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	上海交通大学	
参赛队号	23102480018	
队员姓名	1.	左润衡
	2.	朱家杰
	3.	于飞

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：基于机器学习的智能诊疗

摘　要：

出血性脑卒中发病罕见但死亡率高，目前缺少该病演变的科学性医学解释机制，患者预后悲观。针对这一问题，本文采用了机器学习方法，分析了少量发病样本的放射影像数据，患者就诊数据和临床医疗方案等，探讨影响出血性脑卒中发生的关键因素，构建了针对出血性脑卒中等罕见疾病的智能诊疗模型，为临床决策提供科学依据，提高患者的预后水平。

针对问题一，本文探究了血肿扩张风险相关因素。首先结合患者的个人、发病和影像检测等相关信息，根据所给约束条件判断患者发病后 48 小时内是否发生血肿扩张事件，并准确记录血肿扩张的发生时间；其次，以是否发生血肿扩张事件为目标变量，本文结合患者的个人信息、疾病史、治疗方案和预后等数据，建立基于多层感知机的血肿扩张风险预测模型，以此构建患者相关信息与血肿扩张之间的关联关系。该模型的量化预测结果准确率在 75% 以上。

针对问题二，本文探究了水肿体积进展模式及其与治疗手段的关联关系。为研究水肿体积进展规律，本文采用了高斯函数拟合曲线法，结合患者水肿体积及重复检查时间点数据，构建了水肿体积随时间的进展曲线，平均残差为 73.69；为更加精确地探究水肿体积进展模式，考虑患者个体层面差异，采用 Kmeans 聚类为不同亚组，并构建独立的进展曲线，总平均残差为 68.64，相比之下残差降低了 6.85%，说明将患者分为不同亚组能够更好地反应进展规律。为研究治疗方法对水肿体积进展的影响，本文采用 F 检验方法进行显著性分析，其中降颅压治疗最为显著。最后，采用 Spearsman 系数揭示了血肿体积、水肿体积与治疗方法之间的内在联系，为更好地协助提供个性化治疗方案给予支持。

针对问题三，本文探究了出血性脑卒中患者的预后预测及关键影响因素。根据患者的个人信息、疾病史、发病相关特征及首次影像检查结果等数据，采用多层感知机算法，构建了患者 90 天 mRS 评分的初步预测模型，在测试集准确率 15% 以上。在此基础上结合随访影像检查结果，采用循环神经网络算法构建预测模型，相比初步预测模型的准确率提升了 10% 以上，为 26%。本文进一步分析了预后与个人史、疾病史、治疗方法及影像特征等多个因素的关联关系，明确影响患者预后的关键因素，为临床实践提供了有效的参考意见。

本文基于少样本多特征的临床数据，采用机器学习方法，系统地分析了影响出血性脑卒中的关键因素，同时构建了能够实现对患者血肿扩张风险、血肿周围水肿的发生及演进规律的预测，评估患者的预后情况的一系列智能诊疗预测模型。具有重要的临床应用价值和学习参考价值。

关键词：少样本机器学习　高斯函数拟合　Spearsman 相关系数　智能诊疗模型

目录

1 问题重述

1.1 引言

出血性脑卒中是脑实质内血管破裂引起的一种脑出血疾病，约占脑卒中发病率的 10-15%。常见的病因包括脑动脉瘤破裂和脑动脉异常等，病理复杂。该疾病表现为急性起病、进展快速[1]，严重时急性期内病死率高达 45-50%，并导致约八成左右的患者遗留重度神经功能障碍。因此，准确评估患者的发病风险、预测预后，并优化临床治疗决策具有极大的临床意义。为此，需要整合影像学特征、患者临床信息和治疗策略。

在出血性脑卒中患者中，血肿范围的扩大和血肿周围水肿的发生是导致预后不良的重要因素之一。血肿范围的扩大可能导致颅内压急剧增高，加剧神经功能损伤。此外，血肿周围的水肿可能对脑组织产生额外的压力，进一步损害神经元功能。因此，早期识别和监测这两个关键事件对于改善患者的预后和生活质量至关重要。

医学影像技术和人工智能技术的快速发展为出血性脑卒中患者提供了新的机会[2]。这些技术可以用于监测脑组织损伤和发展，通过分析海量影像数据和医学就诊单、体检信息等。构建出智能诊疗模型。这有助于发现并明确导致出血性脑卒中预后不良的危险因素，实现疗效评估和预测的个性化。未来，这些研究成果将应用于临床实践，改善出血性脑卒中患者的预后。

1.2 问题的提出

以构建出血性脑卒中的临床智能诊断模型为中心，本文依次提出如下问题：

问题一：

血肿扩张是出血性脑卒中临床智能诊断的首要研究问题，血肿扩张趋势与病人的个人特征、发病及治疗相关特征存在密切联系，建立血肿扩张风险相关因素模型的核心问题分为以下两点：

1）诊断：根据病人发病到首次影像检查时间间隔和各时间点流水号及对应的血肿体积，判断患者在 48 小时内是否发生血肿扩张；

2）预测：根据第一点得出的结果，结合前 100 例患者的个人史，疾病史，发病及治疗相关特征、影像检查结果等，预测所有患者发生血肿扩张的概率。

问题二：

水肿通常伴随着血肿而发生，为了建立更加精准的治疗诊断模型，研究水肿的进展模式以及治疗干预手段对水肿进展的影响是关键所在，建立水肿进展相关因素模型的核心问题分为以下两点：

1）发展性：首先，根据前 100 名患者的水肿体积和相应的检查时间点数据，构建一条前提患者水肿体积随时间进展曲线；其次，不同人群的水肿体积进展模式可能有所差异，根据患者的不同个体特征分为 3 至 5 个亚组，构建不同人群的水肿体积进展曲线。

2）关联性：首先，分析不同治疗方法对水肿体积进展模式的影响；其次，水肿体积的进展模式也与血肿体积相关，可分析水肿体积、血肿体积及治疗方法三者之间的关联关系。

问题三：

以上两点问题根据患者个人、疾病及治疗特征，研究构建血肿、水肿以及治疗方法等相关因素模型，在此基础上，智能诊断模型也应针对患者预后效果及关键因素进行探索研究，该问题的核心主要分为以下两点：

1）预后预测：首先，根据前 100 名患者的个人、疾病及治疗、首次影像结果特征，构建全体患者的 90 天 mRS 评分预测模型；其次，为了使 mRS 评分预测模型更加精准，在以上数据不变的条件下，加入随访影像结果特征，构建全体患者的 90 天 mRS 评分预测模型。

2）关键因素：首先，分析患者的预后（90 天 mRS）和个人史、疾病史、治疗方法及影像特征（包括血肿/水肿体积、血肿/水肿位置、信号强度特征、形状特征）等关联关系；其次，根据各个因素相互间的关联关系，为临床相关决策提出建议。

1.3 问题的分析

问题一：

本问的核心问题为判断患者 48 小时内是否发生血肿扩张以及预测所有患者发生血肿扩张的概率，解题思路如下：

1）根据题目所给各时间点的血肿体积数据，并结合评判标准可以初步判断患者是否发生血肿扩张时间；再根据发病到首次影像检查时间间隔和后续影像检查时间间隔，判断该事件发生时间是否小于 48 小时。如果同时满足上述两种情况，则认为患者在发病后 48 内发生了血肿扩张事件。

2）由题可得，患者的个人史、疾病史、发病及治疗相关特征和影像检查结果等数据与患者是否发生血肿扩张之间存在相关性，故本文将前者患者的相关数据作为输入层，患者是否发生血肿扩张作为输出层（若发生血肿扩张则输出 1，反之输出 0），利用相关智能算法建立输入层和输出层之间的关系，从而构建出预测所有患者发生血肿扩张的概率模型。

问题二：

本问的核心问题为研究血肿周围水肿的进展模型，以及治疗手段和水肿进展之间的关联性，解题思路如下：

1）由题目所给条件，已知水肿体积和相应重复检查时间点数据，可以建立回归模型，通过数据拟合构建出一条全体患者水肿体积随时间进展的曲线；为了使水肿体积随时间进展曲线更加精确，可以通过相关聚类算法，根据患者的个体特征差异，将患者分为 3 至 5 个亚组，之后在每个亚组中重复利用上述回归算法，构建不同人群的水肿体积随时间的进展曲线。

2）已知不同的治疗方法对水肿体积的进展模式会产生不同程度的影响，将治疗方式

数据视为 0-1 变量处理（即采取改治疗方式取 1，反之取 0），则可以对治疗方式和水肿体积进展之间进行显著性检验，进而分析得出每种治疗方式对水肿体积的影响程度大小；同理可得，可以采用上述方法研究治疗方式对血肿体积的影响程度，其次，可以采用 Spearman 相关性分析方法研究水肿体积和血肿体积之间的相关性，进而最终得出水肿体积、血肿体积和治疗方式三者之间的关联关系。

问题三：

本问的核心问题为建立出血性脑卒中患者的预后预测模型，并且进行相关关键因素与预后指标之间的相关性研究，解决思路如下：

1）由题目可知，患者 90 天 mRS 评分与患者的个人史、疾病史、发病及治疗相关和首次影像检查结果之间存在关联，将患者所有已知临床、治疗和首次影像检查数据作为输入层，将 90 天 mRS 评分作为输出层，通过第一问第二点的智能算法构建出血性脑卒中患者的初步预后预测模型；为了使预测模型更加精准，加入随访影像结果数据，采用机器学习算法构建血性脑卒中患者的初步预后预测模型。

2）由题目可知，患者的预后指标（90 天 mRS）与个人史、疾病史、治疗方法及影像结果特征等之间存在相关性，对以上数据采用与问题二中的方法进行相关性分析；其次，重点关注与 90 天 mRS 强相关的指标，基于此为临床相关决策提出合理的建议。

综合三个问题可以得到基于机器学习的智能诊疗模型的整体思路图如下：

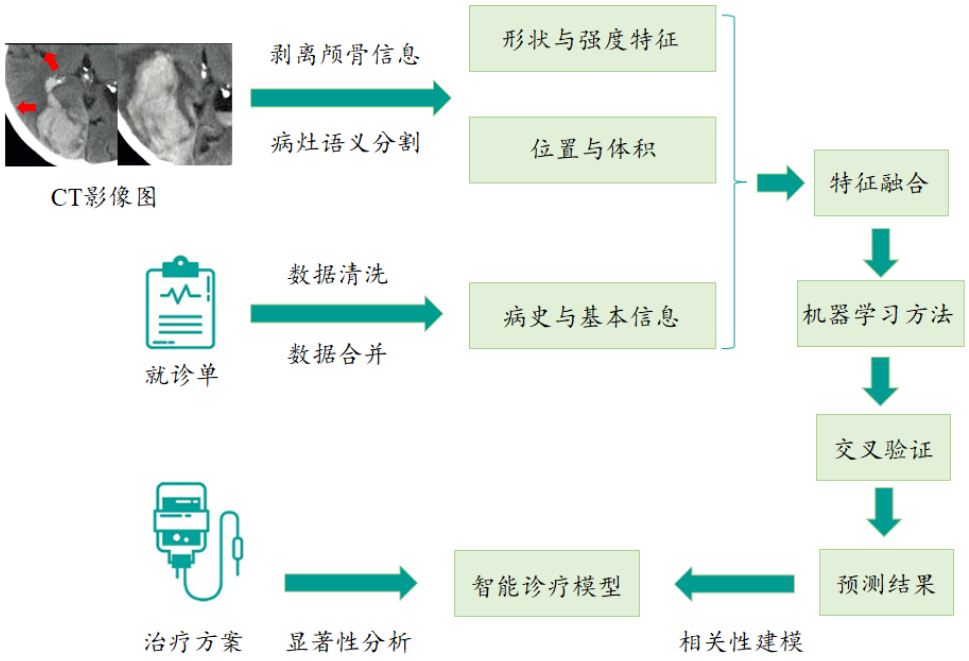

图 1.1　智能诊疗模型的整体思路

2 模型假设与符号说明

2.1 模型假设

1. 血肿和水肿的进展都是以平滑的方式在一定规律下进行的，没有跳变；
2. 获取放射影像的仪器没有差异，不会对病灶的显示产生错误或影响。

2.2 符号说明

符号	意义	量纲
$\mathbf{X}$	ROI 中的灰度强度值集合	
c	混淆偏置量	
T	时间变量	小时
λ	特征向量	
ϵ	残差	
N_p	ROI 像素的数量	
N_v	ROI 体素的数量	
N_f	三角形网格的数量	
N_g	强度一阶直方图中非零项的数量	
p_i	归一化一阶直方图	
V_{ED}	血肿体积	毫升 $/10^3$
V_{HM}	水肿体积	毫升 $/10^3$

3 数据预处理

3.1 数据合并与清洗

数据合并与清洗是数据预处理的重要环节，其主要流程包括以下几个步骤：

1) 数据收集：收集需要合并和清洗的原始数据，可以是来自不同来源、不同格式的数据集。

2) 数据审查与理解：仔细审查原始数据，了解数据集的结构、内容和特征，包括变量名、数据类型、缺失值、异常值等。

3) 数据合并：如果需要合并多个数据集，可以根据共同的关键字段（如 ID、日期等）进行数据合并。常见的合并方式包括连接（join）和拼接（concatenate），根据具体情况选择合适的合并方法。

4) 数据清洗：清洗数据是为了处理缺失值、异常值、重复值和格式问题，以确保数据的准确性和一致性。具体的清洗步骤包括：

处理缺失值

判断缺失值的原因和模式，并选择合适的方法进行处理，如删除包含缺失值的行、填充缺失值、插值等。

处理异常值

识别和处理异常值，可以根据数据分布、业务知识或统计方法来判断异常值，并选择适当的处理策略，如修正、删除或替换。

处理重复值

检测和处理重复的数据行或记录，可以通过唯一标识符（如 ID）来判断重复值，并选择保留一条或删除重复值。

格式转换

将数据转换为正确的格式，如日期、时间、数字等，以便后续分析和建模。

5) 数据转换与衍生：根据需求，进行数据的转换和衍生。例如，计算新的变量、进行数据标准化、归一化或离散化等操作。

6) 数据验证与质量控制：验证清洗后的数据是否符合预期，检查数据的一致性、完整性和准确性，通过可视化、统计分析等方法进行数据质量控制。

7) 数据存储：将经过合并和清洗后的数据保存为新的数据集，以备后续分析和建模使用。

根据上述方法，本文首先附表 1 提供的流水号与患者信息检索表，用首次医学影像的时间对表 1，表 2 及表 3 提供的每位患者的数据进行合并；其次，对于一些随访影像的缺失值，故将去缺失影像检查数据的检查时间点删去；最后，对于重复性的数据，本文选择保留一条确定信息作为数据源进行处理。

3.2 数据编码

题目所给全部数据特征数量为 84 维，其中包含患者个人史、疾病史、患病及治疗等基本信息特征 19 维，以及首次和随访影像检测结果 65 维。

为了消除特征之间的量纲差异、提高模型的收敛速度、改善特征的解释和可解释性以及降低异常值对模型的影响，需要对合并和清洗后的数据进行数据标准化处理。

最大最小值标准化（Min-Max Normalization）是一种常用的数据标准化方法，将数据线性缩放到给定的范围内，通常是 0 到 1 之间。以下是使用最大最小值进行数据标准化的一般步骤：

1) 确定需要标准化的特征：选择需要进行标准化的特征。这些特征可能具有不同的尺度、范围或单位。

2) 计算特征的最小值（min）和最大值（max）：对于每个特征，计算其所有样本的最小值和最大值。

3) 应用标准化公式：使用以下标准化公式对每个特征进行标准化处理，其中标准化值：

$$std = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

这个公式计算了每个样本的标准化值，其中原始值减去最小值，再除以最大值与最小值的差。这个过程将使得特征的值缩放到 0 到 1 之间。mRS 的处理为将 0 至 6 的指标缩放在 0 到 1 的区间内，再乘以 3 加 3，进行 round 来作为回归的特征。

4) 应用标准化结果：将标准化后的特征值应用于进一步的数据分析、建模或机器学习算法中。

对于题目所给的分类变量，本文采用独热编码进行处理。独热编码（One-Hot Encoding）是一种常用的数据编码方法，用于将分类变量转换为机器学习算法可以处理的数值型数据。它将每个类别表示为一个新的二进制变量（也称为虚拟变量），其中只有一个变量为 1，其他变量都为 0。这种编码方法可以避免给变量赋予不正确的大小关系，并且在一定程度上解决了分类变量中的顺序问题。独热编码的步骤如下：

