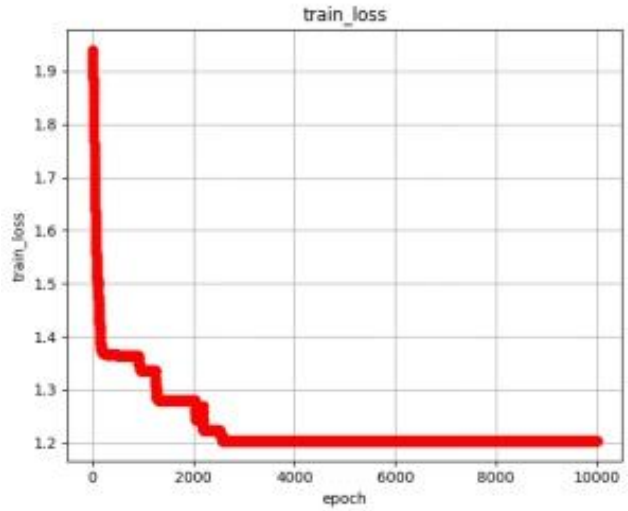

图 19 基于首次影像结果构建的模型的损失情况

基于首次+随访影像结果构建的模型的损失情况如图所示：

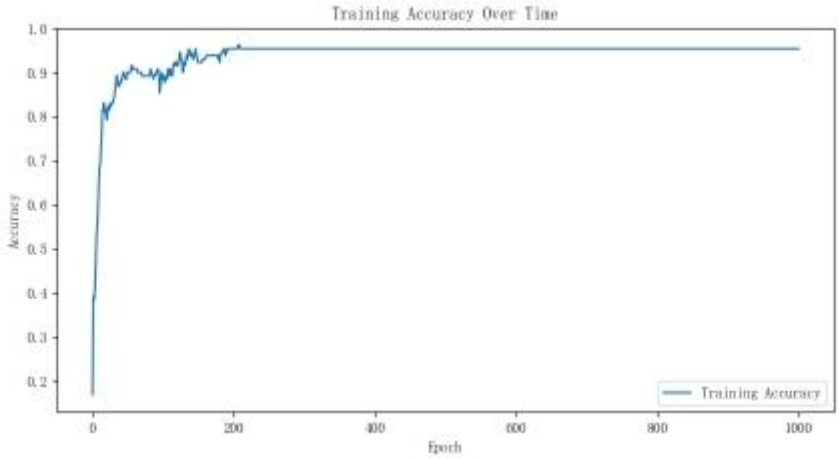

图 20 基于首次+随访影像结果构建的模型的损失情况

基于首次影像结果构建的模型的准确率如图 21 所示：

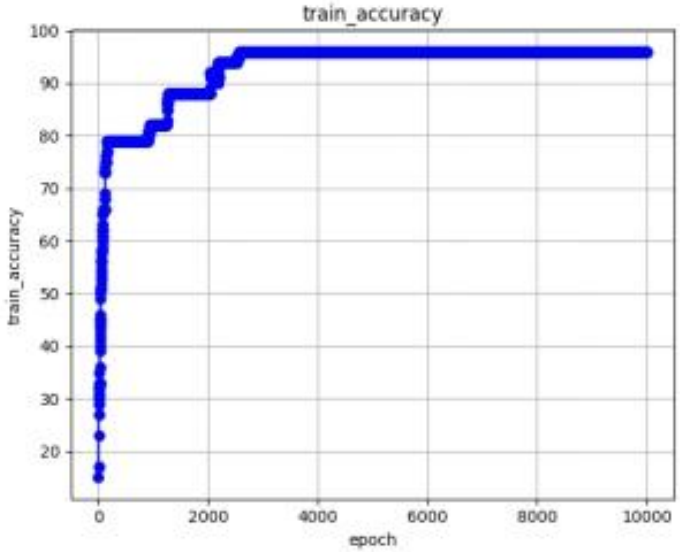

图 21 基于首次影像结果构建的模型的准确率

我们建立的 MLP 模型准确率稳定在 96%。

基于首次+随访影像结果构建的模型的准确率如图 22 所示：

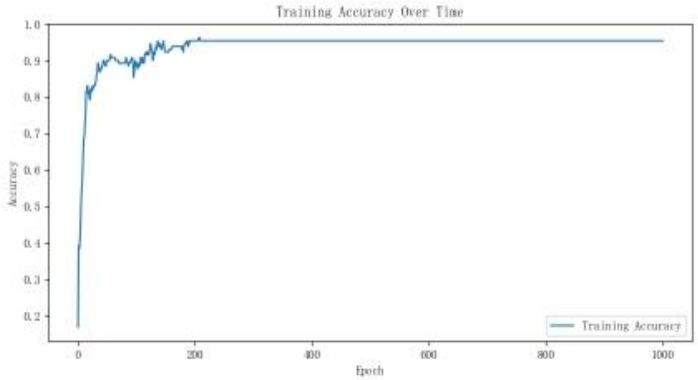

图 22 基于首次+随访影像结果构建的模型的准确率

我们建立的 MLP 模型准确率稳定在 95%

保存准确率最高模型训练参数后，使用该参数对目标值 mRS 进行预测。

4.3.2 预后关联分析

个人史和疾病史：

年龄：较高年龄可能与较高的 90 天 mRS 评分相关，因为老年患者更容易受到脑卒中的影响。

性别：性别可能不直接影响 90 天 mRS 评分，但性别差异可能在其他因素中起作用。

高血压、卒中、糖尿病、房颤、冠心病、吸烟和饮酒史：这些疾病史可能会增加患者的卒中风险，从而对 90 天 mRS 评分产生不利影响。

治疗方法：

脑室引流、止血治疗、降颅压治疗、降压治疗、镇静与镇痛治疗、止吐护胃和营养神经：不同治疗方法可能对 90 天 mRS 评分产生不同影响。例如，及早的脑室引流可能有助于减轻颅内压，提高预后。

影像特征：

血肿和水肿体积：较大的血肿和水肿体积可能与较高的 90 天 mRS 评分相关，因为它们可能导致更严重的脑组织损伤。

血肿和水肿位置：血肿和水肿的位置可能会影响不同脑区的功能，从而影响 90 天 mRS 评分。

