问题一

血肿扩张风险相关因素探索建模

思路：

根据题目要求,首先需要判断每个患者是否发生了血肿扩张事件。根据定义,如果后续检查的血肿体积比首次检查增加≥6 mL 或≥33%,则判断为发生了血肿扩张。
具体判断步骤:
(1) 从表 1 中提取每个患者的入院首次影像检查流水号;

(2) 根据流水号在附表 1 中查找对应首次检查的时间点;

(3) 计算发病到首次检查的时间间隔;

(4) 在表 2 中找到每个随访时间点的血肿体积;

(5) 依次计算相邻两次检查血肿体积变化量和变化百分比;

(6) 如果变化量≥6 mL 或变化百分比≥33%,则记为发生血肿扩张,记录下血肿扩张发生的时间点。

3.使用 logistic 回归建模,以是否发生血肿扩张作为目标变量,个人史、疾病史和首次影像特征作为自变量,建立预测模型。

目标变量: Y = 是否发生血肿扩张(1 是,0 否)

自变量: X1, X2, …, Xn(个人史、疾病史等)

建模公式: P(Y=1|X) = 1 / (1+e^-(b0+b1X1+…+bnXn))

4.使用训练集拟合 logistic 回归模型

(1) 将训练集的个人史、疾病史和首次影像特征整理为自变量 X

(2) 将训练集的血肿扩张标记(1 或 0)作为目标变量 Y

(3) 将自变量 X 和目标变量 Y 喂入 logistic 回归模型进行拟合

(4) 使用最大似然估计获得变量系数 b0, b1, …, bn

(5) 得到拟合后的模型:

P(Y=1|X) = 1 / (1+e^-(b0+b1X1+…+bnXn))

5.用拟合好的模型对测试集进行预测

(1) 对测试集数据进行同样的特征工程,提取自变量 X

(2) 将测试集的自变量 X 代入上面得到的模型中

(3) 计算每个样本的血肿扩张概率 P(Y=1|X)

(4) 如果 P(Y=1|X) ≥ 0.5,则预测该样本发生了血肿扩张

(5) 计算模型在测试集上的评估指标,如 AUC 等

(6) 根据变量系数的大小分析变量与血肿扩张的相关性

```
import pandas as pd

from sklearn.linear_model import LogisticRegression



# 读取表1 和表2 中的数据

table1 = pd.read_excel('表1.xlsx')

table2 = pd.read_excel('表2.xlsx')



# 将表1 和表2 进行合并

data = pd.merge(table1, table2, on='ID')



# 提取需要的特征

features = ['age', 'gender', 'history', ...]
```

```
# 获得每个患者的首次影像时间和血肿体积

first_scan = data.groupby('ID')['time'].min()

first_volume = data[data['time'] == first_scan]['HM_volume']
```

代码主要步骤包括:

读取和合并表格
特征工程
标记目标变量
划分训练集和测试集
模型训练和预测
输出结果
此处我们使用 xgboost 训练模型：

主要步骤为:

导入 xgboost
设置 xgboost 的参数:
eta:学习率
max_depth:树的最大深度
objective:二分类的逻辑回归
eval_metric:评估指标设为 AUC
将训练数据转换为 DMatrix 格式
使用 xgboost 训练模型
将测试数据也转为 DMatrix 格式
用训练好的模型进行预测
输出结果
XGBoost 是一个流行且高效的树模型库,可以提取数据的复杂特征关系。

相比逻辑回归,XGBoost 可处理各种类型的特征,也便于调参优化模型。

问题二

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。

构建水肿体积随时间变化的模型
可以使用 Curve Fitting 的方法,以时间为自变量,水肿体积为目标变量,拟合出水肿体积随时间的曲线模型:

$$VED = f(t)$$

其中,VED 表示水肿体积,t 表示时间。

可以试用不同的曲线拟合方法,如线性回归、多项式回归、局部加权回归等。

计算患者真实值与拟合曲线的残差
对第 i 个样本:

$$ri = VEDi - f(ti)$$

其中,$VEDi$ 为第 i 个样本的真实水肿体积,$f(ti)$为对应时间点上的拟合值。

划分患者亚组,拟合各亚组的水肿体积曲线
可以使用聚类算法如 K-means 对患者进行分群,然后对每一群体单独拟合曲线。

分析不同治疗对水肿演变的影响
可以将治疗方法作为类别特征,构建不同的曲线模型,然后比较模型效果。

也可以通过统计学方法(如 t 检验)比较不同治疗组水肿体积变化的差异。

分析三者之间的关系
可以采用相关性分析等统计学方法探索血肿体积、水肿体积和治疗之间的关系。

也可以构建包含三者作为特征的预测模型,通过分析系数等来发现三者之间的关联。
具体来说，相关性分析法
(1) 计算每个样本的血肿体积、水肿体积和各种治疗方式的 0/1 表示