1) 确定需要进行独热编码的特征：选择需要进行编码的分类变量。这些变量可能包括性别、地区、颜色等。

2) 创建虚拟变量：对于每个分类变量的每个取值，创建一个新的二进制变量。例如，对于性别变量，可以创建两个新的二进制变量” 男性” 和” 女性”。

3) 将对应的变量置为 1：对于每个样本，根据其原始值，将对应的虚拟变量置为 1，其他虚拟变量置为 0。每个样本只有一个虚拟变量为 1，其他变量都为 0。

本文中对于患者的疾病史、治疗手段等数据都进展独热编码处理，以此保留分类变量的离散性和类别之间的距离，并且使其适用于大多数的机器学习算法，为后续本文采用多种智能算法求解模型做好准备。

4 模型的建立

4.1 血肿扩张风险预测模型

pyradiomics 提供了用来描述脑 CT 影像的基本灰度特征和形状特征。

表 4.1　灰度特征的定义

特征名称	计算表达式	含义
10Percentile	$\mathbf{P}_{10}$	X 的第 10 个百分位数
90Percentile	$\mathbf{P}_{90}$	X 的第 90 个百分位数
Energy	$\sum_{i=1}^{N_p}(\mathbf{X}(i)+c)^2$	图像中体素值大小的度量
Entropy	$-\sum_{i=1}^{N_g} p(i)\log_2(p(i)+\epsilon)$	图像值中的不确定性
InterquartileRange	$\mathbf{P}_{75}-\mathbf{P}_{25}$	四分位距
Kurtosis	$\frac{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^4}{\left(\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^2\right)^2}$	中值分布的峰值的度量
Maximum	$max(\mathbf{X})$	灰度强度最大值
MeanAbsoluteDeviation	$\frac{1}{N_p}\sum_{i=1}^{N_p}\lvert\mathbf{X}(i)-\bar{X}\rvert$	所有强度值与图像数组平均值的平均距离
Mean	$\frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{X}(i)$	灰度强度平均数
Median	$\mathbf{P}_{50}$	灰度强度中位数
Minimum	$min(\mathbf{X})$	灰度强度最小值
Range	$max(\mathbf{X})-min(\mathbf{X})$	灰度值范围
RobustMean-	$\frac{1}{N_{10-90}}\sum_{i=1}^{N_{10-90}}$	稳健范围内图像子集平均值的平均距离
-AbsoluteDeviation	$\left\lvert\mathbf{X}_{10-90}(i)-\bar{X}_{10-90}\right\rvert$	
RootMeanSquared	$\sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)+c)^2}$	含混淆的平方强度值均值的平方根
Skewness	$\frac{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^3}{\left(\sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^2}\right)^3}$	平均值的值分布的不对称性
Uniformity	$\sum_{i=1}^{N_g} p(i)^2$	每个强度值的平方和的度量
Variance	$\frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i)-\bar{X})^2$	每个强度值与平均值的平方距离的平均值

表 4.2　形状特征的定义

特征名称	计算表达式	含义	值域
Elongation	$\sqrt{\frac{\lambda_{minor}}{\lambda_{major}}}$	最大和次大主成分轴的关系	[0,1]
Flatness	$\sqrt{\frac{\lambda_{least}}{\lambda_{major}}}$	最大和最小主成分轴的关系	[0,1]
LeastAxisLength	$4\sqrt{\lambda_{least}}$	封闭椭球体的最小轴长度	[0,+∞]
MajorAxisLength	$4\sqrt{\lambda_{major}}$	封闭椭球体的最大轴长度	[0,+∞]
Maximum2DDiameterColumn	$Euclidean_{colmax}$	行切冠状面	[0,+∞]
Maximum2DDiameterRow	$Euclidean_{rowmax}$	柱切矢状面	[0,+∞]
Maximum2DDiameterSlice	$Euclidean_{slimax}$	行列轴状面	[0,+∞]
MeanAbsoluteDeviation	$\frac{1}{N_p}\sum_{i=1}^{N_p}\lvert\mathbf{X}(i)-\bar{X}\rvert$	强度与图像数组均值的平均距离	[0,+∞]
Maximum3DDiameter	$Euclidean_{max}$	表面网格顶点最大成对欧氏距离	[0,+∞]
MeshVolume	$\sum_{i=1}^{N_f}\frac{Oa_i\cdot(Ob_i\times Oc_i)}{6}$	三角形网格体积	[0,+∞]
MinorAxisLength	$4\sqrt{\lambda_{minor}}$	封闭椭球体的次小轴长度	[0,+∞]
Sphericity	$\frac{\sqrt[3]{36\pi V^2}}{A}$	形状相对于球体的圆度的量度	[0,1]
SurfaceArea	$\sum_{i=1}^{N_f}\frac{1}{2}\lvert\mathrm{a}_i\,\mathrm{b}_i\times\mathrm{a}_i\mathrm{c}_i\rvert$	三角形网格表面积	[0,+∞]
SurfaceVolumeRatio	$\frac{A}{V}$	接近球状的形状	[0,+∞]
VoxelVolume	$\sum_{k=1}^{N_v}V_k$	体素数乘以单个体积	[0,+∞]

由于临床因素对血肿扩增的影响更不确定，且文献记载较少。临床因素对延迟血肿扩大的影响。其机制仍不甚明了。特别是延迟血肿扩大的潜在作用。与持续炎症反应相关的标志物（如白细胞计数、细胞因子）的潜在作用。

丰富的脑图像信息提供了纹理、波形、强度和形状等特征。本文对放射组学的特征进行提取与筛选后交由编码器编码和医学检验报告等特征一并送入基于机器学习的预测模型中。

灰度特征反应了目标区域内体素强度的分布，是对一维数据最直观的一种描述。它包含 17 种相关特征。其具体的含义已在上表呈现。

不规则形状和不均匀密度是典型血肿生长的特征，其轴向方向会在不同时间窗内发生

变化。形状特征能够对目标区域内的三维形状进行描述。一般地，对一个三维形状的物体，可以用若干主方向的向量来描述它的空间分布情况。

4.1.1 约束条件下血肿扩张时间预测

血肿扩张判断标准为：如果后续检查的血肿体积比首次检查增加 ≥6mL 或 ≥33%，则判断为发生了血肿扩张，反之亦然。通过分析题目的约束条件，具体的判断步骤：

根据问题要求，首先提取表 1 中入院首次影像检查流水号、发病到首次影像检查时间间隔的数据，以及表 2 中各时间点流水号及对应的 HM_volume 的血肿体积等特征数据。根据流水号在附表 1 中查找对应首次检查的时间点，以及后续影像检查的时间点。依次计算相邻两次检查血肿体积变化量和变化百分比。

$$\begin{cases} T_{i,j} - T_{0,j} + t_j < 48 \ \ (i = 1, ..., N_{max,j}, j = 1, ..., 100) \\ V_{i,j} - V_{0,j} > 6000 \ \ (i = 1, ..., N_{max,j}, j = 1, ..., 100) \\ \frac{V_{i,j} - V_{0,j}}{V_{0,j}} > 0.33 \ \ (i = 1, ..., N_{max,j}, j = 1, ..., 100) \end{cases} \tag{4.2}$$

其中 i 表示随访影像检查结果的次数，j 表示患者编号，N_{max} 表示患者的最大随访次数。

若满足公式 (4.2) 中的条件记为发生血肿扩张，并记录下血肿扩张发生的时间点。对于血肿扩张发生的时间，本文通过计算发病至首次影像检查时间间隔与后续随意影像检查时间点与首次影像检查时间间隔的和求得。

4.1.2 基于多层感知机算法的血肿扩张预测模型

本问以是否发生血肿扩张事件为目标变量，结合前 100 例患者的个人史，疾病史，发病及治疗相关特征、影像检查结果等，采用了支持向量机 (SVM)、逻辑回归 (LR)、随机森林 (RF)、多层感知机 (MLP) 四种算法，预测所有患者发生血肿扩张的概率。

以下为四种智能算法原理简介：

• 支持向量机

是一种强大的监督学习算法，其核心原理是找到一个超平面，可以将不同类别的数据点分隔开，并最大化数据点到该超平面的距离，从而实现高效的分类。同时支持向量机也可以用作回归问题。具体旨在通过最大化间隔、使用核函数处理非线性数据、优化不敏感损失函数以容忍误差、并通过正则化参数 C 控制模型复杂度，以拟合回归问题中的数据并保持鲁棒性。

$$\gamma = \frac{1}{\|w\|} \tag{4.3}$$

其中 γ 是间隔，ω 是超平面的法向量。

$$f(x) = \langle w, x \rangle + b \tag{4.4}$$

x 是输入特征向量，b 是偏置项。

$$y_i(\langle w, x_i \rangle + b) - 1 \geq \epsilon \tag{4.5}$$

y_i 是第 i 个训练样本的标签，本文中为血肿发生与否。

支持向量的几何间隔如下：

$$\frac{y_i(\langle w, x_i \rangle + b)}{\|w\|} \tag{4.6}$$

本文求解的血肿扩张概率回归问题可以描述为：

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \tag{4.7}$$

其中 C 为正则化参数，ϵ 是松弛变量，x 和 x' 为特征向量，σ 为带宽参数。采用的核函数如下：

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{4.8}$$

• 逻辑回归

逻辑回归通过建立一个逻辑函数 (Sigmod) 来估计观测数据属于某一类别的概率。

等式左侧表示观测数据属于类别 1 的概率，X 表示特征向量，β 表示模型参数。

$$P(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}\cdot\mathbf{x}+b)}} \tag{4.9}$$

逻辑回归存在对线性可分数据的依赖性和处理高维数据时可能出现的过拟合问题。

• 随机森林

随机森林基于决策树构建多个决策树，并通过投票或平均来进行预测或分类。能够降低过拟合风险并提高鲁棒性。

随机选择一个数据子集，样本 *。接着随机选择一部分特征 *。使用选定的数据子集和特征子集训练一棵决策树。

T* = TrainDecisionTree(样本 *, 特征 *)，从而构建多棵决策树:$\{T_1, T_2, \ldots, T_N\}$

$$\hat{y}(\mathbf{x}) = \text{argmax}_C \sum_{i=1}^{N} P_i(C|\mathbf{x}) \tag{4.10}$$

最终预测过程见上述表达式，其中左侧表示输入样本 x 的预测类别，N 代表随机森林中的决策树数量，是第 i 棵决策树预测样本属于类别 C 的概率。对于本文二分类问题亦然行之有效。

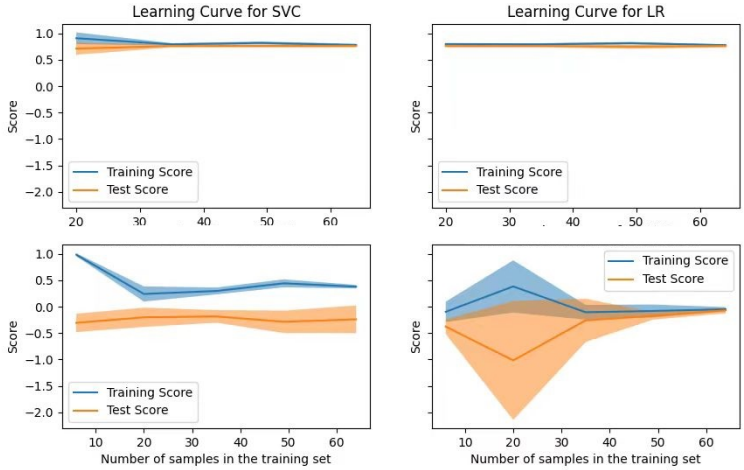

图 4.2　多种算法的预测结果评价其中 sklearn 的神经网络与下文 mlp 不同

• 多层感知机算法

多层感知机由多个神经元组成，分布在一个或多个隐藏层中，每个神经元与前一层的所有神经元相连。输入层、隐藏层和输出层。在每个隐藏层中，输入数据经过权重和偏置的线性组合，然后经过非线性激活函数的变换。这些变换逐层传播，最终得到输出层的预测结果。反向传播算法来更新模型参数，以减小预测值与实际值之间的误差。选择合适的损失函数，来度量模型的性能。

$$z_j^{(2)} = \sum_{i=1}^{N} w_{ij}^{(1)} x_i + b_j^{(1)} \tag{4.11}$$

$$a_j^{(2)} = \sigma(z_j^{(2)}) \tag{4.12}$$

$$z_k^{(3)} = \sum_{j=1}^{M} w_{jk}^{(2)} a_j^{(2)} + b_k^{(2)} \tag{4.13}$$

$$a_k^{(3)} = \sigma(z_k^{(3)}) \tag{4.14}$$

上述式子 (4.11) 到式子 (4.14) 是神经元传递的公式，其中 z 为输出，w 和 b 为权重和偏置量。上标括号内为中间层的编号。

本文所搭建的多层感知机具体参数为下，预测结果见问题一结果，另外绘制了损失函数随训练次数的变化曲线，每个 epoch 的训练准确率和混淆矩阵以及 ROC 曲线 (受试者工作特征曲线)。

真正例（True Positives，TP）：实际为正例的样本被模型正确分类为正例的数量。

真负例（True Negatives，TN）：实际为负例的样本被模型正确分类为负例的数量。

假正例（False Positives，FP）：实际为负例的样本被模型错误分类为正例的数量。

假负例（False Negatives，FN）：实际为正例的样本被模型错误分类为负例的数量。

通过观察 ROC 曲线，越远离斜率 45 度的直线越好，其中横坐标为 False Positive Rate，纵坐标为 True Positive Rate。

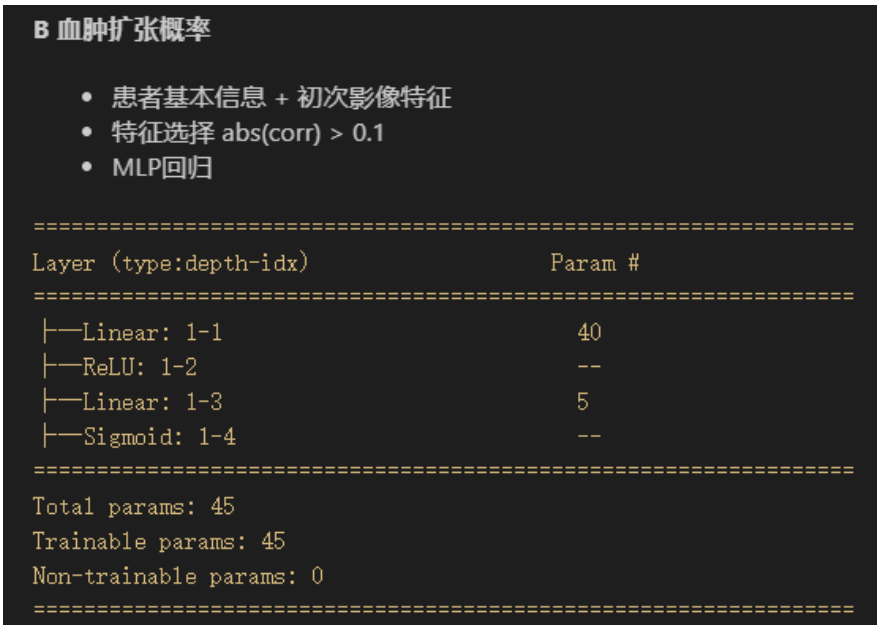

图 4.3　血肿预测 MLP 参数

其中原始特征 19 个，corr 阈值 0.1 筛完之后还剩 9 个，线性层和激活层选择如图。

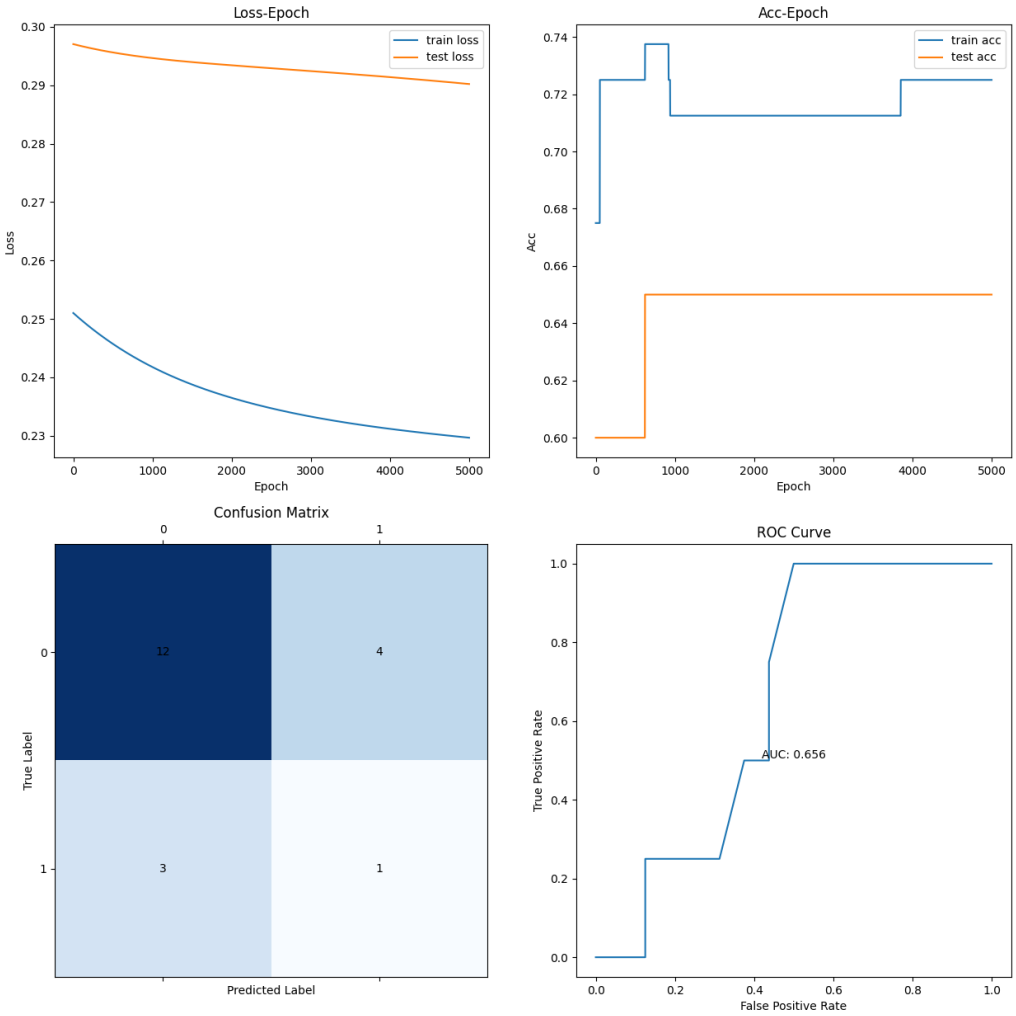

图 4.4　评价 MLP 得到的结果

由上图数据可得，随 epoch 的增加，训练集和测试集的损失函数均有下降趋势，最后分别降到 0.29 和 0.23 左右，可见混淆矩阵的 TP 较多，ROC 曲线反映了本文的 MLP 模型有较高的准确率，为 75% 左右。

