基于 LightGBM 模型的糖尿病风险预测

摘要

本文分析糖尿病的风险预测问题，通过**逐步回归模型**从 42 项变量中筛选出主要变量并进行合理性分析，然后基于主要变量建立了 **lightGBM 模型**实现血糖的预测，采用**基于熵权法改进的 topsis 模型**建立糖尿病风险评估体系，最终对无血糖值的数据进行了较为准确的预测和较合理的风险评估。

针对**问题**一，对于患者的 42 个检测数据指标，本文建立**逐步回归模型**实现了从中筛选出主要变量的目标。首先对原始数据进行**删除**、**填补**、**平滑**、**组合**等操作。然后利用逐步回归模型筛选出主要变量。使用 **lightGBM 特征重要性选择法**对筛选出来的主要变量进行特征重要性分析，进一步验证主要变量的准确性和合理性。结果表明，**甘油三脂**、**碱性磷酸酶**、**年龄**、**丙氨酸氨基转化酶**、**尿素等九种变量对血糖值的影响作用显著**。

针对**问题**二，利用问题一筛选出的主要变量，建立 **lightGBM 模型**对血糖值进行预测。首先对主要变量的数据进行标准化处理，然后通过最小化均方根误差对 lightGBM 模型进行训练，最后将 lightBGM 模型与**逐步回归**、**随机森林**、**SVR**、**XGBoost** 等模型进行算法对比。实验结果表明，**lightGBM 模型在 MAE**、**MAPE**、**CVRMSE 等评价指标上均优于其他模型**。

针对**问题**三，本文采用**基于熵权法改进的 topsis 模型**对糖尿病的患病风险进行评价。首先将问题一筛选出的主要变量作为模型的评价指标，采用**熵权法**为评价指标进行**客观赋权**，最后利用 **topsis 模型**对糖尿病的患病风险进行评估。

针对**问题四**，本文对问题一筛选出的主要变量进行**数据预处理**，利用问题二建立的 **lightBGM 预测模型**和问题三建立的**基于熵权法改进的 topsis 评价模型**进行血糖值预测与糖尿病患病风险进行评估。结果表明，**附件二中 17 人有较高的患病风险**。

关键词：　**糖尿病风险评估**　**逐步回归模型**　**LightGBM 模型**　**熵权法**　**topsis 模型**

目录

1 问题重述

1.1 问题背景

近 30 多年来，随着经济的发展和人们生活方式的改变，人们更多高脂高糖的饮食以及运动量的减少，导致了更多的人超重，我国糖尿病患病率逐年显著增加。糖尿病是一种由遗传和环境因素引起的代谢紊乱，导致胰岛素不敏感、胰岛素缺乏和生物学功能受损。由于该病的高流行率以及相关的残疾和死亡率，它已成为世界范围内一个严重的健康问题。据估计，中国有 1.139 亿成年人糖尿病患者，占全球糖尿病患者的 24%。糖尿病作为一种常见慢性疾病，目前无法根治，需要通过科学有效的干预、预防和治疗，来降低发病率和提高患者的生活质量。基于这样的大环境，对糖尿病的相关分析就变得极为重要，本文是通过分析在医院检查过程中的哪些检测数据变量对糖尿病有较大的影响，并以此进行预测患者的血糖值。