根据前 100 位患者（sub001 至 sub100）的 90 天 mRS 评分个人史、疾病史、发病相关信息、首次和后续随访期间的影像特征，构建一个预测模型用于预测出血性脑卒中患者的治疗方案。

根据所建立的模型我们建议：

1、医生在为患者制订治疗方案时要充分地考虑到患者的病史，针对高风险因素，例如高血压、糖尿病等，应考虑更积极的干预措施，以减少出血性脑卒中的风险。

2、为不同患者指定个性化治疗方案。在治疗方案中可以根据患者的病史、治疗方法和预测的血肿扩张风险进行调整。

3、及时调整治疗方案。随着时间的推移，患者的病情会发生变化，最初制定的治疗方案可能会不再适用，因此最好每隔一段时间根据患者的病情及时调整治疗方案。

4、注意患者可能会出现的并发症，如颅内压增高，在治疗时若发现异常情况要及时处理。

五、 模型评价与推广

5.1 模型的优点

（1）模型充分结合实际，简化脑卒中预测条件，考虑了诸多重要因素得到合理的模型，如患者的年龄、性别、疾病史等。这样得到的模型贴合实际，具有较高的应用价值，可以推广到不同患者群体。

（2）模型运用深度学习和神经网络思想，抓住影响脑卒中预测问题的重要因素，将复杂的医学数据转化为简单的预测问题，合理设置参数，模型的输出结果符合医学需求，能解决实际的预测问题。

（3）本文使用的多层感知器（MLP）算法具有非线性建模能力、自适应学习率等优点，对于求解脑卒中预测模型非常适用。

（4）本文得到的治疗方案具有高效率、稳定性和治疗效果均衡等特点，基本不存在明显的不合理问题，在现有条件下能有效提高脑卒中患者的治疗效果和生活质量。

5.2 模型的不足

（1）实际应用中，患者的遗传因素和生活方式可能也是重要的因素，但本文未能考虑到这些因素的影响，一定程度上影响了模型的准确性

（2）本文提出的模型对于现有条件使用效果较好，由于时间问题没有对其他情况进行检验。对于其他特定患者群体（如老年患者、家族遗传性脑卒中患者等），可能无法达到较好的效果。