4.1.3 问题一结果

表 4.3　问题一 a 结果

患者编号	是否有血肿	血肿扩张的时间	患者编号	是否有血肿	血肿扩张的时间
sub001	0		sub051	0	
sub002	0		sub052	0	
sub003	1	9.5225	sub053	0	
sub004	0		sub054	1	16.225
sub005	1	26.4675	sub055	0	
sub006	0		sub056	0	
sub007	0		sub057	1	14.61583333
sub008	0		sub058	0	
sub009	1	40.05611111	sub059	0	
sub010	0		sub060	1	23.72611111
sub011	0		sub061	1	6.541666667
sub012	0		sub062	0	
sub013	0		sub063	0	
sub014	0		sub064	0	
sub015	0		sub065	0	
sub016	0		sub066	0	
sub017	1	14.87027778	sub067	0	
sub018	0		sub068	0	
sub019	0		sub069	0	
sub020	0		sub070	1	9.653333333
sub021	0		sub071	0	
sub022	0		sub072	0	
sub023	0		sub073	0	
sub024	0		sub074	0	
sub025	0		sub075	0	
sub026	0		sub076	1	15.99361111
sub027	0		sub077	1	14.11944444
sub028	0		sub078	0	
sub029	0		sub079	1	27.84722222

续下页

表 4.3　问题一 a 结果（续）

患者编号	是否有血肿	血肿扩张的时间	患者编号	是否有血肿	血肿扩张的时间
sub030	0		sub080	1	20.57333333
sub031	0		sub081	1	27.42166667
sub032	0		sub082	0	
sub033	1	30.81305556	sub083	0	
sub034	0		sub084	0	
sub035	0		sub085	0	
sub036	1	39.50111111	sub086	0	
sub037	0		sub087	0	
sub038	1	15.80916667	sub088	0	
sub039	1	29.17611111	sub089	0	
sub040	0		sub090	0	
sub041	0		sub091	0	
sub042	0		sub092	1	11.92472222
sub043	0		sub093	0	
sub044	0		sub094	0	
sub045	0		sub095	1	7.431666667
sub046	0		sub096	0	
sub047	0		sub097	0	
sub048	1	12.86083333	sub098	1	42.75666667
sub049	0		sub099	1	17.67277778
sub050	0		sub100	0	

表 4.4　问题一 b 结果

患者编号	48h 内血肿扩张事件发生的概率	患者编号	48h 内血肿扩张发生的概率
sub001	0.2092	sub081	0.0000
sub002	0.1840	sub082	0.0011
sub003	0.2369	sub083	0.0000
sub004	0.9999	sub084	0.0009
sub005	0.1063	sub085	0.2993
sub006	0.0001	sub086	0.0318

续下页

表 4.4　问题一 b 结果（续）

患者编号	48h 内血肿扩张事件发生的概率	患者编号	48h 内血肿扩张发生的概率
sub007	0.0006	sub087	0.4225
sub008	0.2369	sub088	0.0235
sub009	0.3859	sub089	0.0468
sub010	0.0001	sub090	0.2369
sub011	0.9292	sub091	0.0318
sub012	0.8872	sub092	0.0007
sub013	0.9981	sub093	0.0012
sub014	0.2993	sub094	0.0004
sub015	0.0279	sub095	0.2993
sub016	0.2993	sub096	0.0531
sub017	0.9777	sub097	0.1840
sub018	0.3339	sub098	0.0000
sub019	0.0321	sub099	0.1840
sub020	0.9853	sub100	0.0012
sub021	0.9672	sub101	0.9777
sub022	0.0531	sub102	0.9743
sub023	0.0008	sub103	0.9997
sub024	0.0004	sub104	0.0000
sub025	0.0591	sub105	0.9292
sub026	0.2993	sub106	0.2993
sub027	0.0015	sub107	0.2993
sub028	0.2369	sub108	0.0003
sub029	0.0436	sub109	0.0010
sub030	0.0086	sub110	0.3339
sub031	0.3339	sub111	0.0603
sub032	0.0009	sub112	0.2369
sub033	0.0796	sub113	0.0172
sub034	0.3339	sub114	0.0436
sub035	0.0318	sub115	0.0000
sub036	0.1840	sub116	0.9909
sub037	0.2369	sub117	0.8548

续下页

表 4.4　问题一 b 结果（续）

患者编号	48h 内血肿扩张事件发生的概率	患者编号	48h 内血肿扩张发生的概率
sub038	0.2993	sub118	0.0000
sub039	0.2369	sub119	0.0001
sub040	0.2993	sub120	0.0412
sub041	0.2993	sub121	0.0559
sub042	0.0412	sub122	0.0000
sub043	0.0321	sub123	0.0443
sub044	0.0796	sub124	0.0000
sub045	0.0000	sub125	0.0468
sub046	0.2993	sub126	0.2993
sub047	0.2369	sub127	0.0000
sub048	0.2993	sub128	0.0001
sub049	0.0235	sub129	0.0412
sub050	0.0275	sub130	0.9809
sub051	0.1063	sub131	0.2369
sub052	0.0000	sub132	0.9999
sub053	0.0000	sub133	0.3549
sub054	0.2993	sub134	0.8915
sub055	0.2993	sub135	0.9909
sub056	0.2669	sub136	0.0000
sub057	0.3519	sub137	0.0076
sub058	0.9766	sub138	0.1840
sub059	0.0476	sub139	0.9518
sub060	0.9861	sub140	0.9248
sub061	0.3339	sub141	0.0067
sub062	0.1840	sub142	0.0468
sub063	0.0000	sub143	0.0000
sub064	0.0531	sub144	0.0000
sub065	0.2369	sub145	0.9977
sub066	0.0436	sub146	0.0009
sub067	0.8548	sub147	0.3482
sub068	0.8872	sub148	0.0008

续下页

表 4.4　问题一 b 结果（续）

患者编号	48h 内血肿扩张事件发生的概率	患者编号	48h 内血肿扩张发生的概率
sub069	0.9095	sub149	0.9696
sub070	0.8850	sub150	0.1398
sub071	0.0000	sub151	0.2912
sub072	0.0000	sub152	0.0246
sub073	0.2369	sub153	0.0112
sub074	0.2669	sub154	0.0007
sub075	0.0318	sub155	0.1700
sub076	0.0000	sub156	0.0000
sub077	0.0000	sub157	0.0169
sub078	0.0801	sub158	0.1938
sub079	0.9837	sub159	0.0006
sub080	0.0000	sub160	0.1407

4.2 水肿体积发展模型

4.2.1 高斯曲线拟合水肿进展模型

为了预测周围水肿的发生和进展，本文构建多段函数描述周围水肿扩散随时间的变化规律。下图呈现了离散时间对应的水肿体积散点图。通过观察，水肿演变的趋势为先上升达到一个较高的峰值，再缓慢下降，最终趋于平稳。

首先用 4 次多项式拟合水肿进展。

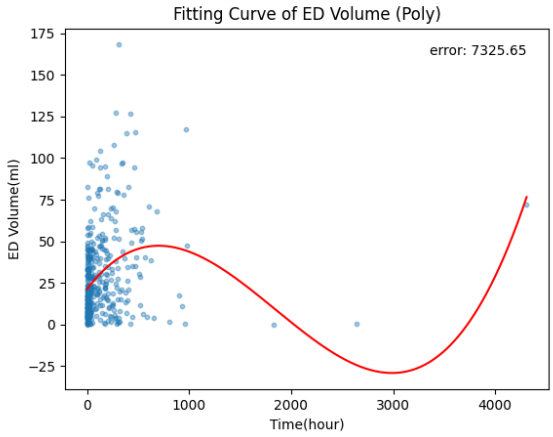

图 4.5　多项式拟合水肿进展曲线

非线性曲线拟合的方法有多项式拟合、样条曲线拟合和高斯拟合等方法。本文得到了

另两种方法的趋势图。

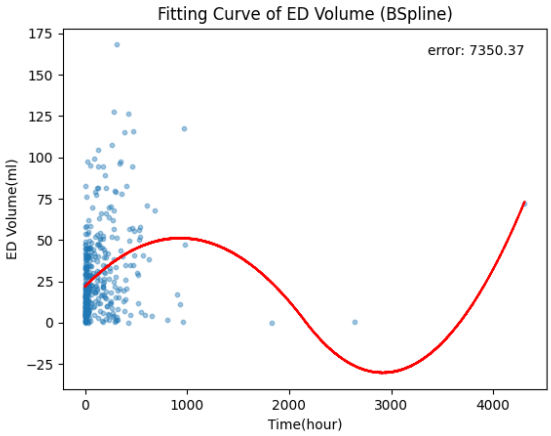

图 4.6　样条曲线拟合水肿进展曲线

样条曲线拟合的原理是用光滑的分段多项式来拟合。每个样条段通常由一个三次多项式来表示。三次样条在每个样条段内是连续光滑的，并且有连续的一阶和二阶导数。即相邻样条段在连接点处的斜率相同，从而确保曲线的光滑性。图中的效果反应了采用多项式拟合的效果较为理想，但是受离群值的影响非常大，如 4000 小时后的突变点。另外本文还采用了信号处理邻域的快速傅里叶变换分析了高频次出现的水肿体积。所以通过对比，本文采用拟合趋势较好的高斯函数拟合全体患者水肿体积随时间进展曲线，

$$f(x) = a_1 e^{a_2 x^2 + a_3 x + a4} \quad , a_1 > 0, a_2 < 0 \tag{4.15}$$

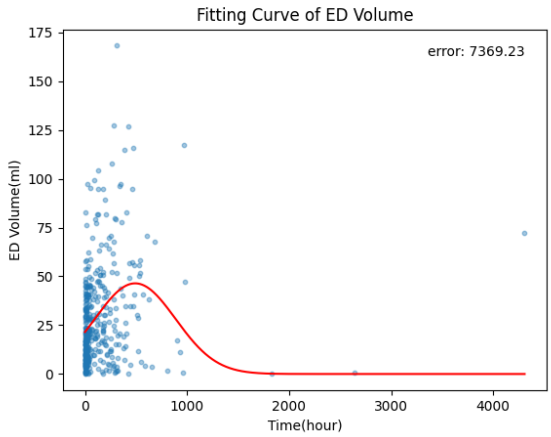

图 4.7　高斯函数拟合水肿进展曲线

4.2.2 病患 Kmeans 亚类选择模型

由于患者个体间差异可能造成其发生周围水肿的演变趋势的不同。同时考虑到年龄、性别等多方面因素对疾病的未知作用，本文首先使用聚类方法将患者聚为 5 类，再对不同的亚类分别拟合。

现有的数据驱动的聚类方法如 Kmeans 聚类、DBSAN 聚类。Kmeans 聚类将数据点划分为 K 个簇，每个簇由其内部的数据点组成，目标是最小化簇内数据点之间的平方距离。

DBSCAN 是一种基于密度的聚类方法，可以识别具有足够高密度的簇，同时将孤立点视为噪声。根据本文数据的特异性，选择 Kmeans 聚类可以较为显著地降低前述拟合模型在亚类预测中的残差表现。

算法 1 K 均值聚类

1: 选择要分成的簇的数量 K.
2: 从数据集中随机选择 K 个数据点作为初始簇的中心.
3: **repeat**
4: 　将每个数据点分配到距离最近的簇中心.
5: 　更新每个簇的中心，计算所有数据点在该簇中的平均值.
6: **until** 簇不再改变或达到最大迭代次数.
7: 输出最终的簇分配结果.

Kmeans 思想为最小化簇内平方和，可以实现簇的紧凑性，即确保每个簇内的数据点都彼此接近：

$$\text{argmin} \sum_{i=1}^{K} \sum_{j=1}^{n_i} ||x_j - \mu_i||^2 \tag{4.16}$$

n_i 是第 i 个簇中的数据点数量。x_j 是第 i 个簇中的第 j 个数据点，表示每个数据点属于哪个簇。μ_i 是第 i 个簇的中心点，由平均数算得。

我们将五种亚类分别采用高斯函数拟合，得到的效果图如下：

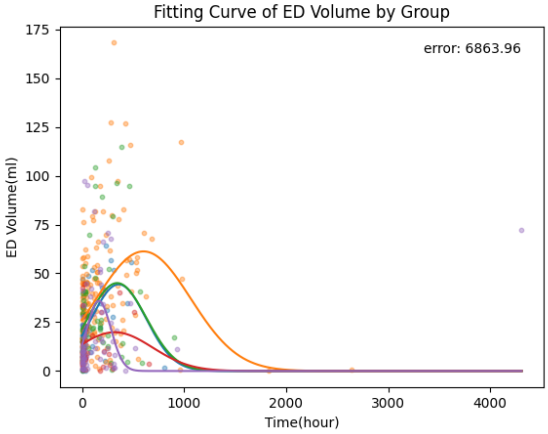

图 4.8　五个亚类的高斯函数拟合水肿进展曲线

由上图可知，红色曲线亚组人群相对水肿体积进展曲线较为平缓，曲线峰值在以上 5 个亚组中最低，即该人群的水肿情况在若干治疗手段干预下能够抑制其增长，然而恢复时间适中；

紫色曲线亚组人群的水肿体积进展曲线在发病初期会产生剧烈增长，然而在短时间内就达到较低的峰值，随后快速下降为接近 0 值的状态，即该人群初期水肿情况发展迅猛，但水肿体积仍然维持在较低范围内，并且能够在短期内迅速恢复；

绿色曲线亚组人群和蓝色曲线亚组人群虽然在个体特征方面存在差异，但是这两个亚组人群的水肿体积进展曲线相似，都在初期快速增长至中等水平，峰值均高于红色和紫色亚组人群，然而恢复时间与红色曲线亚组人群持平；

黄色曲线亚组人群的水肿体积进展曲线的峰值最高、宽度最长，并且在初期的增长速率较快，即该人群在治疗手段干预的条件下的水肿情况恢复最为缓慢，并且水肿情况变化迅速且恶劣，在临床诊断中需要重点关注该亚组人群的水肿情况，以快速恢复患者水肿情况。