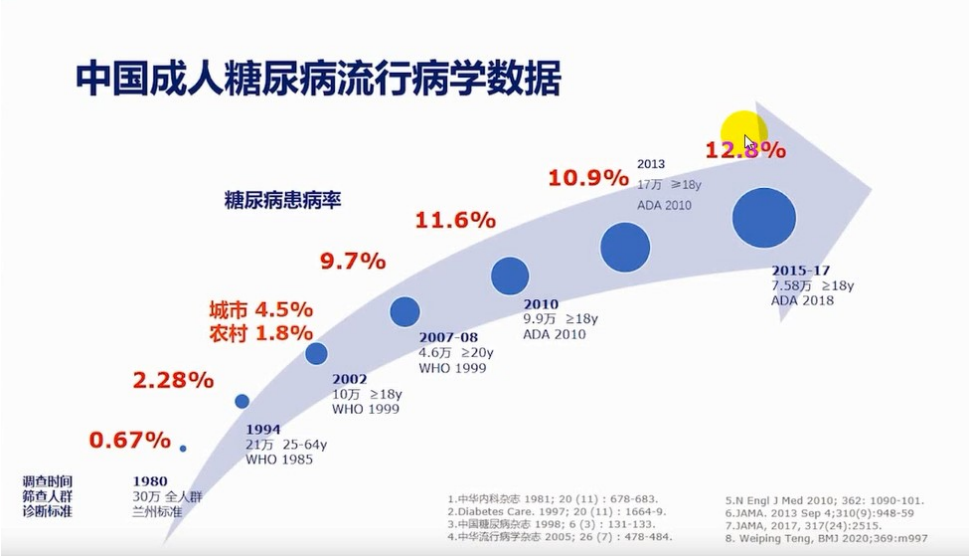


图 1　糖尿病背景图

1.2 问题提出

基于对病人包含年龄、性别、各项体检数据等 42 个监测指标（包含数值型、字符型、日期型等数据类型），附件中的各项数据存在不少缺失值，请数据填补后并建立数学模型来解决以下的问题:

问题 1：利用附件 1 的检测数据，从其中 42 个检测指标中筛选出来主要的变量指标，同时详细解释主要变量的筛选过程及其合理性。

问题 2：利用附件 1 的检测数据，并结合问题 1 筛选出来的主要变量来建立血糖值的预测模型。

问题 3：利用附件 1 的检测数据，根据体检数据对糖尿病的风险进行评估，判断糖尿病具体会导致哪些值发生异常。

问题 4：利用附件 2 无血糖的检测数据和问题 2 建立的模型，对其血糖值进行预测，并对糖尿病风险进行评估。

2 模型的假设

- 假设每个人的血糖测量值和预测值均为空腹血糖值.
- 假设血糖值的标准化按照 2023 年世界卫生组织 (WHO) 标准来判断的
- 各个变量指标之间不存在多重共线性

3 符号说明

符号	含义
y	血糖值
x_i	患者各项数据变量
β_i	病种医保支付定额
μ	医院成本占比
F	F 检验统计量
i	变量指标的序号
j	每项变量指标数据的序号
μ	该项变量指标下数据的均值
σ_i	表示该变量指标的标准差
X_{ij}	归一化处理后得到的新数据
X'_{ij}	标准化处理后得到的新数据
y_i	第 i 条样本的真实值
$\hat{y}_l$	第 i 条样本的预测值
$\bar{y}$	样本真实值的平均值
e_j	熵值
w_j	熵权值
h_{ij}	所占比例
C_i	贴进度

4 问题一模型的建立与求解

4.1 问题一的分析

为了能够从 42 个检测指标中筛选出主要的变量指标，首先先对附件 1 给我们的各项数据进行数据清理，接下来开始筛选主要变量指标，由于逐步回归具有计算量小，回归结果精确的优点，我们采用逐步回归的方法对各变量进行筛选，判断各变量对血糖值的影响大小，得到了最终的变量指标。最后使用 lightGBM 重要特征检验来进行验证，判断模型的合理性和准确性。