（3）实际上，患者的年龄、疾病史等因素的影响可能不一定是线性的，而本文将其作为线性因子处理，忽略了边际效应的影响。

5.3 模型的推广

在模型推广方面，可以将模型参数进行调整，例如，结合遗传信息等因素，从而解决遗传性脑卒中患者的预测问题。

改进方面，可以结合参考文献中的新研究，进一步考虑患者的生活方式、遗传信息等对脑卒中预测的影响，从而得到更全面、更合理的模型。

这些改进将有助于模型在更广泛范围内的应用，提高脑卒中患者的预测准确性，为临床决策提供更多有力的支持。

参考文献

[1] 朱遂强,刘鸣,等.中国脑出血诊治指南(2019)[J].中华神经科杂志, 2019,52(12):12.DOI:10.3760/cma.j.issn.1006-7876.2019.12.003.
[2] 张丽娜,李国春,周学平等.基于支持向量机的急性出血性脑卒中早期预后模型的建立与评价[J].南京医科大学学报(自然科学版),2016,36(01):80-84.
[3] 郭昕. 出血性脑卒中后血压及其控制对预后的影响研究[D].吉林大学,2018.
[4] Hamerly G, Elkan C. Learning the k in k-means[J]. Advances in neural information processing systems, 2003, 16.
[5] Hamerly G, Elkan C. Learning the k in k-means[J]. Advances in neural information processing systems, 2003, 16.
[6] Bond S R, Hoeffler A, Temple J R W. GMM estimation of empirical growth models[J]. Available at SSRN 290522, 2001.
[7] Blundell R, Bond S, Windmeijer F. Estimation in dynamic panel data models: improving on the performance of the standard GMM estimator[J]. Nonstationary panels, panel cointegration, and dynamic panels, 2001: 53-91.
[8] Al Shalabi L, Shaaban Z, Kasasbeh B. Data mining: A preprocessing engine[J]. Journal of Computer Science, 2006, 2(9): 735-739.
[9] Bro R, Smilde A K. Principal component analysis[J]. Analytical methods, 2014, 6(9): 2812-2831.
[10] Roweis S. EM algorithms for PCA and SPCA[J]. Advances in neural information processing systems, 1997, 10.
[11] Van der Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(11).
[12] Gardner M W, Dorling S R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences[J]. Atmospheric environment, 1998, 32(14-15): 2627-2636.
[13] Noriega L. Multilayer perceptron tutorial[J]. School of Computing. Staffordshire University, 2005, 4(5): 444.
[14] Murtagh F. Multilayer perceptrons for classification and regression[J]. Neurocomputing, 1991, 2(5-6): 183-197. MLP
[15] Zhang Z, Sabuncu M. Generalized cross entropy loss for training deep neural networks with noisy labels[J]. Advances in neural information processing systems, 2018, 31. .

附录

附录 1

问题一 a 求解代码

```
    def calculate_relative_time(data_dict):
        """
        计算相对时间的字典，包含两个键 time1 和 time2，值是列表，分别计算为：
        time1 = '发病到首次影像检查时间间隔' + '随访 1 时间点' - '入院首次检查时间点'
        time2 = '发病到首次影像检查时间间隔' + '随访 2 时间点' - '入院首次检查时间点'
        Args:
        data_dict (dict): 包含数据的字典
        Returns:
        dict: 包含两个键 time1 和 time2 的相对时间字典
        """
        time1_values = [data_dict['发病到首次影像检查时间间隔'][i] + ((
                data_dict['随访 1 时间点'][i] - data_dict['入院首次检查时间点'][i]).total_seconds() / 3600) for i in
                    range(len(data_dict['发病到首次影像检查时间间隔']))]
        time2_values = [data_dict['发病到首次影像检查时间间隔'][i] + ((
                data_dict['随访 2 时间点'][i] -
                data_dict['入院首次检查时间点'][
                    i]).total_seconds() / 3600) for i in
                    range(len(data_dict['发病到首次影像检查时间间隔']))]

        relative_time_dict = {
            'time1': time1_values,
            'time2': time2_values
        }

    return relative_time_dict

    def is_volume_increase(previous_volume, current_volume):
        """
        判断体积是否增大的函数。

        Args:
        previous_volume (float): 上一次的体积
        current_volume (float): 当前的体积

        Returns:
        bool: 如果下一次的体积比上一次增加 6 或以上，或者相对体积增加≥33%，返回 True；否则返回 False。
```

```
        """
        volume_increase = current_volume - previous_volume

        # 计算相对体积增加百分比
        relative_increase = (volume_increase / previous_volume) * 100 if previous_volume != 0 else 0

        # 检查条件
        if volume_increase >= 6 or relative_increase >= 33:
            return True
        else:
            return False

    #判断某个时间点是否在 48 小时内且发生血肿
    def is_volume_increase_within_time(time,is_volume_increase):
        if time<=48 and is_volume_increase==True:
            return True
        else:
            return False
```