计算各亚类残差的具体结果见问题二 b 结果。

4.2.3 基于显著分析的医学关联评价模型

本文进而探寻了不同治疗方法如何水肿体积进展。首先描述单个不同的治疗方法以及一定治疗方案的组合与水肿体积进展的相关性，接着进行显著性水平检验[3]。对未来医师设计治疗方案提供数据水平而非经验的参考。

方差分析（Analysis of Variance，ANOVA）是一种统计方法，用于比较两个或更多组之间的均值是否具有显著差异。其原理基于对总体变异性的分解，以确定组间变异性与组内变异性之间的差异。

假设有 k 个不同组，每个组的观测值个数分别为 $n_1, n_2, ..., n_k$。ANOVA 的原理可以用

以下数学公式表达：

F 值通常用于方差分析，p 值则用于判断检验中观察到的结果是否显著。

总平方和如下：

$$SST = \sum\sum(x - \bar{x})^2 \tag{4.17}$$

其中，x 是观测值，x̄ 是所有观测值的平均值。

组间平方和如下：

$$SSB = \sum n_i(\bar{x}_i - \bar{x})^2 \tag{4.18}$$

其中，n_i 是第 i 组的观测值个数，x_i 是第 i 组观测值的平均值，x̄ 是所有观测值的总体平均值。

组内平方和：

$$SSW = \sum\sum(x - \bar{x}_i)^2 \tag{4.19}$$

其中，x 是观测值，x̄i 是第 i 组观测值的平均值。

组间均方：

$$MSB = \frac{SSB}{k-1} \tag{4.20}$$

组内均方：

$$MSW = \frac{SSW}{N-k} \tag{4.21}$$

其中，N 是总的观测值个数。

最后，使用 F 统计量进行假设检验，计算组间均方与组内均方的比值：

$$F = \frac{MSB}{MSW} \tag{4.22}$$

根据 F 统计量的值和自由度，可以进行假设检验，判断组间均值是否存在显著差异。在进行方差分析时，通过计算 F 统计量来比较组间均方与组内均方的比值。然后，根据 F 统计量和自由度，可以计算出 P 值。

P 值表示观察到的数据与原假设一致的概率。具体来说，它是在原假设为真的情况下，观察到比当前数据更极端结果的概率。

$$p = P(\text{观察到的统计量} \geq \text{观察到的值}|\text{零假设成立}) \tag{4.23}$$

通常，如果 P 值小于预先设定的显著性水平（通常是 0.05 或 0.01），则可以拒绝原假设，认为观察到的数据具有统计显著性，即存在组间均值的显著差异。反之，如果 P 值大于显著性水平，则没有足够的证据拒绝原假设，即认为观察到的数据不具有统计显著性，没有足够的证据表明组间均值存在显著差异。

根据本文提供的脑室引流、止血治疗、降颅压治疗、降压治疗、镇静、镇痛治疗、止吐护胃、营养神经等治疗手段的相关数据，结合上述建立的方差分析模型，计算上述 6 中治疗方法的 F 值、P 值，并根据 F 值和 P 值判断该治疗方法对水肿体积进展的显著性，分析结果如表4.7所示。

4.2.4 基于 Spearsman 系数的相关性评价模型

基于以上建立的方差分析模型，首先分析 6 种治疗手段对血肿体积进展的显著性，结果如表4.8所示。

其次，为了分析血肿体积、水肿体积、治疗手段三者之间的关联性，还需要进一步分析血肿体积和水肿体积之间的相关性。鉴于此，本文采用 Spearman 相关性分析模型[4]。

Spearman 相关系数是一种用于衡量两个变量之间的非线性关系的统计指标，通常用于排名数据或顺序数据。

给定两组数据：X 和 Y，把它们的值按照升序排列并赋予排名，得到排名序列 R_X 和 R_Y。计算 Spearman 相关系数 ρ 如下：

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)} \tag{4.24}$$

Spearman 相关系数主要用来检测变量之间的单调相关关系, 能够测量任何单调函数关系。其基于秩的概念，通过变量排序转换来检测变量之间的关系。其原理和计算公式如下: 设有两个变量 X 和 Y, 各有 n 个样本观测值。对 X 和 Y 分别进行排序, 转换为秩:

X 的样本值:x1, x2, ..., xn

X 的秩:x1’, x2’,..., xn’

Y 的样本值:y1, y2, ..., yn

Y 的秩:y1’, y2’,..., yn’

计算 X 和 Y 的秩之间的相关系数 ρ

其中,d 是对应样本的秩差值: d1 = x1’ - y1’, d2 = x2’ - y2’, ..., dn = xn’ - yn’ 即 $d = R_X - R_Y$。n 是数据点的总数，也就是 X 和 Y 中的观测数。

ρ 的值范围在-1 到 1 之间,1 或-1 表示完全相关,0 表示无相关。对 ρ 进行显著性检验, 得到 p 值。p 值小于显著性水平, 则变量之间存在显著相关。

基于上述基于 Spearman 相关系数的相关性分析模型，将水肿体积和血肿体积数据代入模型当中可得，水肿体积和血肿体积的相关系数 rs=0.3916，p 值为 2.95×10^{-16}，

Spearman 相关系数基于排名而不是原始数据的数值，对于非线性和单调关系的数据更加稳健。

4.2.5 问题二结果

表 4.5　问题二 a 结果

患者编号	水肿体积的进展预测曲线残差	患者编号	水肿体积的进展预测曲线残差
sub001	263.5030888	sub051	79.46524276
sub002	16.07234976	sub052	100.4967232
sub003	55.6587883	sub053	18.5449734
sub004	54.09531228	sub054	44.74943096
sub005	41.12349911	sub055	35.70323641
sub006	229.8011818	sub056	84.94130577
sub007	49.89780589	sub057	27.52972525
sub008	42.7757443	sub058	23.52679266
sub009	27.2696528	sub059	77.97789349
sub010	33.13715356	sub060	126.5099603
sub011	35.52687923	sub061	205.8671436
sub012	48.12523661	sub062	33.38035188
sub013	45.88626466	sub063	201.6859182
sub014	19.76004092	sub064	96.65815715
sub015	85.7143048	sub065	60.4384748
sub016	85.4475488	sub066	59.58664476
sub017	32.32975042	sub067	54.79397592
sub018	65.75840362	sub068	47.43283425
sub019	68.73848408	sub069	101.9915246
sub020	66.55029673	sub070	51.36132742
sub021	45.01691843	sub071	95.88202677
sub022	30.37520113	sub072	33.51195726
sub023	106.3980955	sub073	58.8789429
sub024	68.78359246	sub074	192.6427916
sub025	20.00301892	sub075	43.57540566
sub026	32.03348038	sub076	95.0209261
sub027	62.04849134	sub077	41.47488699
sub028	60.41943889	sub078	75.31330573
sub029	61.19500254	sub079	98.55616151
sub030	180.7015975	sub080	53.0747319
sub031	40.13205964	sub081	154.7738182

续下页

表 4.5　问题二 a 结果（续）

患者编号	水肿体积的进展预测曲线残差	患者编号	水肿体积的进展预测曲线残差
sub032	71.56527312	sub082	21.81796553
sub033	50.69437833	sub083	33.87621067
sub034	107.1464504	sub084	40.13359736
sub035	111.9971899	sub085	161.2505475
sub036	74.26844612	sub086	67.94926546
sub037	58.51849643	sub087	119.8400984
sub038	54.67031594	sub088	12.25913612
sub039	219.7127585	sub089	44.6444122
sub040	90.27984227	sub090	110.6508946
sub041	42.68100578	sub091	48.12840094
sub042	50.10793187	sub092	72.16150228
sub043	21.03242326	sub093	102.7589584
sub044	111.9009216	sub094	63.04248195
sub045	40.64588148	sub095	266.5793462
sub046	40.71437323	sub096	122.3755705
sub047	62.37021552	sub097	58.7170372
sub048	75.16664758	sub098	45.58673027
sub049	64.87396604	sub099	21.75394424
sub050	25.70273456	sub100	32.02867146

表 4.6　问题二 b 结果

患者编号	水肿的进展曲线残差	亚类别	患者编号	水肿的进展曲线残差	亚类别
sub001	230.0261168	1	sub051	107.0305016	1
sub002	20.00495787	0	sub052	75.63839716	1
sub003	41.4347631	1	sub053	15.8905912	2
sub004	55.44162784	0	sub054	70.86758846	1
sub005	46.19944539	0	sub055	29.68311514	1
sub006	220.5070582	4	sub056	91.5649742	2
sub007	45.11048848	0	sub057	28.34733739	2
sub008	42.07862608	0	sub058	26.53086479	2

续下页

表 4.6　问题二 b 结果（续）

患者编号	水肿的进展曲线残差	亚类别	患者编号	水肿的进展曲线残差	亚类别
sub009	25.15795565	0	sub059	58.35540193	4
sub010	27.05317878	3	sub060	121.3078044	1
sub011	20.83610392	4	sub061	199.5180958	2
sub012	44.93545254	0	sub062	38.43624604	2
sub013	32.23357464	4	sub063	162.7963322	1
sub014	15.71488995	0	sub064	53.27958215	4
sub015	87.94466623	2	sub065	51.7410076	1
sub016	81.35244034	2	sub066	67.32847529	1
sub017	51.00350404	4	sub067	45.29889184	4
sub018	72.50058988	0	sub068	37.89788192	1
sub019	39.41517281	0	sub069	76.22922033	4
sub020	57.3168105	0	sub070	13.81006207	4
sub021	68.24695639	1	sub071	121.979676	4
sub022	24.0515065	0	sub072	24.06865152	4
sub023	124.121145	3	sub073	74.1453331	1
sub024	30.48647153	3	sub074	147.9261106	4
sub025	31.89339929	1	sub075	31.44612574	4
sub026	26.33399235	1	sub076	101.9245567	2
sub027	69.49339743	2	sub077	58.07471143	1
sub028	45.00617914	1	sub078	41.69645934	3
sub029	60.21424244	1	sub079	53.23213471	4
sub030	178.6267168	2	sub080	70.14602797	1
sub031	35.40293368	1	sub081	117.8811388	4
sub032	39.34646508	3	sub082	45.83361036	3
sub033	70.88758528	1	sub083	28.97792905	2
sub034	92.36737232	1	sub084	54.3377415	3
sub035	105.060613	2	sub085	157.0439611	1
sub036	95.96820469	1	sub086	55.24493129	4
sub037	55.77226247	1	sub087	120.382304	2
sub038	91.89093971	1	sub088	18.080687	4
sub039	189.1984158	1	sub089	51.11709684	4

续下页

表 4.6　问题二 b 结果（续）

患者编号	水肿的进展曲线残差	亚类别	患者编号	水肿的进展曲线残差	亚类别
sub040	83.66918613	1	sub090	114.1393355	1
sub041	57.53381947	1	sub091	63.38386385	1
sub042	15.28294811	4	sub092	54.42138452	4
sub043	26.16432869	2	sub093	48.34762623	3
sub044	82.23066004	1	sub094	19.6802619	3
sub045	68.0552901	1	sub095	252.8941829	1
sub046	43.11026815	1	sub096	53.95248288	4
sub047	99.4599684	1	sub097	74.30730503	1
sub048	63.09660562	1	sub098	51.90298782	2
sub049	79.98047632	1	sub099	26.2252424	1
sub050	35.35696274	1	sub100	15.0354992	3

表 4.7　问题二 c 结果

F 值	p 值	显著性	治疗方法
4.32764911	0.038130867	一般	脑室引流
1.139895583	0.286315119	不显著	止血治疗
14.19039069	0.000189908	显著	降颅压治疗
4.654507631	0.031564024	一般	降压治疗
0.336203858	0.562353485	不显著	镇静、镇痛治疗
0.530614859	0.466772583	不显著	止吐护胃
1.605787333	0.205818581	不显著	营养神经

表 4.8　问题二 d 结果

F 值	p 值	显著性	治疗方法
25.87998656	5.58618E-07	显著	脑室引流
0.807243236	0.36947436	不显著	止血治疗
12.5504808	0.000442589	显著	降颅压治疗
1.415848455	0.234790191	不显著	降压治疗
0.569038039	0.451083075	不显著	镇静、镇痛治疗

0.22109555	0.638461418	不显著	止吐护胃
1.389516211	0.239184046	不显著	营养神经

4.3 预后预测模型及相关性研究

本部分对对出血性脑卒中的患者进行预后预测，探索影响预测模型效果的关键因素。

4.3.1 基于相关优化的多层感知机的入院预后预测

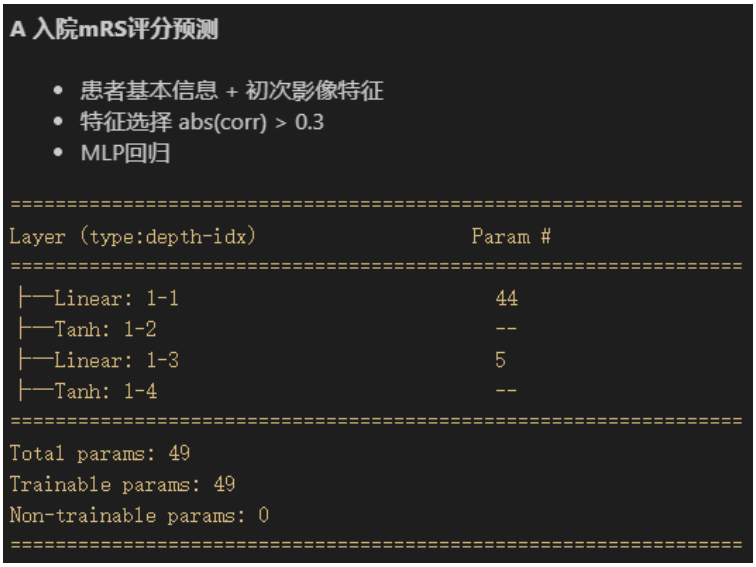

图 4.9　mRS 预测 MLP 参数

上图为优化后的 MLP 模型，使用了患者的基本信息与初次医学影像特征。对特征进行相关性筛选得到绝对相关性大于 0.3 的进行喂入。

原始特征 104 个筛选完毕之后剩余 12 个强特征。线性层和激活层的函数如图所示。

对优化的 MLP 模型进行评价：

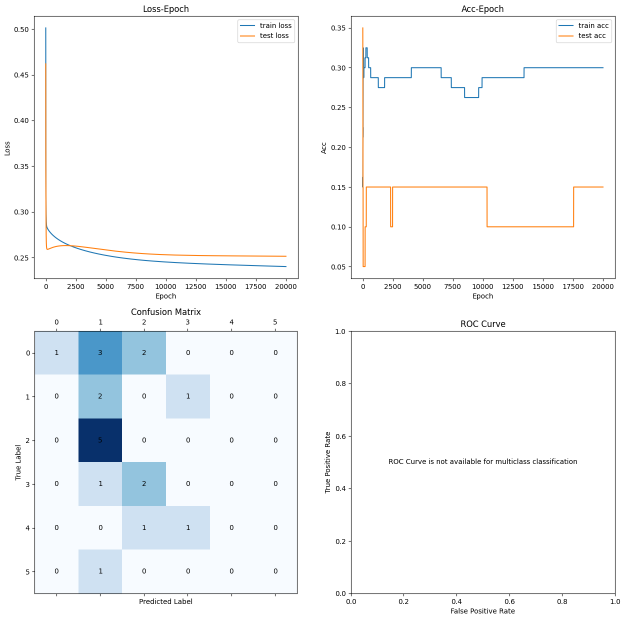

图 4.10　MLP 预测预后 mRS 的结果评价

4.3.2 基于 RNN 回归的出院预后预测

常见的时序预测模型如自回归积分移动平均模型 (ARIMA) 在数据非线性的情景下表现差，长短时记忆网络 (LSTM) 不能在非均匀的时序中取得较好的结果。

所以本文采用能够弥补长时间间隔相关性特点序列信息的循环神经网络构建所有特征作用下的预后预测模型。其中 encoder 影像序列，decoder 为预测值。

以下是循环神经网络 (RNN) 算法的伪代码：

算法 2 RNN 伪代码

初始化 RNN 参数
初始化隐藏状态 h_0
for $t = 1$ 到 T **do**
    从输入序列中获取当前时间步的输入向量 x_t
    计算当前时间步的隐藏状态: $h_t = \text{RNN}(x_t, h_{t-1})$
    预测、损失计算等
对患者 90mRS 分类

已知 100 个患者所有已知临床、治疗。首次及随访的全部影像结果，预测所有含随访影像检查的患者（除 sub101 至 sub130 外）90 天 mRS 评分。本文构建 RNN 预测模型，从而预测所有患者 90 天 mRS。