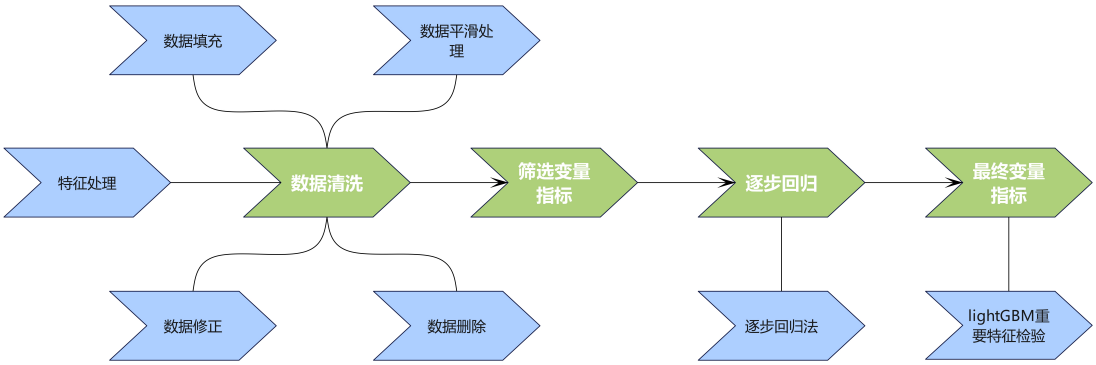


图 2　问题一的分析

4.2 数据清洗

对于附件一给我们的数据进行分析，发现数据中有着大量的缺失值，无法直接用来分析，所以采用数据清洗的操作，包括数据删除、填充、平滑、特征处理。

4.2.1 数据删除

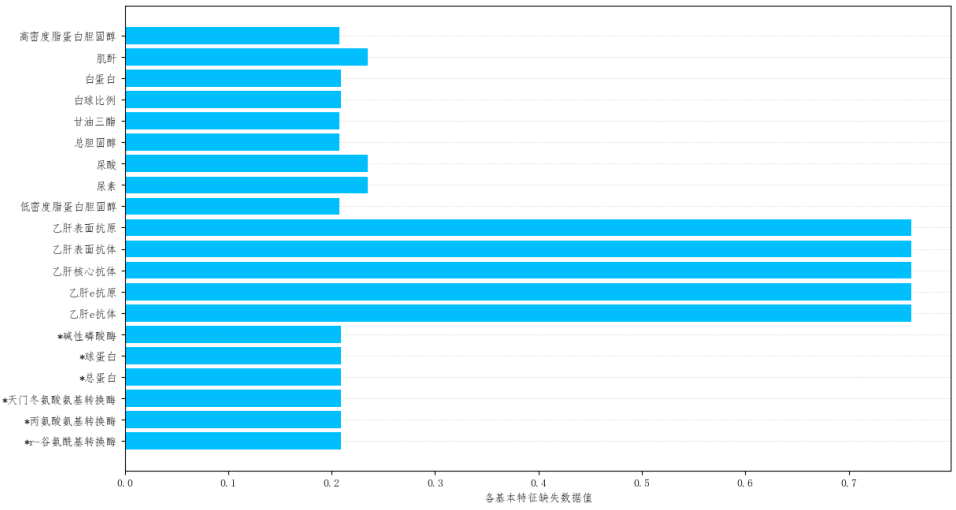


图 3　各基本特征缺失数据值图

由图可知：乙肝表面抗原、乙肝表面抗体等五个检测指标数据缺失项具有非常高的占比，约为 0.75；蛋白胆固醇、肌酐等 15 个检测指标数据缺失项占比为 0.2。

去除高缺失比例数据。去除缺失比例非常高的基本特征，是指删除缺失比例超过 70% 的数据，由上图中可知乙肝表面抗原、乙肝表面抗体、乙肝核心抗体、乙肝 e 抗原、乙肝 e 抗体这五个基本特征缺失比例超过 70%，且这五种基本特征对于预测模型的影响权重较小，没有合理的办法对这些缺失的数据进行填充，因此选择将这些缺失比例很高的基本特征剔除. 最终留下了 37 个检测指标

去除连续大量缺失数据我们在约 900 人的所有检测指标中发现了蛋白胆固醇、肌酐等十五个检测指标都出现了连续大量数据缺失的情况，这种情况下对于数据进行插值，会对结果产生较大的影响。所以也对其进行删除。最终留下了共 5000 条数据

4.2.2 数据填充

由于附件 1 中的数据存在着较多的缺失，我们使用了 PCHIP 插值法、三次样条插值法以及移动均值法对进行去除后的数据进行插值，发现三次样条插值数据误差太大，会对后续结果产生巨大的误差；移动均值法在插值时插值数据变化很小，与前后相差较大，故我们采用 PCHIP 插值法对缺失数据较少的变量进行了插值，对数据进行合理的填补，使其具有较高的稳定性和可信度。