附录 2

问题一 b 求解代码

```
    class MLP(nn.Module):
        def __init__(self, input_size, hidden_size, output_size):
            super(MLP, self).__init__()
            self.fc1 = nn.Linear(input_size, hidden_size)
            self.relu = nn.ReLU()
            self.fc2 = nn.Linear(hidden_size, hidden_size)
            self.softmax = nn.Softmax(dim=1)
            self.fc3 = nn.Linear(hidden_size, output_size)
            self.softmax = nn.Softmax(dim=1)
            init.xavier_normal_(self.fc1.weight)
            init.xavier_normal_(self.fc2.weight)

        def forward(self, x):
            x = self.fc1(x)
            x = self.relu(x)
            x = self.fc2(x)
            x = self.relu(x)
            x = self.fc3(x)
            x = self.softmax(x)
            return x
    def normalize(data, index):
        column_to_normalize = index
```

```
        # 计算列的最小值和最大值
        min_val = data[:, column_to_normalize].min()
        max_val = data[:, column_to_normalize].max()

        # 执行 Min-Max 归一化
        normalized_column = (data[:, column_to_normalize] - min_val) / (max_val - min_val)

        # 将归一化后的列替换回原数组
        data[:, column_to_normalize] = normalized_column

        return data
```

附录 3

问题二 a 求解代码

```
    def show_residual(pre_y, y):
        residuals = y - pre_y
        sse = np.sum(residuals ** 2)
        mean_actual = np.mean(y)
        tss = np.sum((y - mean_actual) ** 2)
        r_square = 1 - (sse / tss)
        mean_residual = np.mean(residuals)
        std_residual = np.std(residuals)
        plt.scatter(y, residuals)
        plt.xlabel("Actual Values")
        plt.ylabel("Residuals")
        plt.title("Residual Plot")
        plt.axhline(y=0, color='r', linestyle='--')
        plt.show()
        print(f"平均残差: {mean_residual}")
        print(f"标准差: {std_residual}")
        print("SSE:", sse)
    print("R-square:", r_square)
    def merge_data(CheckNumInfos):
        CheckNumInfos = sorted(CheckNumInfos, key=lambda x: x.relative_time)
        tmps = []
        ks = CheckNumInfos[0].k
        cnt = 1
        for i in range(1, len(CheckNumInfos)):
            if CheckNumInfos[i].relative_time == CheckNumInfos[i - 1].relative_time:
                cnt += 1
                ks += CheckNumInfos[i].k
            else:
                tmps.append(
                    CheckNumInfo(-1, -1, CheckNumInfos[i - 1].check_time,
```

```
CheckNumInfos[i - 1].first_time, -1, -1, -1,
                                k=ks / cnt))
                cnt = 1
                ks = CheckNumInfos[i].k
        if cnt >= 1:
            tmps.append(
                CheckNumInfo(-1, -1, CheckNumInfos[-1].check_time, CheckNumInfos[-
1].first_time, -1, -1, -1, k=ks / cnt))
    return tmps
```