B 出院mRS评分预测

- 患者基本信息 + 影像序列
- 特征选择 abs(corr) > 0.3
- RNN回归

```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─RNN: 1-1                               176
│    └─weight_ih_10
│    └─weight_hh_10
├─Linear: 1-2                            9
==========================================================================
Total params: 185
Trainable params: 185
Non-trainable params: 0
==========================================================================
```

图 4.11　mRS 预测 RNN 参数

对 RNN 模型进行评价：

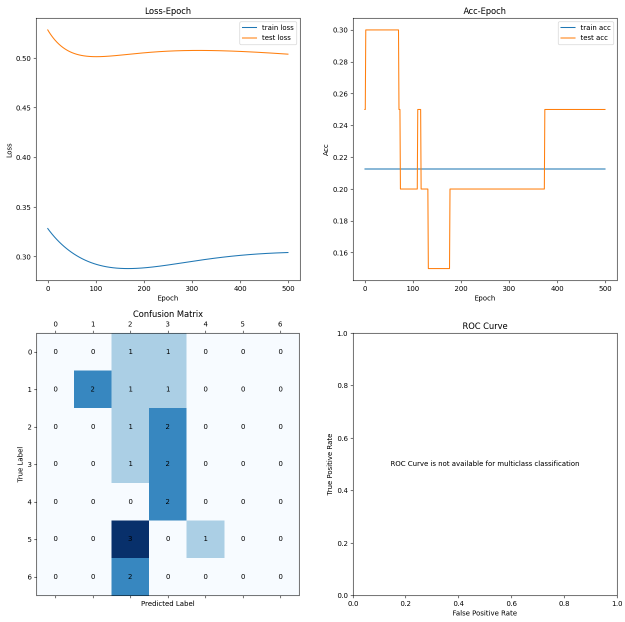

图 4.12　RNN 预测所有患者预后 mRS 的结果评价

4.3.3 基于统计的离散关联规则算法

为探寻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（包括血肿/水肿体积、血肿/水肿位置、信号强度特征、形状特征）等关联关系。参考样本统计的方法，对离散的特征进行关联规则的构建。

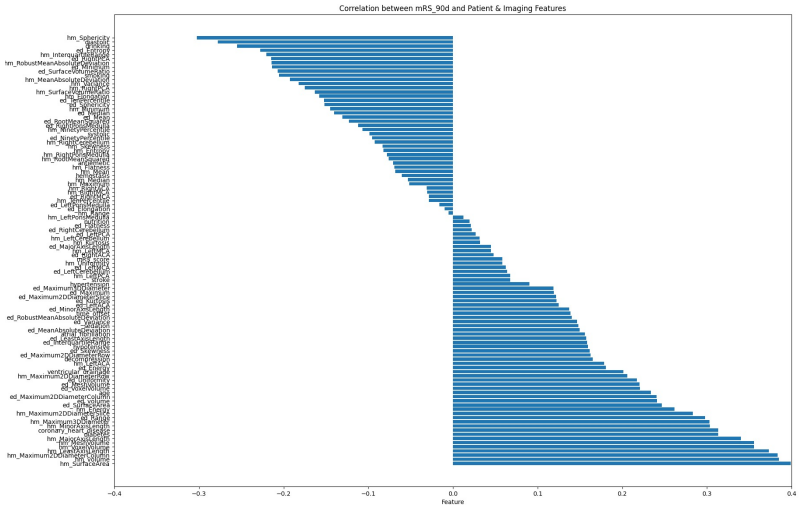

图 4.13　所有特征与 mRS 关联

接着绘制热力图描述特征之间的相关关系。

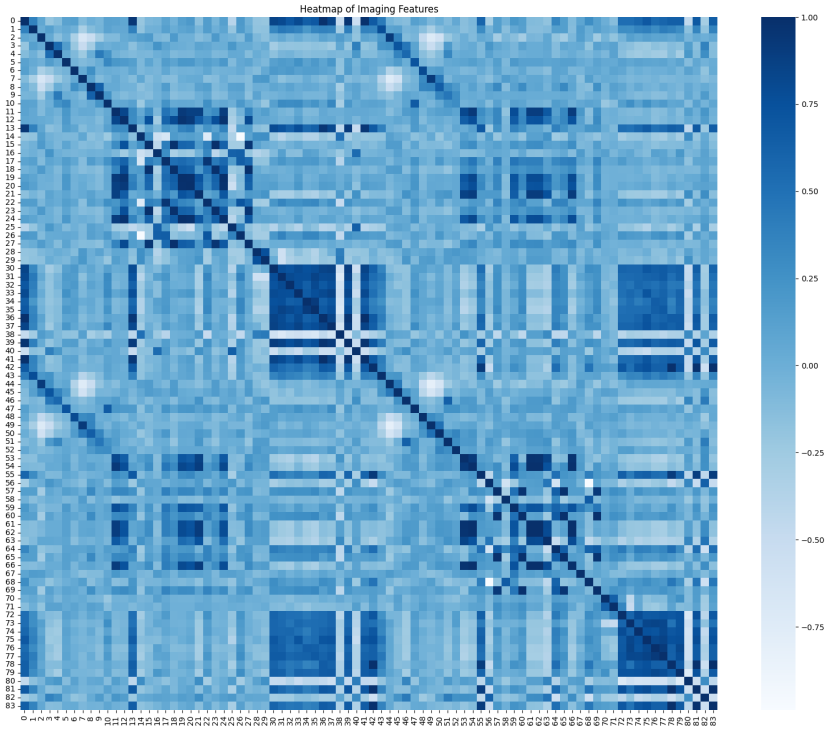

图 4.14　所有特征的关联关系

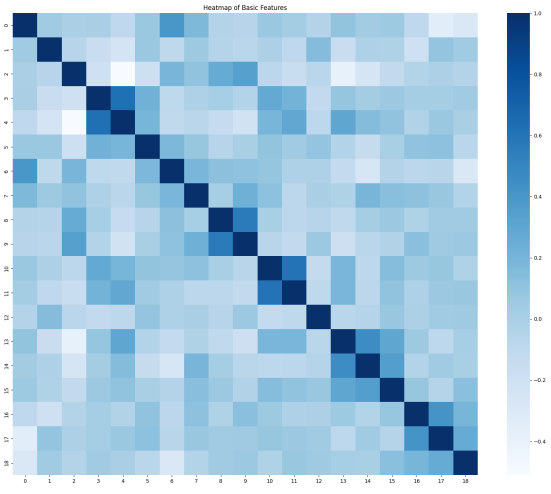

图 4.15　基础特征的关联关系

对于得到的其它特征与 mRS 关联性的柱状图，结合医学科学相关知识[5]，对与 90 天 mRS 相关性较高的特征变量作出以下合理的解释：

个人史

个人史因素如年龄、性别、体重、吸烟史、饮酒史等可能会影响出血性脑卒中患者的预后。年龄较大的患者通常有更差的预后。

疾病史

患者的基础健康状况和疾病史，如高血压、糖尿病、心血管疾病等，也可能会影响出血性脑卒中的预后。

治疗方法

早期和有效的治疗可以改善出血性脑卒中患者的预后。治疗方法包括药物治疗、手术干预、康复等。治疗越早开始，通常预后越好。

影像特征

出血性脑卒中患者的血肿或水肿体积与预后相关系数为。较大的血肿或水肿可能会导致更严重的神经功能损伤，从而影响 90 天后的 mRS 评分。通常情况下，较小的血肿或水肿预后较好。

血肿/水肿的位置与预后的相关系数为，若出血或水肿位于重要的脑区域，如大脑皮层或深部结构，可能会导致更严重的神经功能障碍。

信号强度特征与出血性脑卒中的类型相关。例如，高信号强度的血肿可能暗示新鲜出血，而低信号强度可能暗示较早的出血。这可以对预后的评估产生影响。

出血性脑卒中血肿的形状也对预后产生了影响。不规则形状的血肿可能更难处理，而较规则的血肿可能更容易手术干预。

90mRS 是一个越小越好的值，所以根据关联模型给出的结果图4.13，本文提出了几点临床相关决策如下：

基于关联模型的临床决策建议

1. 应尽早评估患者并确定最合适的治疗方法，这可能包括药物治疗、手术干预或康复计划。治疗越早开始，通常预后越好。其中手术治疗中降颅压治疗和脑室引流可能在未来临床实践中对较大 mRSD 的患者效果较好。另外进行医疗决策的组合可以发挥一定效果。如手术加保守治疗的组合；
2. 应对影像特征进行详细评估，根据评估结果制定个性化的治疗计划。较大或临床经验方法确定的一定形状（星状、块状）或位于重要脑区域的血肿可能需要更积极的治疗，而较小或较早期的血肿可能可以采用更保守的方法；
3. 个人史因素如年龄、体重、心 (脑) 血管病史等可能会严重影响出血性脑卒中的预后。医生应综合考虑这些因素，并在治疗和康复计划中进行相应的调整。对年龄较大的患者 (水肿拟合演变最明显的一种亚类人群) 需要更密切的监测和支持。

4.3.4 问题三结果

表 4.9　问题三 a 结果

患者编号	基于首次影像的（90 天 mRS）	患者编号	基于首次影像的（90 天 mRS）
sub001	3	sub081	3
sub002	3	sub082	3
sub003	4	sub083	3
sub004	4	sub084	2
sub005	2	sub085	4
sub006	5	sub086	2
sub007	2	sub087	2
sub008	3	sub088	2
sub009	2	sub089	4
sub010	3	sub090	2
sub011	2	sub091	2
sub012	3	sub092	2
sub013	2	sub093	5
sub014	3	sub094	2
sub015	2	sub095	2

续下页

表 4.9　问题三 a 结果（续）

患者编号	基于首次影像的（90 天 mRS）	患者编号	基于首次影像的（90 天 mRS）
sub016	2	sub096	3
sub017	2	sub097	2
sub018	2	sub098	1
sub019	3	sub099	3
sub020	2	sub100	2
sub021	3	sub101	3
sub022	2	sub102	3
sub023	2	sub103	2
sub024	1	sub104	2
sub025	2	sub105	2
sub026	3	sub106	5
sub027	2	sub107	2
sub028	3	sub108	2
sub029	3	sub109	3
sub030	2	sub110	6
sub031	3	sub111	5
sub032	2	sub112	2
sub033	2	sub113	2
sub034	3	sub114	3
sub035	3	sub115	6
sub036	3	sub116	2
sub037	3	sub117	2
sub038	2	sub118	2
sub039	3	sub119	2
sub040	2	sub120	2
sub041	4	sub121	2
sub042	1	sub122	2
sub043	2	sub123	2
sub044	2	sub124	2
sub045	2	sub125	5
sub046	4	sub126	2

续下页

表 4.9　问题三 a 结果（续）

患者编号	基于首次影像的（90 天 mRS）	患者编号	基于首次影像的（90 天 mRS）
sub047	1	sub127	2
sub048	2	sub128	2
sub049	2	sub129	2
sub050	2	sub130	5
sub051	3	sub131	4
sub052	2	sub132	4
sub053	2	sub133	2
sub054	3	sub134	2
sub055	3	sub135	3
sub056	3	sub136	2
sub057	2	sub137	2
sub058	4	sub138	2
sub059	2	sub139	2
sub060	3	sub140	2
sub061	3	sub141	3
sub062	2	sub142	3
sub063	1	sub143	2
sub064	2	sub144	3
sub065	4	sub145	2
sub066	4	sub146	2
sub067	2	sub147	2
sub068	2	sub148	2
sub069	5	sub149	3
sub070	2	sub150	2
sub071	2	sub151	3
sub072	2	sub152	5
sub073	2	sub153	3
sub074	2	sub154	2
sub075	1	sub155	2
sub076	2	sub156	3
sub077	5	sub157	2

续下页

表 4.9　问题三 a 结果（续）

患者编号	基于首次影像的（90 天 mRS）	患者编号	基于首次影像的（90 天 mRS）
sub078	2	sub158	3
sub079	2	sub159	3
sub080	4	sub160	3

表 4.10　问题三 b 结果

患者编号	患者的预后指标（90 天 mRS）	患者编号	患者的预后指标（90 天 mRS）
sub001	2	sub081	6
sub002	2	sub082	3
sub003	4	sub083	2
sub004	2	sub084	2
sub005	3	sub085	2
sub006	2	sub086	3
sub007	3	sub087	1
sub008	3	sub088	3
sub009	3	sub089	3
sub010	3	sub090	3
sub011	3	sub091	3
sub012	3	sub092	4
sub013	3	sub093	2
sub014	2	sub094	2
sub015	1	sub095	2
sub016	2	sub096	2
sub017	2	sub097	3
sub018	2	sub098	3
sub019	2	sub099	3
sub020	3	sub100	3
sub021	2	sub101	2
sub022	3	sub102	1
sub023	3	sub103	1
sub024	2	sub104	1

续下页

表 4.10　问题三 b 结果（续）

患者编号	患者的预后指标（90 天 mRS）	患者编号	患者的预后指标（90 天 mRS）
sub025	4	sub105	1
sub026	2	sub106	2
sub027	2	sub107	2
sub028	2	sub108	1
sub029	3	sub109	1
sub030	1	sub110	1
sub031	3	sub111	2
sub032	3	sub112	1
sub033	2	sub113	1
sub034	3	sub114	2
sub035	2	sub115	4
sub036	2	sub116	1
sub037	2	sub117	1
sub038	2	sub118	2
sub039	3	sub119	1
sub040	2	sub120	1
sub041	3	sub121	1
sub042	3	sub122	1
sub043	1	sub123	1
sub044	2	sub124	1
sub045	2	sub125	1
sub046	3	sub126	1
sub047	4	sub127	1
sub048	2	sub128	1
sub049	3	sub129	1
sub050	3	sub130	2
sub051	2	sub131	2
sub052	1	sub132	3
sub053	2	sub133	2
sub054	3	sub134	1
sub055	4	sub135	3

续下页

表 4.10　问题三 b 结果（续）

患者编号	患者的预后指标（90 天 mRS）	患者编号	患者的预后指标（90 天 mRS）
sub056	1	sub136	2
sub057	2	sub137	3
sub058	2	sub138	1
sub059	2	sub139	3
sub060	2	sub140	3
sub061	1	sub141	2
sub062	2	sub142	3
sub063	3	sub143	3
sub064	3	sub144	1
sub065	3	sub145	3
sub066	2	sub146	3
sub067	3	sub147	4
sub068	2	sub148	3
sub069	2	sub149	4
sub070	3	sub150	2
sub071	2	sub151	2
sub072	2	sub152	2
sub073	3	sub153	2
sub074	3	sub154	1
sub075	2	sub155	4
sub076	2	sub156	2
sub077	4	sub157	5
sub078	3	sub158	2
sub079	1	sub159	3
sub080	3	sub160	3

5 模型评价与结论

5.1 模型评价与推广

由于较为可靠的数据预处理和神经网络的设计，本文的实验结果理想。本文今后应该针对稀疏样本采用合理的数据增强算法扩充样本量，并且可以根据丰富的医学影响特征对人脑的出现病灶区域进行获取，这是有一定临床经验积累验证的。从而给机器学习模型带

来更加丰富的特征。此外还要积累临床实操的经验，认真对待稀少病例，提高样本数的可利用率，从而增加神经网络的泛化能力。

值得注意的是对于稀疏样本量的多特征样本集，一定的数据增强方法是无效的，因为其只增加了 TP(True Positive) 的伪样本数量。所以构建鲁棒的神经网络对构建出血性脑卒中智能诊疗模型有着关键影响。

5.2 结论

本文采用了一些在多样本表现良好的机器学习算法，欠拟合程度较大。效果不理想，在面临过拟合风险的情况下，本文使用了部分深度学习的方法进行预后预测。深度学习网络在经过一些列参数调整后减少了过拟合的情况。在现有已知方法改进网络特性的基础上，结合关联算法给出的强特征，对本文的小样本机器学习起到了一定的参考作用。本文能够利用有关医学科学的知识给本文筛选出的强作用特征做出解释。

对出血性脑卒中样本数据偏态分布，使用医学统计学的方差分析和显著性检验往往能取得可解释的结果，并且表现良好。且 Spearsman 相关可以很好的反映特征之间的单调关系。结合数据结果与临床经验，取得了较好的可解释性。

综上，本文研究结果需要在更多的临床实践中进行前瞻性验证。尽管存在局限性，模型训练结果还是达到了预期的水准。

参考文献

[1] 王陇德, 彭斌, 张鸿祺等.《中国脑卒中防治报告 2020》概要 [J]. 中国脑血管病杂志,2022,19(02):136-144.

[2] 袁野. 基于深度学习的多元时序生物医学信号表征方法研究 [D]. 北京工业大学,2019.DOI:10.26935/d.cnki.gbjgu.2019.000366.