4.2.3 数据的平滑处理

为了进一步减少随机误差，提高插值数据的可靠性和准确性，并增强数据的可视性，我们对数据进行了平滑处理。我们运用 Savitzky-Golay filtering 对于插值后的数据进行平滑处理。首先将 37 个检测指标作为自变量 $x_1 \cdots\cdots x_{37}$, 对于列向量的每一个点，提取一个以该点为中心，大小等于奇数 “framelen” 的子向量。然后对于这个子向量，计算多项式曲线的最小二乘拟合，定义为 $p(x)$=a_0+a_1*x+a_2*x^2+$\cdots\cdots$请注意，x 在中心点等于零。最后用适合值计算的值替换初始中心点，得到平滑处理之后的数据。

4.2.4 数据删除

考虑到变量指标均为医学指标，我们结合医学上对所有预处理后的变量指标进行分组处理。将红细胞计数、红细胞总血红蛋白量、红细胞平均血红蛋白量等 10 个变量进行合并处理[1]，将肾、尿酸、尿素、肌酐进行合并处理，将嗜酸细胞、白细胞计数、嗜酸细胞、进行合并处理，其余变量指标都单独进行分组。

4.3 逐步回归模型的建立和求解

4.3.1 逐步回归模型的建立

相较于多元线性回归分析，逐步回归具备更合理的自变量筛选机制，利用该模型在数据筛选方面的优势，我们选择逐步回归法来解决第一问。逐步回归有三种方法：前进法，后退法和逐步回归法[2]，同时，对于前进法不能反映自变量选进模型后的模型本身的变化情况；后退法开始便把全部自变量引入模型，计算量过大，我们使用逐步回归法

恰好可以弥补前进法和后退法各自的缺陷，结合向前法和向后法的优点，在向前引入每一个新自变量之后都要重新对已代入的自变量进行计算，以检验其有无继续保留在方程中的价值，并以此为依据进行自变量的引入和剔除交替进行，直到没有新的变量可以引入或剔除为止，此方法可以较为准确得到最优解。[3]

我们使用逐步回归法选择自变量。首先我们定义筛选过后的 35 个指标为 $x_1 \cdots\cdots x_{35}$, 依次将变量一个一个地引入，对于已选入的变量进行逐个检验，当原引入的变量由于后面变量的引入而使得对于血糖值的变化变得不再显著时，就将其剔除。而每引入一个变量或者时剔除一个变量，这就是逐步回归的一步，且每一步都要进行 F 检验，保证每次引入的新变量之前回归方程只包含对于血糖值有显著变化的变量。

此过程实施的流程图如下：

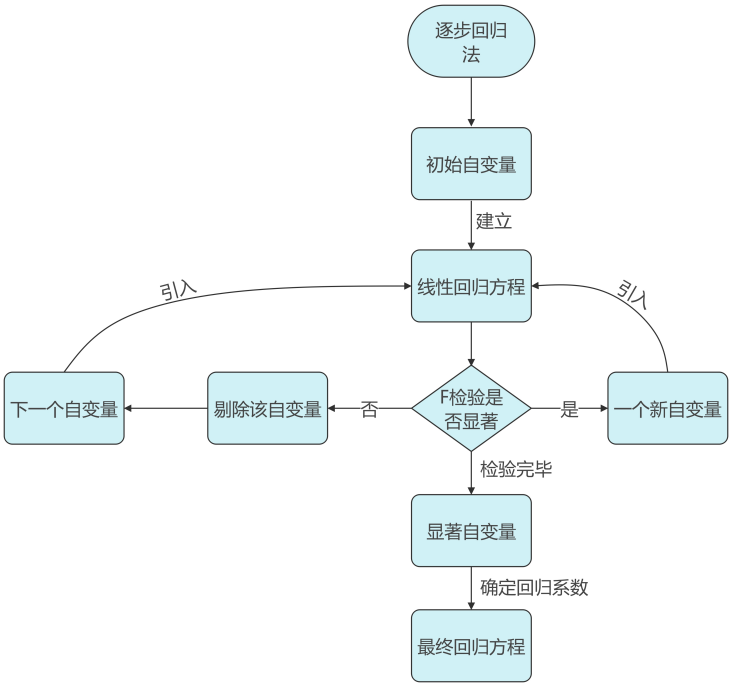


图 4　逐步回归流程图

首先我们用 x_1 来建立线性回归方程，y_1=ax_1，然后进行 F 检验，其公式为：

$$\mathrm{F} = \mathrm{S}_1^2/\mathrm{S}_2^2 \tag{1}$$

其中，F 为 F 检验的统计量，根据自由度查表，当 F 值小于查表值时，没有显著差异，当 F 值大于查表值时，有显著差异。S_1 为 y 的标准差，S_2 为 x_1 的标准差。