附录 4

问题二 b 求解代码

```
    def pca(x, labels, method=-1, n_clusters=-1):
        _pca = PCA(n_components=2)
        reduced_data = _pca.fit_transform(x)
        plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=labels, cmap='viridis')
        sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:, 1], hue=labels,
palette="viridis", legend="full")
        plt.title('PCA method:{}  clusters: {}'.format(method, n_clusters))
        plt.xlabel('特征 1')
        plt.ylabel('特征 2')
    plt.show()

    def tsne(X_high_dim, y_labels, method, n_clusters):
        tsne = TSNE(n_components=2, random_state=42)
        X_low_dim = tsne.fit_transform(X_high_dim)
        plt.figure(figsize=(8, 6))
        sns.scatterplot(x=X_low_dim[:, 0], y=X_low_dim[:, 1], hue=y_labels,
palette="viridis", legend="full")
        plt.title("t-SNE Clustering {}".format(n_clusters))
        plt.xlabel("t-SNE Dimension 1")
        plt.ylabel("t-SNE Dimension 2")
        plt.title('TSNE method:{}  clusters: {}'.format(method, n_clusters))
        plt.legend(title="Class")
    plt.show()
    def k_means():
        X = get_data()
        random_state = 42
        n_clusters = 5
        kmeans = KMeans(n_clusters=n_clusters, random_state=random_state)
        kmeans.fit(X)

        # 获取聚类结果和聚类中心
        labels = kmeans.labels_
    pca(X, labels, "K-means", n_clusters=n_clusters)
    def clean_data(x, y):
```

```
        data = list(zip(x, y))
        data = sorted(data,  key=lambda x: x[0])
        z_scores_x = np.abs(stats.zscore(x))
        threshold = 3
        outliers_x = np.where(z_scores_x > threshold)
        data_cleaned = [data[i] for i in range(len(data)) if i not in outliers_x[0]]
        x_cleaned, y_cleaned = zip(*data_cleaned)
    return x_cleaned, y_cleaned
    def cal_residual_by_label(ed_volume_by_lable):
        re = [] * 100
        for label, labels in enumerate(ed_volume_by_lable):
            dic = {}
            for person in labels:
                if person.id not in dic:
                    id = int(person.id.split('sub')[1])
                    dic[id] = [[], []]
                    dic[id][1].append(person.ed_volume)
                    dic[id][0].append(person.relative_time)
                else:
                    dic[id][1].append(person.ed_volume)
                    dic[id][0].append(person.relative_time)
            for k, v in dic.items():
                x = np.sum(np.array(v[0])) / len(v[0])
                y = np.sum(np.array(v[1])) / len(v[1])
                if label == 0:
                    cal_re = a2.cal_gaussian(x, 1.798e+05, 360.1, 44.49)
                elif label == 1:
                    cal_re = a2.cal_exp(x, 1.587e+04, 0.001485)
                elif label == 2:
                    cal_re = a2.cal_gaussian(x, 4.367e+04, 362.5, 464.3)
                elif label == 3:
                    cal_re = a2.cal_gaussian(x,  4.508e+04, 363.4, 436.1)
                re.append([k, x, y, cal_re, y - cal_re])
    def gmm():
        from sklearn.mixture import GaussianMixture
        X = get_data()
        n_components = 4
        gmm = GaussianMixture(n_components=n_components)
        # 拟合模型
        gmm.fit(X)
        labels = gmm.predict(X)
        pca(X, labels, "gmm", n_clusters=n_components)
        tsne(X, labels, "gmm", n_clusters=n_components)
probs = gmm.predict_proba(X)
```

附录 5

问题二 c 求解代码

```
def get_method_code():
    t1 = get_table1()
    v = t1.loc[0:99, '脑室引流':'营养神经'].values.tolist()
    codes = []
    for s in v:
        s = [str(i) for i in s]
        s = ''.join(s)
        codes.append(s)
return codes
```

附录 6

问题二 d 求解代码

```
def get_data():
    at1 = a2.get_appendix_table1()
    t2 = a2.get_table2()
    t1 = a2.get_table1()
    codes = get_method_code()
    Infos = []
    for i in range(100):  # 前 100 个患者
        id = t2.loc[i]['ID']
        # id 对应的下标
        index = int(id.split('sub')[1]) - 1
        repeat_num = at1.loc[index]['重复次数']
        first_time = at1.loc[index]['入院首次检查时间点'] - pd.Timedelta(hours=t1.loc[index]['发病到首次影像检查时间间隔'])
        first_num = at1.loc[index]['入院首次检查流水号']
        original_ed = t2.loc[i][13]
        hm_volume = t2.loc[i][2]
        method = codes[index]
        Infos.append(Info(method, at1.loc[index]['入院首次检查时间点'], first_time, original_ed, hm_volume, original_ed, original_ed, -1))

        for j in range(1, min(repeat_num, 8)):
            key = '随访' + str(j) + '流水号'
            check_num = t2.loc[i][key]
            check_time = at1.loc[index][at1.columns.get_loc(key) - 1]
            ed_volume = t2.loc[index][t2.columns.get_loc(key) + 12]
            if ed_volume == 0:
                break
            last_time_ed = Infos[-1].ed_volume
            Infos.append(Info(method, check_time, first_time, ed_volume, hm_volume, last_time_ed, original_ed, -1))
```

```
        x = []
        y = []
        for info in Infos:
            x.append(info.hm_volume)
            y.append(info.ed_volume)
        x, y = b2.clean_data(x, y)
        plt.scatter(x, y, c='r', cmap='viridis')
        plt.title('血肿与水肿体积关系图')
        plt.xlabel('血肿体积')
        plt.ylabel('水肿体积')
        plt.plot(x, 0.1642*np.array(x) + 2.13e+04, c='g')
        plt.legend(["y = 0.1642*x + 2.13e+04"], loc="upper right")
    plt.show()
```