[3] 吴景芬, 肖军, 陈祥慧等. 认知功能障碍与脑卒中部位的相关性分析 [J]. 临床合理用药杂志,2009,2(09):1-3.

[4] 张世洪, 吴波, 谈颂. 卒中登记研究中 Barthel 指数和改良的 Rankin 量表的适用性与相关性研究 [J]. 中国循证医学杂志,2004(12):871-874.

[5] CHEN Y, QIN C, CHANG J, et al. A machine learning approach for predicting perihematomal edema expansion in patients with intracerebral hemorrhage[J/OL]. European Radiology, 2022, 33(6): 4052-4062.

附录 A　本文 Python 源程序

A.1　数据处理及特征编码程序

tpyedef.py

```
from dataclasses import dataclass
from collections import namedtuple
from datetime import datetime
from enum import Enum

import numpy as np

__all__ = [
    "Gender",
    "DiseaseHistory",
    "BloodPressure",
    "TreatmentMethod",
    "VoxelFeatures",
    "FirstOrderFeatures",
    "ShapeFeatures",
    "Imaging",
    "Patient",
    "NegativeExponentFunc"
]


class Gender(int, Enum):
    male = 0
    female = 1


DiseaseHistory = namedtuple(
    "DiseaseHistory",
    [
        "hypertension",  # 高血压
        "stroke",  # 中风
        "diabetes",  # 糖尿病
        "atrial_fibrillation",  # 房颤
        "coronary_heart_disease",  # 冠心病
        "smoking",  # 吸烟
```

```
        "drinking"  # 饮酒
    ]
)

BloodPressure = namedtuple(
    "BloodPressure",
    [
        "systolic",  # 收缩压
        "diastolic",  # 舒张压
    ]
)

TreatmentMethod = namedtuple(
    "TreatmentMethod",
    [
        "ventricular_drainage",  # 脑室引流
        "hemostasis",  # 止血治疗
        "decompression",  # 降颅压治疗
        "hypotensive",  # 降压治疗
        "sedation",  # 镇静、镇痛治疗
        "antiemetic",  # 止吐护胃
        "nutrition"  # 营养神经
    ]
)

VoxelFeatures = namedtuple(
    "VoxelFeatures",
    [
        "RightACA",
        "RightMCA",
        "RightPCA",
        "RightPonsMedulla",
        "RightCerebellum",
        "LeftACA",
        "LeftMCA",
        "LeftPCA",
        "LeftPonsMedulla",
        "LeftCerebellum",
    ]
```

```
)

FirstOrderFeatures = namedtuple(
    "FirstOrderFeatures",
    [
        "TenPercentile",
        "NinetyPercentile",
        "Energy",
        "Entropy",
        "InterquartileRange",
        "Kurtosis",
        "Maximum",
        "MeanAbsoluteDeviation",
        "Mean",
        "Median",
        "Minimum",
        "Range",
        "RobustMeanAbsoluteDeviation",
        "RootMeanSquared",
        "Skewness",
        "Uniformity",
        "Variance"
    ]
)

ShapeFeatures = namedtuple(
    "ShapeFeatures",
    [
        "Elongation",
        "Flatness",
        "LeastAxisLength",
        "MajorAxisLength",
        "Maximum2DDiameterColumn",
        "Maximum2DDiameterRow",
        "Maximum2DDiameterSlice",
        "Maximum3DDiameter",
        "MeshVolume",
        "MinorAxisLength",
        "Sphericity",
```

```
        "SurfaceArea",
        "SurfaceVolumeRatio",
        "VoxelVolume"
    ]
)


@dataclass
class Imaging:
    time: datetime | None = None
    time_offset: float | None = None
    serial_number: str = ""
    hm_volume: float | None = None
    hm_voxel_features: VoxelFeatures | None = None
    hm_first_order_features: FirstOrderFeatures | None = None
    hm_shape_features: ShapeFeatures | None = None
    ed_volume: float | None = None
    ed_voxel_features: VoxelFeatures | None = None
    ed_first_order_features: FirstOrderFeatures | None = None
    ed_shape_features: ShapeFeatures | None = None


@dataclass
class Patient:
    sub_id: str
    mRS_90d: int
    age: int
    gender: int  # male: 0, female: 1
    mRS_score: int
    time_offset: float  # 发病到首次检查间隔 / hour
    disease_history: DiseaseHistory
    blood_pressure: BloodPressure
    treatment_method: TreatmentMethod
    imgs: list["Imaging"]


class NegativeExponentFunc(object):
    def __init__(self, params):
        self.params = params
```

```
    def __call__(self, x):
        a0, a1, a2, a3 = self.params
        return a0 * np.exp(a1 * x ** 2 + a2 * x + a3)
```

data_process.py

```
import pickle
import pandas as pd
from typedef import *

def load_patient(discard_empty=True) -> list[Patient]:
    df_1 = pd.read_excel("../raw/表1-患者列表及临床信息.xlsx")
    df_2 = pd.read_excel("../raw/表2-患者影像信息血肿及水肿的
        体积及位置.xlsx")
    df_3_ed = pd.read_excel("../raw/表3-患者影像信息血肿及水肿
        的形状及灰度分布.xlsx", 0)
    df_3_hm = pd.read_excel("../raw/表3-患者影像信息血肿及水肿
        的形状及灰度分布.xlsx", 1)
    df_a1 = pd.read_excel("../raw/附表1-检索表格-流水号vs时间
        _new.xlsx")

    patients = []
    for index in range(df_1.shape[0]):
        basic_info = df_1.iloc[index]

        sub_id = basic_info[0]
        assert sub_id == df_2.iloc[index][0]
        assert sub_id == df_a1.iloc[index][0]

        p = Patient(
            sub_id=basic_info[0],
            mRS_90d=basic_info[1],
            mRS_score=basic_info[6],
            time_offset=basic_info[14],
            age=basic_info[4],
            gender=Gender.male if basic_info[5] == "男" else
                Gender.female,
            disease_history=DiseaseHistory(*basic_info[7:14]),
            blood_pressure=BloodPressure(*(int(v) for v in
                basic_info[15].split("/"))),
```

```
            treatment_method=TreatmentMethod(*basic_info
                [16:23]),
            imgs=[]
        )

        ct_num = df_a1.iloc[index][1]
        for ct_index in range(ct_num):
            serial_number = df_a1.iloc[index][3 + ct_index *
                2]
            voxel_feat_empty = False
            try:
                assert serial_number == df_2.iloc[index][1 +
                    ct_index * 23]
            except (AssertionError, IndexError):
                voxel_feat_empty = True
                print(f"[{p.sub_id}] Voxel feature of {int(
                    serial_number)} is empty")

            row_ed_feat = df_3_ed.loc[df_3_ed["流水号"] ==
                serial_number]
            if row_ed_feat.empty:
                print(f"[{p.sub_id}] ED Shape/FirstOrder
                    features of {int(serial_number)} is empty")
            row_hm_feat = df_3_hm.loc[df_3_hm["流水号"] ==
                serial_number]
            if row_hm_feat.empty:
                print(f"[{p.sub_id}] HM Shape/FirstOrder
                    features of {int(serial_number)} is empty")

            if discard_empty:
                if row_ed_feat.empty or row_hm_feat.empty or
                    voxel_feat_empty:
                    print(f"[{p.sub_id}] Discard Imaging {int(
                        serial_number)}")
                    continue

            img = Imaging(
                time=df_a1.iloc[index][2 + ct_index * 2],
                serial_number=serial_number,
```

```
                hm_volume=df_2.iloc[index][2 + ct_index * 23]
                    if not voxel_feat_empty else None,
                hm_voxel_features=VoxelFeatures(*df_2.iloc[
                    index][3 + ct_index * 23: 13 + ct_index *
                    23]) if not voxel_feat_empty else None,
                hm_first_order_features=FirstOrderFeatures(*
                    row_hm_feat.iloc[0][16:33]) if not
                    row_hm_feat.empty else None,
                hm_shape_features=ShapeFeatures(*row_hm_feat.
                    iloc[0][2:16]) if not row_hm_feat.empty else
                     None,
                ed_volume=df_2.iloc[index][13 + ct_index * 23]
                     if not voxel_feat_empty else None,
                ed_voxel_features=VoxelFeatures(*df_2.iloc[
                    index][14 + ct_index * 23: 24 + ct_index *
                    23]) if not voxel_feat_empty else None,
                ed_first_order_features=FirstOrderFeatures(*
                    row_ed_feat.iloc[0][16:33]) if not
                    row_ed_feat.empty else None,
                ed_shape_features=ShapeFeatures(*row_ed_feat.
                    iloc[0][2:16]) if not row_ed_feat.empty else
                     None
            )
            img.time_offset = p.time_offset \
                if not p.imgs \
                else (img.time - p.imgs[0].time).total_seconds
                    () / 3600 + p.time_offset
            p.imgs.append(img)

        patients.append(p)

    return patients


if __name__ == "__main__":
    pickle.dump(load_patient(discard_empty=True), open("../
        data/patient_data.pkl", "wb"))
```

data.py

```
import pickle
```

```
from pathlib import Path

import numpy as np

from typedef import *

__all__ = [
    "patients",
    "dilate_feat",
    "basic_feat",
    "first_imaging_feat",
    "all_imaging_feat",
    "mrs_feat",
    "basic_feature_names",
    "first_imaging_feature_names",
    "all_imaging_feature_names"
]

if Path("../data/patient_data.pkl").exists():
    patients: list[Patient] = pickle.load(open("../data/
       patient_data.pkl", "rb"))
else:
    from data_process import load_patient

    patients: list[Patient] = load_patient()

if Path("../data/dilate_feature.pkl").exists():
    dilate_feat: np.ndarray = pickle.load(open("../data/
       dilate_feature.pkl", "rb"))
else:
    from q_1_a import dilate_feat

if Path("../data/basic_features.pkl").exists():
    basic_feat: np.ndarray = pickle.load(open("../data/
       basic_features.pkl", "rb"))
else:
    from feature_process import extract_basic_features

    basic_feat: np.ndarray = extract_basic_features()
```

```
if Path("../data/first_imaging_features.pkl").exists():
    first_imaging_feat: np.ndarray = pickle.load(open("../data
       /first_imaging_features.pkl", "rb"))
else:
    from feature_process import extract_first_imaging_features

    first_imaging_feat: np.ndarray =
       extract_first_imaging_features()

if Path("../data/all_imaging_features.pkl").exists():
    all_imaging_feat: np.ndarray = pickle.load(open("../data/
       all_imaging_features.pkl", "rb"))
else:
    from feature_process import extract_all_imaging_features

    all_imaging_feat: np.ndarray =
       extract_all_imaging_features()

if Path("../data/mrs_feature.pkl").exists():
    mrs_feat: np.ndarray = pickle.load(open("../data/
       mrs_feature.pkl", "rb"))
else:
    from feature_process import extract_mrs_feature

    mrs_feat: np.ndarray = extract_mrs_feature()

from feature_process import (
    basic_feature_names,
    first_imaging_feature_names,
    all_imaging_feature_names
)
```

feature_process.py

```
import pickle
from pathlib import Path

import numpy as np
from sklearn.preprocessing import minmax_scale
```

```
from typedef import *

if Path("../data/patient_data.pkl").exists():
    patients: list[Patient] = pickle.load(open("../data/
       patient_data.pkl", "rb"))
else:
    from data_process import load_patient

    patients: list[Patient] = load_patient()

basic_feature_names = [
    "mRS_score",
    "time_offset",
    "age",
    *BloodPressure._fields,
    *DiseaseHistory._fields,
    *TreatmentMethod._fields
]


def extract_basic_features(*, scale_range=(0, 1),
   feature_names=None):
    feat_array = []
    for p in patients:
        feat = [
            p.mRS_score,
            p.time_offset,
            p.age,
            *p.blood_pressure,
            *p.disease_history,
            *p.treatment_method
        ]
        feat_array.append(feat)
    feat = np.array(feat_array)
    feat = minmax_scale(feat, feature_range=scale_range)

    if feature_names:
        filter_ = np.array([feat_name in feature_names for
           feat_name in basic_feature_names])
```

```
        feat = feat[:, filter_]

    return feat


first_imaging_feature_names = [
    "hm_volume",
    *[f"hm_{f_name}" for f_name in VoxelFeatures._fields],
    *[f"hm_{f_name}" for f_name in FirstOrderFeatures._fields
       ],
    *[f"hm_{f_name}" for f_name in ShapeFeatures._fields],
    "ed_volume",
    *[f"ed_{f_name}" for f_name in VoxelFeatures._fields],
    *[f"ed_{f_name}" for f_name in FirstOrderFeatures._fields
       ],
    *[f"ed_{f_name}" for f_name in ShapeFeatures._fields]
]


def extract_first_imaging_features(*, scale_range=(0, 1),
   feature_names=None):
    feat_array = []
    for p in patients:
        if not p.imgs:
            print(f"[{p.sub_id}] has no imaging data")
            continue
        img = p.imgs[0]
        feat = [
            img.hm_volume,
            *img.hm_voxel_features,
            *img.hm_first_order_features,
            *img.hm_shape_features,
            img.ed_volume,
            *img.ed_voxel_features,
            *img.ed_first_order_features,
            *img.ed_shape_features
        ]
        feat_array.append(feat)
    feat = np.array(feat_array)
```

```
    feat = minmax_scale(feat, feature_range=scale_range)

    if feature_names:
        filter_ = np.array([feat_name in feature_names for
           feat_name in first_imaging_feature_names])
        feat = feat[:, filter_]

    return feat


all_imaging_feature_names = basic_feature_names + ["
   imaging_time_offset"] + first_imaging_feature_names


def extract_all_imaging_features(*, scale_range=(0, 1),
   feature_names=None):
    img_feat_array = []
    for p in patients:
        for img in p.imgs:
            feat = [
                img.time_offset,
                img.hm_volume,
                *img.hm_voxel_features,
                *img.hm_first_order_features,
                *img.hm_shape_features,
                img.ed_volume,
                *img.ed_voxel_features,
                *img.ed_first_order_features,
                *img.ed_shape_features
            ]
            img_feat_array.append(feat)

    img_feat = np.array(img_feat_array)
    img_feat = minmax_scale(img_feat, feature_range=
       scale_range)

    p_basic_feat = extract_basic_features()

    # concat basic feats & img feats
```

```
    feat_array = []
    img_feat_idx = 0
    for idx, p in enumerate(patients):
        if not p.imgs:
            print(f"[{p.sub_id}] has no imaging data")
            continue

        p_feat_array = []
        for _ in range(len(p.imgs)):
            feat = np.concatenate((p_basic_feat[idx], img_feat
                [img_feat_idx]))
            p_feat_array.append(feat)
            img_feat_idx += 1

        p_feat = np.array(p_feat_array)
        if feature_names:
            filter_ = np.array([feat_name in feature_names for
                feat_name in all_imaging_feature_names])
            p_feat = p_feat[:, filter_]

        feat_array.append(p_feat)

    return feat_array


def extract_mrs_feature(scale_range=(-1, 1)):
    feat_array = [p.mRS_90d for p in patients[:100]]
    feat = np.array(feat_array)
    feat = minmax_scale(feat, feature_range=scale_range)

    return feat


if __name__ == '__main__':
    pickle.dump(extract_basic_features(), open("../data/
        basic_features.pkl", "wb"))
    pickle.dump(extract_first_imaging_features(), open("../
        data/first_imaging_features.pkl", "wb"))
    pickle.dump(extract_all_imaging_features(), open("../data/
```

```
        all_imaging_features.pkl", "wb"))
    pickle.dump(extract_mrs_feature(), open("../data/
        mrs_feature.pkl", "wb"))
```

A.2 第 1a 问程序

q_1_a.py

```
import pickle
from pathlib import Path

import pandas as pd
import numpy as np

from typedef import *
from data import patients

patients = patients[:100]


def is_dilated(patient: Patient):
    flag = False
    dilate_time = None
    base_hm_volume = patient.imgs[0].hm_volume
    for img in patient.imgs[1:]:
        if img.time_offset > 48:
            break

        increase = img.hm_volume - base_hm_volume
        if increase > 6000 or increase / base_hm_volume >
            0.33:
            flag = True
            dilate_time = img.time_offset
            break
    return int(flag), dilate_time


flag_array = [[p.sub_id, *is_dilated(p)] for p in patients]
df = pd.DataFrame(flag_array, columns=["sub_id", "is_dilated",
    "dilate_time"])
dilate_feat = np.array(df["is_dilated"])
```

```
if __name__ == "__main__":
    Path("../result/q_1").mkdir(exist_ok=True, parents=True)
    df.to_excel("../result/q_1/q_1_a.xlsx", index=False)
    pickle.dump(dilate_feat, open("../data/dilate_feature.pkl"
        , "wb"))
```