若 x_1 的检验指标显著，则保留 x_1；以此类推，我们用对于 x_{35} 采用相同方式处理，最后经过检验显著性分析留下了 8 个个较为显著的 x_n，并得到了线性回归方程:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2}$$

最终筛选出来的变量有 8 组，所以 n 为 8，β_i 为各个变量对血糖值的影响程度系数。

4.3.2 逐步回归模型的求解

通过对数据清洗过的变量进行逐步回归分析，我们得到了各变量对血糖值相关性数值图，结果如下图：

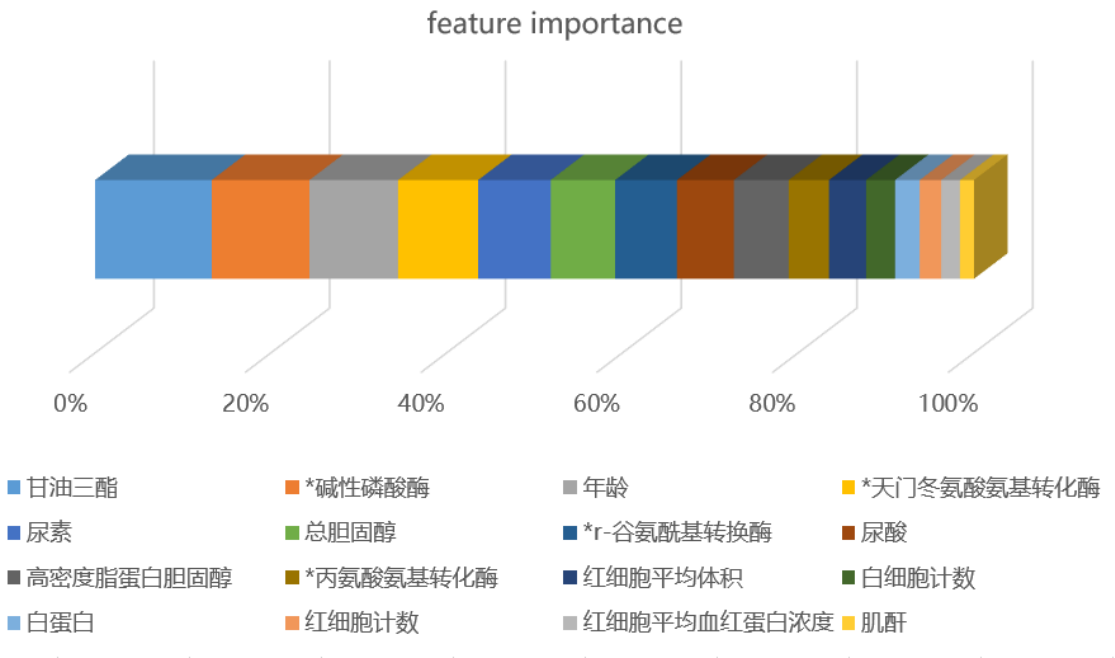
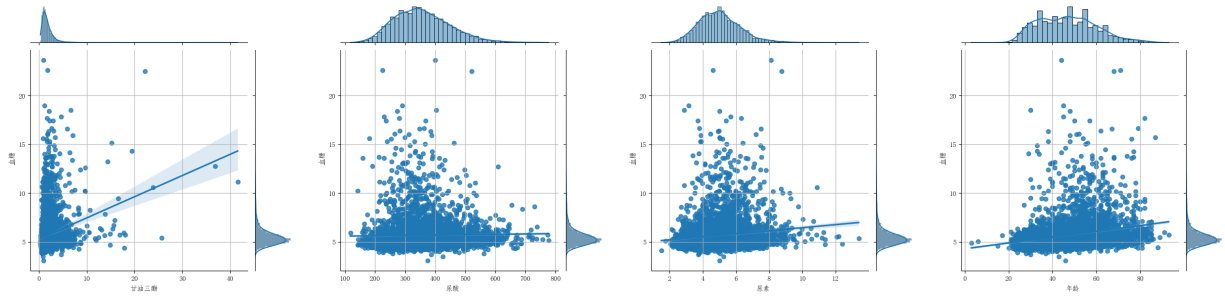