(2) 使用 Pearson 相关系数计算血肿体积和水肿体积的线性相关性

(3) 使用 Spearman 秩相关系数计算血肿体积与各治疗方法的秩相关性

(4) 使用 Spearman 秩相关系数计算水肿体积与各治疗方法的秩相关性

(5) 比较不同系数的大小,分析三者之间的相关程度

建模法

(1) 将血肿体积、水肿体积作为连续特征,治疗方法作为分类特征

(2) 构建回归模型,以水肿体积为目标变量,血肿体积和治疗作为自变量

(3) 训练模型,得到各变量的系数

(4) 比较各治疗类别的系数,看其对水肿体积的影响效果

(5) 通过变量的显著性检验,选择关键的影响因素

(6) 分析模型总体表现,评估各变量的解释能力

```
import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.cluster import KMeans



# 读取数据

data = pd.read_excel('table2.xlsx')



# 特征工程:提取时间和水肿体积

X = data[['time']]

y = data[['ED_volume']]
```

```
# 构建线性回归模型
lr = LinearRegression()

# 训练模型
lr.fit(X, y)

# 获取拟合的系数
print('模型 Slope:', lr.coef_)
print('模型 Intercept:', lr.intercept_)

# 预测水肿体积
y_pred = lr.predict(X)
```

问题三

出血性脑卒中患者预后预测及关键因素探索
1.基于首次影像结果预测预后
使用回归模型,以 90 天 mRS 评分为目标变量,个人史、疾病史和首次影像特征为自变量:

$$mRS=w_0+w_1x_1+...+w_nx_n$$

其中,mRS 为预后评分,x_i 为各特征, w_i 为对应的权重系数。

可以试用线性回归、LASSO 回归等算法。

2.基于全部影像结果预测预后

同上,不仅使用首次影像,还结合后续各时间点的影像特征,构建回归模型进行预测。

3.分析关键影响因素

通过分析各变量的权重w_i,确定对mRS影响最大的特征。

使用统计检验分析不同特征对mRS的显著影响。

采用则特征选择的方法(如 RFE),选择关键特征。

将无关特征删除后,观察模型评分的变化。

具体来说,

1）建模算法的选择

可以尝试线性回归、LASSO 回归、GBDT 等多种算法

比较不同算法的误差、过拟合情况,选择较优算法

调参优化模型,提升准确率

2）特征工程

处理缺失值:删除/填充

编码类别特征:One-hot 编码

标准化连续特征:去均值和方差归一化

提取时间序列特征:趋势、周期性等

采用 PCA 等方法降维

3）模型评估

划分训练集、验证集、测试集

多次交叉验证,观察方差

计算 RMSE、R2、MAE 等评价指标

绘制学习曲线,检查过拟合问题

4）关键因素分析

计算特征影响力,排序筛选

通过添加/删除特征,比较模型效果变化

使用统计学检验(t-test 等)判断显著性

采用正则化方法自动特征筛选

分析特征在不同亚群中的效果

4.提出建议

对具有显著影响的特征,分析临床意义,给出干预建议。

对预后良好和预后不良的患者组,进行对比分析,找出影响因素的差异。
代码:

```
# 导入需要的库

import pandas as pd

from sklearn.linear_model import Lasso

from sklearn.model_selection import cross_val_score

import matplotlib.pyplot as plt



# 读取数据

data = pd.read_csv('data.csv')



# 特征工程

X = data[['age', 'gender', 'treatment', 'image_features']]

y = data['mRS']



# 拆分数据集

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=2020)



# Lasso 回归

model = Lasso()
```

```
# 使用网格搜索找到最优参数

from sklearn.model_selection import GridSearchCV

params = {'alpha': [0.001, 0.01, 0.1, 1]}

gs = GridSearchCV(model, params,
scoring='neg_mean_squared_error', cv=5)

gs.fit(X_train, y_train)

print('最优参数:', gs.best_params_)

model = gs.best_estimator_#见完整版
```