A.3 第 1b 问程序

q_1_b.py

```
from pathlib import Path

import numpy as np
import pandas as pd
from scipy.stats import pearsonr
import torch
import torch.nn as nn
from torch.optim import Adam, SGD
from torchsummary import summary
from sklearn.model_selection import train_test_split

from data import basic_feat, dilate_feat, basic_feature_names
from feature_process import extract_basic_features
from visual import plot_training_process


def select_features(threshold=0.1) -> pd.DataFrame:
    df = pd.DataFrame(
        np.concatenate((basic_feat[:100], dilate_feat.reshape
            (-1, 1)), axis=1),
        columns=basic_feature_names + ["is_dilated"]
    )
    corr_map = {}
    for feat in df.columns:
        corr_map[feat] = pearsonr(df[feat], df["is_dilated"])
    df_corr = pd.DataFrame(corr_map).T
    df_corr = df_corr.rename(columns={0: "correlation", 1: "p-
        value"})
    df_corr = df_corr.reindex(df_corr["correlation"].
        sort_values(ascending=False).index)
```

```
    feature_names = df_corr[df_corr["correlation"].abs() >
       threshold].index.tolist()
    return feature_names[1:]


device = "cuda" if torch.cuda.is_available() else "cpu"
basic_feat = extract_basic_features(feature_names=
   select_features())
x_train, x_test, y_train, y_test = train_test_split(basic_feat
   [:100], dilate_feat, test_size=0.2)

x_train = torch.tensor(x_train, dtype=torch.float32, device=
   device)
y_train = torch.tensor(y_train, dtype=torch.float32, device=
   device).reshape(-1, 1)
x_test = torch.tensor(x_test, dtype=torch.float32, device=
   device)
y_test = torch.tensor(y_test, dtype=torch.float32, device=
   device).reshape(-1, 1)

model = nn.Sequential(
    nn.Linear(basic_feat.shape[1], 4),
    nn.ReLU(),
    nn.Linear(4, 1),
    nn.Sigmoid()
)
summary(model)
model.to(device)

learning_rate = 1e-5
n_epochs = 5000
loss_fn = nn.MSELoss()
optimizer = Adam(model.parameters(), lr=learning_rate, betas
   =(0.9, 0.999), eps=1e-8)


def train():
    train_losses = []
    test_losses = []
```

```
    train_acc = []
    test_acc = []

    for t in range(n_epochs):
        y_pred = model(x_train)

        loss = loss_fn(y_pred, y_train)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        train_losses.append(loss.item())
        train_acc.append((y_pred.round() == y_train).sum().
           item() / len(y_train))

        with torch.no_grad():
            test_y_pred = model(x_test)
            test_loss = loss_fn(test_y_pred, y_test)
            test_losses.append(test_loss.item())
            test_acc.append((test_y_pred.round() == y_test).
               sum().item() / len(y_test))

        if t % 100 == 99:
            print(f"Epoch {(t + 1)}", loss.item())
            print(f"Loss: {loss.item():.4f}, Acc: {train_acc
               [-1]:.4f}, "
                  f"Test Loss: {test_loss.item():.4f}, Test
                     Acc: {test_acc[-1]:.4f}\n")

    plot_training_process(
        train_losses, test_losses, train_acc, test_acc,
        y_test.cpu().numpy(), test_y_pred.detach().round().cpu
           ().numpy(), test_y_pred.detach().cpu().numpy(),
        dump_args_to="../result/q_1/q_1_b-training_process-mlp
           .pkl",
        save_img_to="../result/q_1/q_1_b-training_process-mlp.
           png"
    )
```

```
def predict():
    x = torch.tensor(basic_feat, dtype=torch.float32, device=
       device)
    with torch.no_grad():
        y = model(x)
    y = y.detach().cpu().numpy().reshape(-1)

    res = [f"{p:.4f}" for p in y]
    df = pd.DataFrame(res, columns=["pred_dilate_prob"])
    df.to_excel("../result/q_1/q_1_b-answer.xlsx", index=False
       )


def load():
    model.load_state_dict(torch.load("../result/q_1/q_1_b-
       model-mlp.pt"))


def save():
    torch.save(model.state_dict(), "../result/q_1/q_1_b-model-
       mlp.pt")


if __name__ == '__main__':
    Path("../result/q_1").mkdir(exist_ok=True, parents=True)

    train()
    save()

    load()
    predict()
```

q1_b.py

```
from sklearn.model_selection import GridSearchCV,
   train_test_split, LearningCurveDisplay
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import normalize, minmax_scale,
   scale
import matplotlib.pyplot as plt

import numpy as np

import pickle

from typedef import *

train_set: list[Patient] = pickle.load(open("../data/train_set
   .pkl", "rb"))

models = {
    "SVC": {
        "model": SVC(),
        "param_grid": {
            "C": [0.1, 1, 10, 100, 1000],
            "gamma": [1, 0.1, 0.01, 0.001, 0.0001],
            "kernel": ["rbf", "linear", "poly", "sigmoid"]
        }
    },
    "LR": {
        "model": LogisticRegression(),
        "param_grid": {
            "C": [0.1, 1, 10, 100],
            "penalty": ["l1", "l2"],
            "solver": ["liblinear", "saga"],
            "max_iter": [10000]
        }
    },
    "RF": {
        "model": RandomForestRegressor(),
        "param_grid": {
            "n_estimators": [10, 100, 1000],
            "max_features": ["log2", "sqrt"],
            "max_depth": [10, 100, 1000],
            "min_samples_split": [2, 5, 10],
```

```
            "min_samples_leaf": [1, 2, 4]
        }
    },
    "MLP": {
        "model": MLPRegressor(),
        "param_grid": {
            "hidden_layer_sizes": [(50,), (100,), (50, 50),
               (100, 100)],
            "activation": ["tanh", "relu"],
            "solver": ["sgd", "adam"],

            "alpha": [0.0001, 0.05],
            "learning_rate": ["constant", "adaptive"],
        }

    }
}


def balanced_subsample(x, y, subsample_size=1.0):
    class_xs = []
    min_elems = None

    for yi in np.unique(y):
        elems = x[(y == yi)]
        class_xs.append((yi, elems))
        if min_elems == None or elems.shape[0] < min_elems:
            min_elems = elems.shape[0]

    use_elems = min_elems
    if subsample_size < 1:
        use_elems = int(min_elems * subsample_size)

    xs = []
    ys = []

    for ci, this_xs in class_xs:
        if len(this_xs) > use_elems:
            np.random.shuffle(this_xs)
```

```
            x_ = this_xs[:use_elems]
            y_ = np.empty(use_elems)
            y_.fill(ci)

            xs.append(x_)
            ys.append(y_)

    xs = np.concatenate(xs)
    ys = np.concatenate(ys)

    return xs, ys


def extract_feature(patient: Patient):
    ct = patient.ct[0]
    return [
            ct.hm_shape_features.SurfaceVolumeRatio,
            ct.ed_shape_features.LeastAxisLength,
            patient.age,
            patient.blood_pressure.systolic,
            patient.blood_pressure.diastolic,
            patient.disease_history.hypertension,
            patient.treatment_method.hypotensive,
            ct.hm_volume,
            ct.ed_volume,
    ]


X = [extract_feature(p) for p in train_set]
X = np.array(X)
X = minmax_scale(X)
# X[:, -2:] = np.log(X[:, -2:])
y = [int(p.dilatation) for p in train_set]
# one hot
y = np.array(y)
y_ = np.eye(2)[y]
```

```
def train():
    x_train, x_test, y_train, y_test = train_test_split(X, y,
       test_size=0.2)
    # model, param_grid = models[model_name]["model"], models[
       model_name]["param_grid"]

    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(10, 6),
        sharey=True)

    for idx, (model_name, model_config) in enumerate(models.
       items()):
        grid_search = GridSearchCV(model_config["model"],
           model_config["param_grid"], cv=5, n_jobs=-1, verbose
           =2)
        # grid_search.fit(x_train, y_train)
        ax = axs[idx // 2][idx % 2]
        LearningCurveDisplay.from_estimator(grid_search, X=
           x_train, y=y_train, ax=ax)

        handles, label = ax.get_legend_handles_labels()
        ax.legend(handles[:2], ["Training Score", "Test Score"
           ])
        ax.set_title(f"Learning Curve for {model_name}")

    plt.show()
    return
    # print("best params: ", grid_search.best_params_)
    # print("best score: ", grid_search.best_score_)

    accuracy = grid_search.best_estimator_.score(x_test,
       y_test)
    y_pred = grid_search.best_estimator_.predict(x_test)

    print("accuracy: ", accuracy)
    print("test result:")
    print(y_test)
    print(y_pred)

    # save model
```

```
    pickle.dump(grid_search.best_estimator_, open("../data/
        model.pkl", "wb"))


def simple_train():
    x_train, x_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2)
    weight = 1 / np.array([y_train.count(0), y_train.count(0)
        ])
    sample_weight = np.array([weight[i] for i in y_train])
    model = SVC()
    model.fit(x_train, y_train, sample_weight=sample_weight)

    print(model.score(x_test, y_test))
    print(y_test)
    print(model.predict(x_test))


def evaluate():
    model: SVC = pickle.load(open("../data/model.pkl", "rb"))

    y_pred = model.predict(X)
    print(y_pred, y)


if __name__ == "__main__":
    train()
    # simple_train()
```

A.4 第 2a 问程序

q_2_a.py

```
from pathlib import Path

import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures,
   SplineTransformer
from scipy.optimize import leastsq
```

```
from data import patients
from visual import plot_fitting_curve
from typedef import *

patients = patients[:100]

x_time = []
y_ed_volume = []
for p in patients:
    for i in p.imgs:
        x_time.append(i.time_offset)
        y_ed_volume.append(i.ed_volume / 1000)

x_time = np.array(x_time)
y_ed_volume = np.array(y_ed_volume)


def fit_negative_exponent(X, Y, p0=None):
    def fitting_func(p, x):
        a1, a2, a3, a4 = p
        return a1 * np.exp(a2 * x ** 2 + a3 * x + a4)

    def fitting_error(p, x, y):
        return fitting_func(p, x) - y

    params, *_ = leastsq(
        fitting_error,
        p0 or np.array([1, -1, 0, 0]),
        args=(X, Y)
    )

    func = NegativeExponentFunc(params)
    error = np.sum(np.abs(fitting_error(params, X, Y)))

    return func, error


def fit_poly(X, Y, degree=3):
```

```
    poly_transform = PolynomialFeatures(degree)

    X = X.reshape(-1, 1)
    X = poly_transform.fit_transform(X)

    model = LinearRegression()
    model.fit(X, Y)

    def func(x):
        x = x.reshape(-1, 1)
        x = poly_transform.fit_transform(x)

        return model.predict(x)

    error = np.sum(np.abs(model.predict(X) - Y))

    return func, error


def fit_bspline(X, Y, n_knots=5, degree=3):
    bspline_transform = SplineTransformer(n_knots=3, degree=2)

    X = X.reshape(-1, 1)
    X = bspline_transform.fit_transform(X)

    model = LinearRegression()
    model.fit(X, Y)

    def func(x):
        x = x.reshape(-1, 1)
        x = bspline_transform.fit_transform(x)

        return model.predict(x)

    error = np.sum(np.abs(model.predict(X) - Y))

    return func, error
```

```
def calc_error(func, patient: Patient):
    X = np.array([img.time_offset for img in patient.imgs])
    Y = np.array([img.ed_volume / 1000 for img in patient.imgs
       ])
    return np.sum(np.abs(func(X) - Y))


if __name__ == '__main__':
    Path("../result/q_2").mkdir(exist_ok=True, parents=True)

    f, err = fit_negative_exponent(
        x_time, y_ed_volume,
        p0=[10, -1e-2, 0, 0]
    )
    plot_fitting_curve(
        [x_time], [y_ed_volume], [f],
        title="Fitting Curve of ED Volume",
        x_label="Time(hour)", y_label="ED Volume(ml)",
        text=f"error: {err:.2f}",
        save_img_to="../result/q_2/q_2_a-curve-ed_volume.png",
        dump_args_to="../result/q_2/q_2_a-curve-ed_volume.pkl"
    )
    sub_ids = [p.sub_id for p in patients]
    errors = [calc_error(f, p) for p in patients]
    df = pd.DataFrame({"sub_id": sub_ids, "error": errors})
    df.to_excel("../result/q_2/q_2_a-error.xlsx", index=False)

    f_poly, err_poly = fit_poly(x_time, y_ed_volume)
    plot_fitting_curve(
        [x_time], [y_ed_volume], [f_poly],
        title="Fitting Curve of ED Volume (Poly)",
        x_label="Time(hour)", y_label="ED Volume(ml)",
        text=f"error: {err_poly:.2f}",
        save_img_to="../result/q_2/q_2_a-curve-ed_volume-poly.
           png"
    )

    f_bspline, err_bspline = fit_bspline(x_time, y_ed_volume)
    plot_fitting_curve(
```

```
        [x_time], [y_ed_volume], [f_bspline],
        title="Fitting Curve of ED Volume (BSpline)",
        x_label="Time(hour)", y_label="ED Volume(ml)",
        text=f"error: {err_bspline:.2f}",
        save_img_to="../result/q_2/q_2_a-curve-ed_volume-
           bspline.png"
    )
```

A.5 第 2b 问程序

q_2_b.py

```
import pickle
from pathlib import Path

import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

from q_2_a import fit_negative_exponent, calc_error
from data import patients, basic_feat
from visual import plot_fitting_curve

patients = patients[:100]
feat = basic_feat[:100]

n_clusters = 5


def cluster():
    kmeans = KMeans(n_clusters)
    kmeans.fit(feat)
    pickle.dump(kmeans, open("../result/q_2/q_2_b-kmeans.pkl",
        "wb"))
    return kmeans.labels_


def build_groups(labels_):
    times = [[] for _ in range(n_clusters)]
    ed_volumes = [[] for _ in range(n_clusters)]
```

```
    for idx, p in enumerate(patients):
        cluster_idx = labels_[idx]
        for img in p.imgs:
            times[cluster_idx].append(img.time_offset)
            ed_volumes[cluster_idx].append(img.ed_volume /
               1000)

    times = [np.array(t) for t in times]
    ed_volumes = [np.array(v) for v in ed_volumes]
    return list(zip(times, ed_volumes))


def fit_curve(group):
    x_time, y_ed_volume = group
    x_time = np.array(x_time)
    y_ed_volume = np.array(y_ed_volume)

    return fit_negative_exponent(
        x_time, y_ed_volume,
        p0=[10, -1e-2, 0, 0]
    )


if __name__ == '__main__':
    Path("../result/q_2").mkdir(exist_ok=True, parents=True)

    labels = cluster()
    groups = build_groups(labels)
    curves = [fit_curve(g) for g in groups]

    xs, ys = zip(*groups)
    funcs, errs = zip(*curves)
    err = np.sum(errs)
    plot_fitting_curve(
        xs, ys, funcs,
        title="Fitting Curve of ED Volume by Group",
        x_label="Time(hour)", y_label="ED Volume(ml)",
        text=f"error: {err:.2f}",
        save_img_to="../result/q_2/q_2_b-curve-ed_volume_group
```

```
            .png",
        dump_args_to="../result/q_2/q_2_b-curve-
            ed_volume_group.pkl"
    )

    sub_ids = [p.sub_id for p in patients]
    errors = [calc_error(funcs[labels[idx]], p) for idx, p in
        enumerate(patients)]
    df = pd.DataFrame({"sub_id": sub_ids, "error": errors, "
        group": labels})
    df.to_excel("../result/q_2/q_2_b-error.xlsx", index=False)
```