附录 7

问题三 a 求解代码

```
    def data_process(data_type):
        f1_path = '表 1-患者列表及临床信息.xlsx'
        f2_path = '表 2-患者影像信息血肿及水肿的体积及位置.xlsx'
        f3_path = '表 3-患者影像信息血肿及水肿的形状及灰度分布.xlsx'
        n_rows_to_read = 100  # 替换为你希望读取的行数

        columns_to_read1 = ['入院首次影像检查流水号', '年龄','性别','脑出血前 mRS 评分','高血压病史','卒中病史','糖尿病史','房颤史','冠心病史','吸烟史','饮酒史','发病到首次影像检查时间间隔','血压','脑室引流','止血治疗','降颅压治疗','降压治疗','镇静、镇痛治疗','止吐护胃','营养神经'] # 替换为你希望读取的列名
        columns_to_read2 = ['HM_volume','HM_ACA_R_Ratio',
    'HM_MCA_R_Ratio','HM_PCA_R_Ratio','HM_Pons_Medulla_R_Ratio','HM_Cerebellum_R_Ratio','HM_ACA_L_Ratio','HM_MCA_L_Ratio','HM_PCA_L_Ratio','HM_Pons_Medulla_L_Ratio','HM_Cerebellum_L_Ratio','ED_volume','ED_ACA_R_Ratio','ED_MCA_R_Ratio','ED_PCA_R_Ratio','ED_Pons_Medulla_R_Ratio','ED_Cerebellum_R_Ratio',
    'ED_ACA_L_Ratio','ED_MCA_L_Ratio','ED_PCA_L_Ratio','ED_Pons_Medulla_L_Ratio','ED_Cerebellum_L_Ratio'] # 替换为你希望读取的列名
        columns_to_read3 = ['流水号', 'original_shape_Elongation','original_shape_Flatness',
    'original_shape_LeastAxisLength','original_shape_MajorAxisLength','original_shape_Maximum2DDiameterColumn',
    'original_shape_Maximum2DDiameterRow','original_shape_Maximum2DDiameterSlice','original_shape_Maximum3DDiameter','original_shape_MeshVolume','original_shape_MinorAxisLength','original_shape_Sphericity','original_shape_SurfaceArea',
    'original_shape_SurfaceVolumeRatio','original_shape_VoxelVolume','NCCT_original_firstorder_10Percentile','NCCT_original_firstorder_90Percentile','NCCT_original_firstorder_Energy','NCCT_original_firstorder_Entropy','NCCT_original_firstorder_InterquartileRange','NCCT_original_firstorder_Kurtosis','NCCT_original_firstorder_Maximum',
    'NCCT_original_firstorder_MeanAbsoluteDeviation','NCCT_original_firstorder_Mean','NCCT_original_firstorder_Median',  'NCCT_original_firstorder_Minimum',
    'NCCT_original_firstorder_Range','NCCT_original_firstorder_RobustMeanAbsoluteDevi
```

```
ation','NCCT_original_firstorder_RootMeanSquared',
    'NCCT_original_firstorder_Skewness','NCCT_original_firstorder_Uniformity','NCCT_ori
ginal_firstorder_Variance'] # 替换为你希望读取的列名

        #将每个 Excel 表转换成一个字典
        if data_type == 'train':
            t1_range = [0, 99]
            t2_range = [0, 99]
            t3_range = [0, 576]
        else:
            t1_range = [100, 160]
            t2_range = [0, 100]
            t3_range = [0, 578]
        data_dict1 = read_excel_columns_first_n_rows(f1_path, t1_range,
*columns_to_read1)
        data_dict2 = read_excel_columns_first_n_rows(f2_path, t2_range,
*columns_to_read2)
        data_dict3 = read_excel_columns_first_n_rows(f3_path, t3_range,
*columns_to_read3)
        # #合并字典
        # data_dict = {**data_dict1,**data_dict2,**data_dict11}
        #处理血压键值对
        data_dict1['高压'] = []
        data_dict1['低压'] = []

        for value in data_dict1['血压']:
            high,low = value.split('/')
            data_dict1['高压'].append(int(high))
            data_dict1['低压'].append(int(low))

        del data_dict1['血压']

        data_dict1['性别'] = [1 if gender == '男' else 0 for gender in data_dict1['性别']]


        #筛选出表 3 首次检查的数据
        # 获取匹配 '入院首次影像检查流水号' 的索引位置

        matching_indices = []
        for value in data_dict1['入院首次影像检查流水号']:
            if value in data_dict3['流水号']:
                index = data_dict3['流水号'].index(value)
                matching_indices.append(index)

        # 筛选出符合条件的数据
        filtered_data_dict3 = {}
        for key, values in data_dict3.items():
            filtered_data_dict3[key] = [values[index] for index in matching_indices]
```

```
    data_dict = {**data_dict1, **data_dict2, **filtered_data_dict3}

return data_dict
class MyDataset(Dataset):
    def __init__(self, data, labels):
        self.data = data
        self.labels = labels

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

    def __getitem__(self, idx):
        x = self.data[idx]
        y = self.labels[idx]
        # 添加噪声
        rnd = random.randint(1, 5)
        # if rnd == 1:
        #     x = add_noise(x, noise_level=0.01)

        # # 平移
        # elif rnd == 2:
        #     x = shift(x, shift_amount=5)
        #
        # # 拉伸
        # elif rnd == 3:
        #     x = stretch(x, scale_factor=1.2)

        # 截断
        # elif rnd == 4:
        #     x = truncate(x, length=100)

        return x, y class MyDataset(Dataset):
    def __init__(self, data, labels):
        self.data = data
        self.labels = labels

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

    def __getitem__(self, idx):
        x = self.data[idx]
        y = self.labels[idx]
        # 添加噪声
        rnd = random.randint(1, 5)
        # if rnd == 1:
        #     x = add_noise(x, noise_level=0.01)

        # # 平移
```

```
            # elif rnd == 2:
            #     x = shift(x, shift_amount=5)
            #
            # # 拉伸
            # elif rnd == 3:
            #     x = stretch(x, scale_factor=1.2)

            # 截断
            # elif rnd == 4:
            #     x = truncate(x, length=100)

            return x, y
```