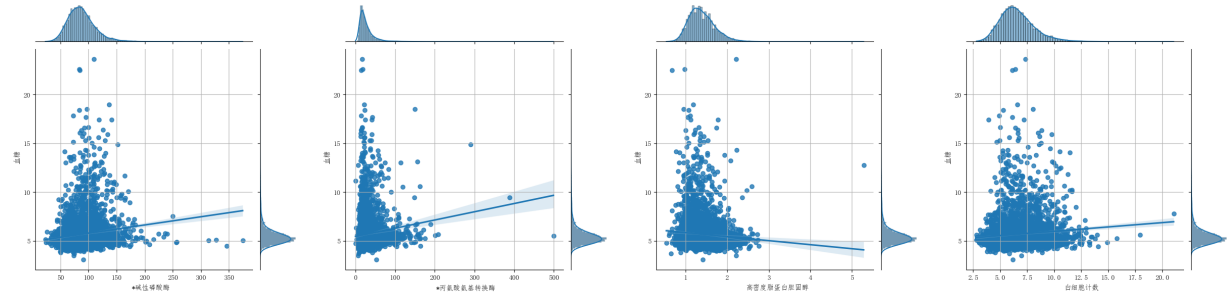
图 5　重要性指标图

上图说明了甘油三酯、碱性磷酸酶、年龄、丙氨酸氨基转化酶等八个自变量对于血糖的的变化具体有很显著的作用，而、血小板体积分布宽度、中性粒细胞等七个自变量对于血糖的变化的显著性不是特别明显。因此我们选择甘油三酯、碱性磷酸酶、年龄、丙氨酸氨基转化酶、尿素、总胆固醇、r-谷氨酰基转换酶、尿酸这八个变量指标作为主要变量指标。