A.6 第 2c d 问程序

q_2_cd.py

```
from pathlib import Path

import pandas as pd
from scipy.stats import f_oneway, spearmanr

from data import patients

patients = patients[:100]
therapies = ["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"]


def significance(f, p):
    if p < 0.01 and f > 4:
        return "显著"
    if p < 0.05 and f > 4:
        return "一般"
    return "不显著"


def variance_analysis_ed():
    res = []
    for therapy_idx in range(len(patients[0].treatment_method)
        ):
        group_1 = []
```

```
        group_2 = []

        for p in patients:
            for img in p.imgs:
                if p.treatment_method[therapy_idx]:
                    group_1.append(img.ed_volume)
                else:
                    group_2.append(img.ed_volume)

        f, p = f_oneway(group_1, group_2)
        res.append((f, p, significance(f, p)))

    df = pd.DataFrame(res, columns=["F值", "p值", "显著性"])
    df["治疗方法"] = therapies
    df.to_excel("../result/q_2/q_2_c-sign-ed.xlsx", index=
       False)


def variance_analysis_hm():
    res = []
    for therapy_idx in range(len(therapies)):
        group_1 = []
        group_2 = []

        for p in patients:
            for img in p.imgs:
                if p.treatment_method[therapy_idx]:
                    group_1.append(img.hm_volume)
                else:
                    group_2.append(img.hm_volume)

        f, p = f_oneway(group_1, group_2)
        res.append((f, p, significance(f, p)))

    df = pd.DataFrame(res, columns=["F值", "p值", "显著性"])
    df["治疗方法"] = therapies
    df.to_excel("../result/q_2/q_2_d-sign-hm.xlsx", index=
       False)
```

```
def correlation():
    hm_volumes = []
    ed_volumes = []
    for p in patients:
        for img in p.imgs:
            hm_volumes.append(img.hm_volume)
            ed_volumes.append(img.ed_volume)

    corr, p = spearmanr(hm_volumes, ed_volumes)
    Path("../result/q_2/q_2_d-corr.txt").write_text(f"
       correlation: {corr}\np-value: {p}")


if __name__ == '__main__':
    Path("../result/q_2").mkdir(exist_ok=True, parents=True)

    variance_analysis_hm()
    variance_analysis_ed()
    correlation()
```

A.7 第 3a 问程序

q_3_a.py

```
from pathlib import Path

import pandas as pd
import torch
import torch.nn as nn
from torch.optim import Adam, SGD, Adagrad
from torchsummary import summary
from sklearn.model_selection import train_test_split

from data import mrs_feat
from feature_process import extract_first_imaging_features
from visual import plot_training_process
from q_3_c import filter_features

device = "cuda" if torch.cuda.is_available() else "cpu"
```

```
feature_names = filter_features(threshold=0.3)
first_imaging_feat = extract_first_imaging_features(
   feature_names=feature_names)
x_train, x_test, y_train, y_test = train_test_split(
   first_imaging_feat[:100], mrs_feat, test_size=0.2)

x_train = torch.tensor(x_train, dtype=torch.float32, device=
   device)
y_train = torch.tensor(y_train, dtype=torch.float32, device=
   device).reshape(-1, 1)
x_test = torch.tensor(x_test, dtype=torch.float32, device=
   device)
y_test = torch.tensor(y_test, dtype=torch.float32, device=
   device).reshape(-1, 1)


def _label(y):
    return (y * 3 + 3).round()


label_train = _label(y_train.detach())
label_test = _label(y_test.detach())

model = nn.Sequential(
    nn.Linear(first_imaging_feat.shape[1], 4),
    nn.Tanh(),
    nn.Linear(4, 1),
    nn.Tanh()
)
summary(model)
model.to(device)

learning_rate = 5e-3
n_epochs = 20000
loss_fn = nn.MSELoss()
optimizer = Adagrad(model.parameters(), lr=learning_rate)


def train():
```

```
    train_losses = []
    test_losses = []
    train_acc = []
    test_acc = []

    for t in range(n_epochs):
        y_pred = model(x_train)

        loss = loss_fn(y_pred, y_train)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        train_losses.append(loss.item())
        train_acc.append((_label(y_pred) == label_train).sum()
           .item() / len(y_train))

        with torch.no_grad():
            test_y_pred = model(x_test)
            test_loss = loss_fn(test_y_pred, y_test)
            test_losses.append(test_loss.item())
            test_acc.append((_label(test_y_pred) == label_test
               ).sum().item() / len(y_test))

        if t % 100 == 99:
            print(f"Epoch {(t + 1)}", loss.item())
            print(f"Loss: {loss.item():.4f}, Acc: {train_acc
               [-1]:.4f}, "
                  f"Test Loss: {test_loss.item():.4f}, Test
                     Acc: {test_acc[-1]:.4f}\n")

    plot_training_process(
        train_losses, test_losses, train_acc, test_acc,
        label_test.cpu().numpy(), _label(test_y_pred).cpu().
           numpy(), test_y_pred.detach().cpu().numpy(),
        dump_args_to="../result/q_3/q_3_a-training_process-mlp
           .pkl",
        save_img_to="../result/q_3/q_3_a-training_process-mlp.
```

```
            png"
    )


def predict():
    x = torch.tensor(first_imaging_feat, dtype=torch.float32,
       device=device)
    with torch.no_grad():
        y = model(x)

    labels = _label(y).cpu()

    df = pd.DataFrame(labels, columns=["mRS_90d"])
    # lack of sub131 & sub132
    df["sub_id"] = [f"sub{i if i < 131 else i + 2:#03d}" for i
        in range(1, len(labels) + 1)]
    df.to_excel("../result/q_3/q_3_a-answer.xlsx", index=False
       )


def load():
    model.load_state_dict(torch.load("../result/q_3/q_3_a-
       model-mlp.pt"))


def save():
    torch.save(model.state_dict(), "../result/q_3/q_3_a-model-
       mlp.pt")


if __name__ == '__main__':
    Path("../result/q_3").mkdir(exist_ok=True, parents=True)

    train()
    save()

    load()
    predict()
```

A.8 第 3b 问程序

q_3_b.py

```
from pathlib import Path

import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.optim import Adam, SGD
from torchsummary import summary

from data import mrs_feat
from feature_process import extract_all_imaging_features
from visual import plot_training_process
from q_3_c import filter_features

device = "cuda" if torch.cuda.is_available() else "cpu"


def _label(y):
    return (y * 3 + 3).round()


def _split_train_test(samples):
    np.random.shuffle(samples)
    x_feat, y_feat = zip(*samples)
    _x_train, _x_test = list(x_feat[:80]), list(x_feat[80:])
    _y_train, _y_test = list(y_feat[:80]), list(y_feat[80:])
    for i in range(len(_x_train)):
        _x_train[i] = torch.tensor(_x_train[i], dtype=torch.
           float32, device=device)
    for i in range(len(_x_test)):
        _x_test[i] = torch.tensor(_x_test[i], dtype=torch.
           float32, device=device)
    _y_train = torch.tensor(_y_train, dtype=torch.float32,
       device=device).reshape(-1, 1)
    _y_test = torch.tensor(_y_test, dtype=torch.float32,
       device=device).reshape(-1, 1)
    return _x_train, _x_test, _y_train, _y_test
```

```
feature_names = filter_features(threshold=0.3)
all_imaging_feat = extract_all_imaging_features(feature_names=
   feature_names)
x_train, x_test, y_train, y_test = _split_train_test(list(zip(
   all_imaging_feat[:100], mrs_feat)))

label_train = _label(y_train.detach()).squeeze()
label_test = _label(y_test.detach()).squeeze()


class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers,
       output_size):
        super().__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers

        self.rnn = nn.RNN(input_size, hidden_size, num_layers,
            batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, self.hidden_size,
           device=device)

        out, _ = self.rnn(x, h0)
        out = self.fc(out[-1, :])
        out = torch.tanh(out)

        return out


model = RNN(all_imaging_feat[0].shape[1], 8, 1, 1)
summary(model, verbose=2)
model.to(device)

learning_rate = 1e-5
```

```
n_epochs = 500
loss_fn = nn.MSELoss()
optimizer = SGD(model.parameters(), lr=learning_rate)


def train():
    model.train()
    train_losses = []
    test_losses = []
    train_acc = []
    test_acc = []

    for t in range(n_epochs):
        optimizer.zero_grad()
        for x in x_train:
            y_pred = model(x)
            loss = loss_fn(y_pred, y_train)
            loss.backward()

        optimizer.step()

        train_losses.append(loss.item())
        train_acc.append((_label(y_pred) == label_train).sum()
           .item() / len(y_train))

        with torch.no_grad():
            test_y_pred = torch.tensor([model(x) for x in
               x_test], device=device)
            test_loss = loss_fn(test_y_pred, y_test)
            test_losses.append(test_loss.item())
            test_acc.append((_label(test_y_pred) == label_test
               ).sum().item() / len(y_test))

        if t % 100 == 99:
            print(f"Epoch {(t + 1)}", loss.item())
            print(f"Loss: {loss.item():.4f}, Acc: {train_acc
               [-1]:.4f}, "
                  f"Test Loss: {test_loss.item():.4f}, Test
                     Acc: {test_acc[-1]:.4f}\n")
```

```
    plot_training_process(
        train_losses, test_losses, train_acc, test_acc,
        label_test.cpu().numpy(), _label(test_y_pred).cpu().
           numpy(), test_y_pred.detach().cpu().numpy(),
        dump_args_to="../result/q_3/q_3_b-training_process-mlp
           .pkl",
        save_img_to="../result/q_3/q_3_b-training_process-mlp.
           png"
    )


def predict():
    with torch.no_grad():
        y = torch.tensor([
            model(torch.tensor(x, dtype=torch.float32, device=
               device))
            for x in all_imaging_feat
        ])

    labels = _label(y).cpu()

    df = pd.DataFrame(labels, columns=["mRS_90d"])
    df["sub_id"] = [f"sub{i:#03d}" for i in range(1, len(
       labels) + 1)]
    df.to_excel("../result/q_3/q_3_b-answer.xlsx", index=False
       )


def load():
    model.load_state_dict(torch.load("../result/q_3/q_3_b-
       model-mlp.pt"))


def save():
    torch.save(model.state_dict(), "../result/q_3/q_3_b-model-
       mlp.pt")
```

```
if __name__ == '__main__':
    Path("../result/q_3").mkdir(exist_ok=True, parents=True)

    train()
    save()

    load()
    predict()
```

A.9 第 3c 问程序

q_3_c.py

```
from pathlib import Path

import numpy as np
import pandas as pd
from scipy.stats import pearsonr

from data import (
    basic_feat, basic_feature_names,
    first_imaging_feat, first_imaging_feature_names,
    mrs_feat
)
from visual import plot_correlation

df = pd.DataFrame(
    np.concatenate((basic_feat[:100], first_imaging_feat
       [:100], mrs_feat.reshape(-1, 1)), axis=1),
    columns=basic_feature_names + first_imaging_feature_names
       + ["mRS_90d"]
)


def calc_correlation() -> pd.DataFrame:
    corr_map = {}
    for feat in df.columns:
        corr_map[feat] = pearsonr(df[feat], df["mRS_90d"])
    df_corr = pd.DataFrame(corr_map).T
    df_corr = df_corr.rename(columns={0: "correlation", 1: "p-
       value"})
```

```
    df_corr = df_corr.reindex(df_corr["correlation"].
       sort_values(ascending=False).index)
    return df_corr.iloc[1:]


if __name__ == "__main__":
    Path("../result/q_3").mkdir(exist_ok=True, parents=True)

    corr = calc_correlation()
    corr.to_excel("../result/q_3/q_3_c-correlation.xlsx")
    plot_correlation(
        corr.index, corr["correlation"],
        title="Correlation between mRS_90d and Patient &
           Imaging Features",
        save_img_to="../result/q_3/q_3_c-correlation.png",
        dump_args_to="../result/q_3/q_3_c-correlation.pkl"
    )
```

A.10 结果可视化

visual.py

```
import pickle

import numpy as np
from seaborn import heatmap
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn.metrics import confusion_matrix, roc_curve,
   roc_auc_score
from sklearn.utils.multiclass import type_of_target

from typedef import *


def plot_training_process(
        train_losses, test_losses, train_acc, test_acc,
        y_true, y_pred, y_score,
        dump_args_to="", save_img_to=""
):
    fig, axes = plt.subplots(2, 2, figsize=(15, 15))
```

```
loss_ax: plt.Axes = axes[0][0]
loss_ax.plot(train_losses, label="train loss")
loss_ax.plot(test_losses, label="test loss")
loss_ax.legend()
loss_ax.set_title("Loss-Epoch")
loss_ax.set_xlabel("Epoch")
loss_ax.set_ylabel("Loss")

acc_ax: plt.Axes = axes[0][1]
acc_ax.plot(train_acc, label="train acc")
acc_ax.plot(test_acc, label="test acc")
acc_ax.legend()
acc_ax.set_title("Acc-Epoch")
acc_ax.set_xlabel("Epoch")
acc_ax.set_ylabel("Acc")

confusion_ax: plt.Axes = axes[1][0]
mat = confusion_matrix(y_true, y_pred)
confusion_ax.matshow(mat, cmap=cm.Blues)
for (i, j), z in np.ndenumerate(mat):
    confusion_ax.text(j, i, f"{z:,}", ha="center", va="
       center")
confusion_ax.set_title("Confusion Matrix")
confusion_ax.set_xlabel("Predicted Label")
confusion_ax.set_ylabel("True Label")


roc_ax: plt.Axes = axes[1][1]
if type_of_target(y_true) == "multiclass":
    roc_ax.text(0.5, 0.5, "ROC Curve is not available for
       multiclass classification", ha="center", va="center"
       )
else:
    roc_ax.plot(*roc_curve(y_true, y_score)[:2])
    roc_ax.text(0.5, 0.5, f"AUC: {round(roc_auc_score(
       y_true, y_score), 3)}", ha="center", va="bottom")
roc_ax.set_title("ROC Curve")
roc_ax.set_xlabel("False Positive Rate")
```

```
    roc_ax.set_ylabel("True Positive Rate")

    if dump_args_to:
        pickle.dump({
            "train_losses": train_losses, "test_losses":
               test_losses,
            "train_acc": train_acc, "test_acc": test_acc,
            "y_true": y_true, "y_pred": y_pred, "y_score":
               y_score
        }, open(dump_args_to, "wb")
        )

    if save_img_to:
        fig.savefig(save_img_to)

    plt.show()


def plot_fitting_curve(
        xs, ys, funcs,
        title="", x_label="", y_label="", text="",
        dump_args_to="", save_img_to=""
):
    x_max = max([x.max() for x in xs])
    y_max = max([y.max() for y in ys])
    x_pred = np.linspace(0, x_max, 10000)

    for x, y, func in zip(xs, ys, funcs):
        y_pred = func(x_pred)
        scatter = plt.scatter(x, y, s=10, alpha=0.4)
        if len(funcs) == 1:
            color = "r"
        else:
            color = scatter.get_edgecolor()
            color[:, -1] = 1
        plt.plot(x_pred, y_pred, color=color)

    plt.text(x_max, y_max, text, ha="right", va="top")
    plt.title(title)
```

```
    plt.xlabel(x_label)
    plt.ylabel(y_label)

    if dump_args_to:
        pickle.dump({
            "xs": xs, "ys": ys, "funcs": funcs, "title": title
               ,
            "x_label": x_label, "y_label": y_label, "text":
               text
        }, open(dump_args_to, "wb")
        )

    if save_img_to:
        plt.savefig(save_img_to)

    plt.show()


def plot_correlation(
        feat, corr,
        title="",
        dump_args_to="", save_img_to=""
):
    plt.figure(figsize=(20, 16))
    plt.barh(feat, corr)
    plt.xlabel('Feature')
    plt.ylabel('Correlation')
    plt.title(title)
    plt.xlim(-0.4, 0.4)
    plt.xticks(fontsize=5)

    if dump_args_to:
        pickle.dump({
            "corr": corr, "title": title
        }, open(dump_args_to, "wb")
        )

    if save_img_to:
        plt.savefig(save_img_to)
```

```
    plt.show()


def plot_heatmap(
        corr, title="",
        dump_args_to="", save_img_to=""
):
    plt.figure(figsize=(20, 16))
    plt.title(title)
    heatmap(corr, cmap=cm.Blues)

    if dump_args_to:
        pickle.dump({
            "corr": corr, "title": title
        }, open(dump_args_to, "wb")
        )

    if save_img_to:
        plt.savefig(save_img_to)

    plt.show()
```

data_analysis.py

```
from pathlib import Path

import pandas as pd

from data import basic_feat, first_imaging_feat
from visual import plot_heatmap

basic_feat = basic_feat[:100]
first_imaging_feat = first_imaging_feat[:100]


def analyze(feat, feat_name):
    df = pd.DataFrame(feat)
    corr = df.corr()
```

```
    plot_heatmap(
        corr,
        title=f"Heatmap of {feat_name} Features",
        dump_args_to=f"../result/data/heatmap-{feat_name.lower
            ()}.pkl",
        save_img_to=f"../result/data/heatmap-{feat_name.lower
            ()}.png"
    )


if __name__ == "__main__":
    Path("../result/data").mkdir(exist_ok=True, parents=True)

    analyze(basic_feat, "Basic")
    analyze(first_imaging_feat, "Imaging")
```