附录 8

问题三 b 求解代码

```
def get_x_data():
    at1 = a2.get_appendix_table1()
    t2 = a2.get_table2()
    t1 = a2.get_table1()
    Infos = [[] for _ in range(130)]
    label_num = 4
    for i in range(100):  # 前 100 个患者
        id = t2.loc[i]['ID']
        # id 对应的下标
        index = int(id.split('sub')[1]) - 1
        repeat_num = min(at1.loc[index]['重复次数'], 5)
        first_num = at1.loc[index]['入院首次检查流水号']
        mRS = t1.loc[index]['90 天 mRS']
        method = t1.loc[index]['脑室引流':'营养神经'].values.tolist()
        base_info = t1.loc[index]['年龄':'饮酒史']
        if base_info['性别'] == '女':
            base_info['性别'] = 0
        else:
            base_info['性别'] = 1
        Infos[i].extend(method)
        Infos[i].extend(base_info)
        v1 = t1.loc[index]['血压']
        v1 = v1.split('/')
        v1 = [int(i) for i in v1]
        Infos[i].extend(v1)
        first_check_info = t2.loc[index][2:23]
        Infos[i].extend(first_check_info)
        img_info = get_t3_info_by_checknum(int(first_num))
        Infos[i].extend(img_info)
        for j in range(1, repeat_num):
            key = '随访' + str(j) + '流水号'
```

```
                check_num = t2.loc[i][key]
                check_info = t2.loc[index][t2.columns.get_loc(key) +
1:t2.columns.get_loc(key) + 22]
                img_info = get_t3_info_by_checknum(int(first_num))
                Infos[i].extend(check_info)
                Infos[i].extend(img_info)

        for i in range(131, 160):  # 前 100 个患者
            id = t2.loc[i]['ID']
            # id 对应的下标
            index = int(id.split('sub')[1]) - 1
            repeat_num = min(at1.loc[index]['重复次数'], 5)
            first_num = at1.loc[index]['入院首次检查流水号']
            mRS = t1.loc[index]['90 天 mRS']
            method = t1.loc[index]['脑室引流':'营养神经'].values.tolist()
            base_info = t1.loc[index]['年龄':'饮酒史']
            if base_info['性别'] == '女':
                base_info['性别'] = 0
            else:
                base_info['性别'] = 1
            # Infos[i].append(mRS)
            Infos[i - 30].extend(method)
            Infos[i - 30].extend(base_info)
            v1 = t1.loc[index]['血压']
            v1 = v1.split('/')
            v1 = [int(i) for i in v1]
            Infos[i - 30].extend(v1)
            first_check_info = t2.loc[index][2:23]
            Infos[i - 30].extend(first_check_info)
            img_info = get_t3_info_by_checknum(int(first_num))
            Infos[i - 30].extend(img_info)
            for j in range(1, repeat_num):
                key = '随访' + str(j) + '流水号'
                check_num = t2.loc[i][key]
                check_info = t2.loc[index][t2.columns.get_loc(key) +
1:t2.columns.get_loc(key) + 22]
                img_info = get_t3_info_by_checknum(int(first_num))
                Infos[i - 30].extend(check_info)
                Infos[i - 30].extend(img_info)
        max_size = 279
        for info in Infos:
            while len(info) < max_size:
                info.append(0)
        x = np.array(Infos, dtype=float)
        mean = np.mean(x, axis=0)
        std = np.std(x, axis=0)
        normalized_data = (x - mean) / std
    return normalized_data
```