针对这八种变量，我们分别做出了他们各自数据与血糖的关系图以及拟合曲线，如下：



(a) 甘油三脂　(b) 尿酸　(c) 尿素　(d) 年龄



(e) 碱性磷酸酶　(f) 丙氨酸氨基转换酶　(g) 高密度脂蛋白胆固醇　(h) 白细胞计数

图 6　主要特征变量指标与血糖关系图

从图中我们可以看出这八个主要变量均与血糖值符合正态分布的关系，可以进一步验证模型的合理性。

4.4 基于 LightGBM 特征重要性选择算法

4.4.1 lightGBM 特征重要性选择的求解

统计所有被清理过的患者各项数据，采用了 lightGBM 特征重要性选择方法来对各变量的特征重要性进行分析，分析结果如下图所示：

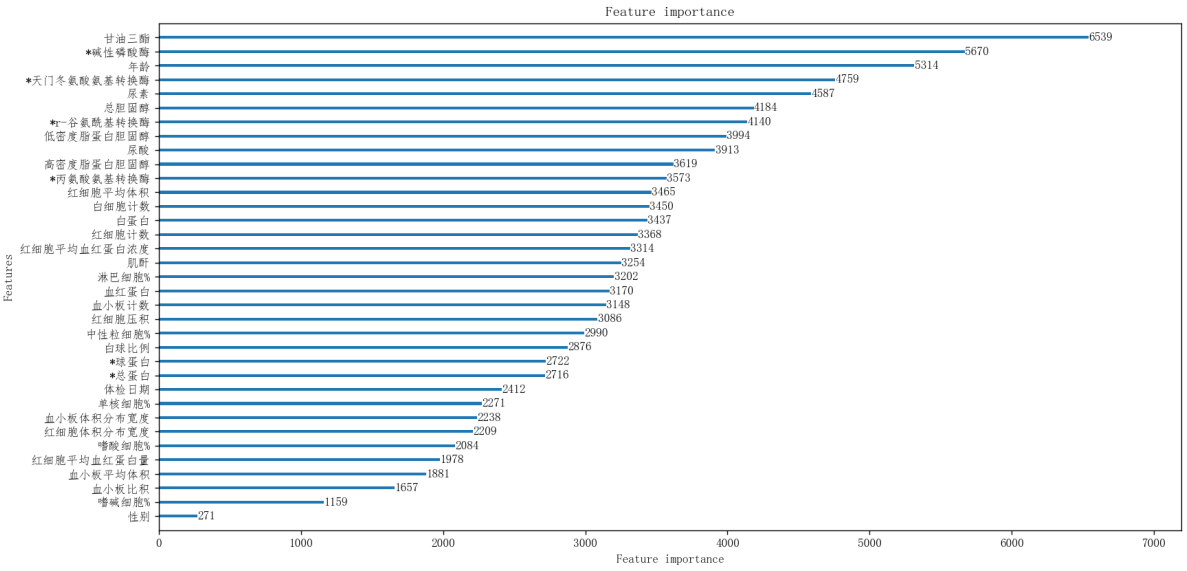


图 7　各变量 lightGBM 特征重要性图

由图可知，在附件 1 给出的患者的各项数据中，与血糖值特征重要性最大的为甘油三酯、然后依次为：碱性磷酸酶、年龄、丙氨酸氨基转化酶、尿素、总胆固醇、r-谷氨酰基转换酶、低密度脂蛋白胆固醇、尿酸等，得到的结论与使用逐次回归得到的结论高度吻合，只有低密度脂蛋白胆固醇一个变量不一致，可以再次验证我们得到的主要变量指标的合理性。

为了验证我们筛选变量的合理性，我们将总体年龄划分为三个不同的年龄段（0-18岁为青年，19-45 岁为中年，46-100 岁为老年），分别为青年、中年、老年，并分析这三个年龄段的平均血糖值情况，得到结果如下图：

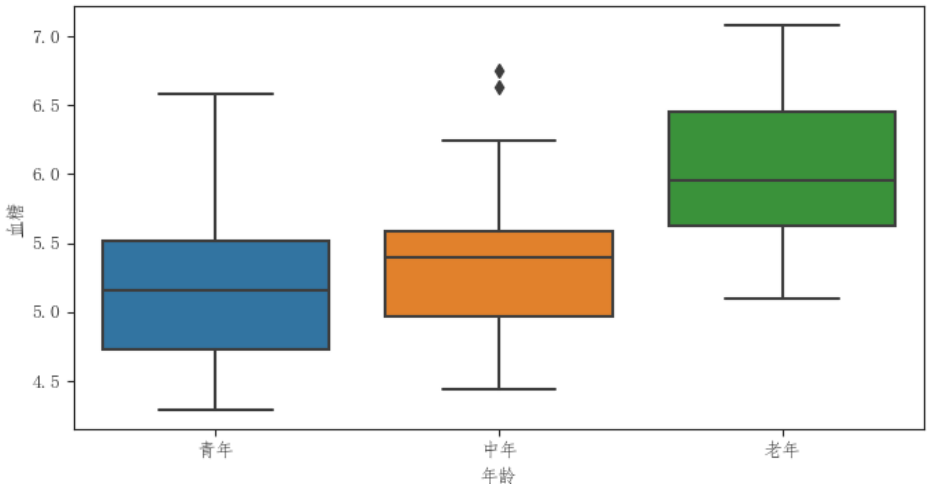


图 8　不同年龄段对血糖的影响图

可以看出，三种不同年龄段的人群与血糖值的大小有较为明显相关关系，青年的血糖值最低，老年时期的血糖值会达到较高水平，进一步验证了年龄对血糖值有较大影响，验证了我们对只要特征变量的筛选是合理的。