附录 9

问题三 c 求解代码

```
    def train():
        X = get_x_data()
        Y = get_y_data()
        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
        X_train, X_test, Y_train, Y_test = torch.FloatTensor(X_train),
torch.FloatTensor(X_test), torch.FloatTensor(
            Y_train), torch.FloatTensor(Y_test)
        X_test, Y_test = torch.FloatTensor(X), torch.FloatTensor(Y)
        # 定义模型超参数
        input_size = X_train.shape[1]
        hidden_size = 256
        output_size = Y_train.shape[1]
        learning_rate = 0.001
        num_epochs = 100
        # 初始化模型、损失函数和优化器
        model = MLP(input_size, hidden_size, output_size)
        # criterion = nn.BCELoss()  # 使用二元交叉熵损失
        criterion = torch.nn.CrossEntropyLoss()
        optimizer = optim.Adam(model.parameters(), lr=learning_rate)
        # 创建数据加载器
        train_dataset = TensorDataset(X_train, Y_train)
        train_loader = DataLoader(train_dataset, batch_size=80, shuffle=True)
        # 训练模型
        losses = []  # 用于保存每个 epoch 的损失值
        accuracies = []  # 用于保存每个 epoch 的准确度
        for epoch in range(num_epochs):
            epoch_loss = 0.0  # 用于保存当前 epoch 的总损失
            correct_predictions = 0  # 用于保存当前 epoch 的正确预测数量
            total = 0
            for inputs, labels in train_loader:
                optimizer.zero_grad()
                outputs = model(inputs)
                loss = criterion(outputs, labels)
                loss.backward()
                optimizer.step()
                epoch_loss += loss.item() * inputs.size(0)  # 累积当前 batch 的损失
                # 计算当前 batch 的准确度
                predictions = (outputs >= 0.5).float()
                correct_predictions += torch.sum(predictions == labels).item()
                total += outputs.size(0) * outputs.size(1)
            # 计算平均损失和准确度
            epoch_loss /= len(train_loader.dataset)
```

```
            accuracy = correct_predictions / total
            losses.append(epoch_loss)
            accuracies.append(accuracy)
            if (epoch + 1) % 10 == 0:
                print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {epoch_loss:.4f},
Accuracy: {accuracy * 100:.2f}%')
        # 在测试集上评估模型
        model.eval()
        with torch.no_grad():
            test_outputs = model(X_test)
            test_outputs[test_outputs >= 0.5] = 1
            test_outputs[test_outputs < 0.5] = 0
            for i in test_outputs:
                print("{}".format(i.tolist()))
            accuracy = (test_outputs == Y_test).float().mean()
            print(f'Test Accuracy: {accuracy.item() * 100:.2f}%')
        # 绘制损失趋势图
        plt.figure(figsize=(10, 5))
        plt.plot(range(1, num_epochs + 1), losses, label='Training Loss')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.title('Training Loss Over Time')
        plt.legend()
        plt.show()
        # 绘制准确度趋势图
        plt.figure(figsize=(10, 5))
        plt.plot(range(1, num_epochs + 1), accuracies, label='Training Accuracy')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.title('Training Accuracy Over Time')
        plt.legend()
        plt.show()
```