4.5 结果分析

通过逐步回归模型的分析和 lightGBM 特征重要性算法的检验，我们得出了在附件 1 给出的患者的各项数据中，与血糖值特征重要性最大的变量为甘油三酯、然后依次为：碱性磷酸酶、年龄、丙氨酸氨基转化酶、尿素、总胆固醇、r-谷氨酰基转换酶、低密度脂蛋白胆固醇、尿酸这九种变量，因此我们将这九种变量作为我们最终筛选出来的主要变量指标。通过查阅医学刊物，我们验证了甘油三酯、碱性磷酸酶、年龄等确实与血糖值有着较大的影响，进一步验证我们得到的主要变量的合理性。

5 问题二模型的建立与求解

5.1 问题二的分析

问题一中我们分析筛选出了与血糖值有关的主要变量，所以在这一问对血糖值进行预测也主要靠这 9 个主要变量，在预测之前，首先进行数据预处理，即对数据进行归一化处理和标准化处理，然后我们运用多种模型对血糖值进行预测，包括随机森林、SVR、XGBoost、LightGBM 四种模型进行预测，其中 lightGBM 为主模型。最后比较各个模型预测值与真实值的误差，找到误差最小的模型，能够对血糖值做出较为准确的预测。

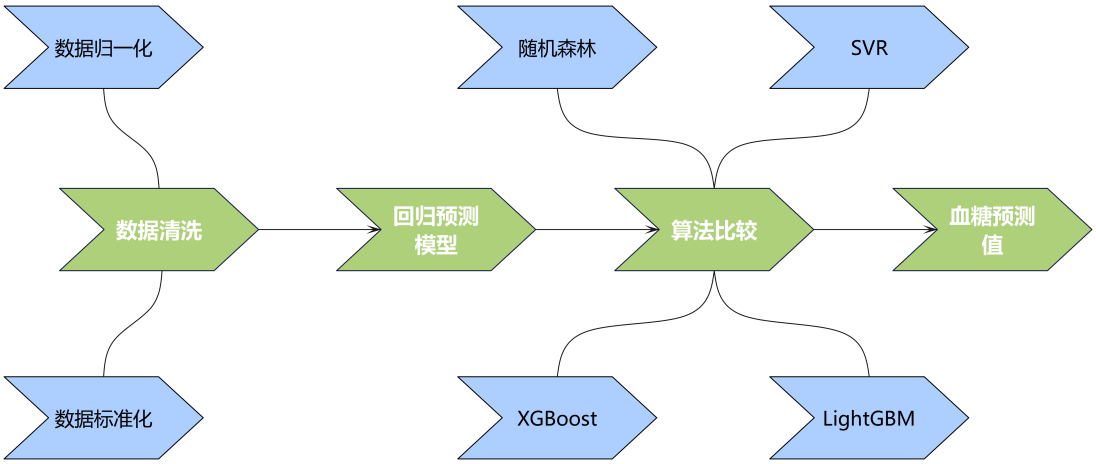


图 9　问题二的分析

5.2 数据的标准化

将第一问求出的 9 个检测指标分别进行 Z-score 标准化处理，有：

$$X'_{ij} = \frac{(X_{ij} - \mu_i)}{\sigma_i} \quad (i = 1, 2, \cdots n) \quad j = (1, 2, \cdots \cdot .) \tag{3}$$

这里 i 是变量指标的序号，j 是每项变量指标数据的序号，X'_{ij} 表示标准化处理后得到的新数据,X_{ij} 表示对该变量指标的归一化数据,μ_i 表示在该项变量指标下数据的均值,σ_i 表示该变量指标的标准差。经过这一步我们达到了去量纲的目的。

5.3 LightGBM 模型的建立

LightGBM 算法

LightGBM 是 GBDT 模型的一个进化版。LightGBM 是 boosting 集合模型中的新进成员，由微软提供，它和 XGBoost 一样是对 GBDT 的高效实现，原理上它和 GBDT 及 XGBoost 类似，都采用损失函数的负梯度作为当前决策树的残差近似值，去拟合新的决策树.[4]

lightBGM 主要有以下特点：

- 基于 Histogram 的决策树算法

- 带深度限制的 Leaf-wise 的叶子生长策略
- 直方图做差加速
- 直接支持类别特征 (Categorical Feature)
- Cache 命中率优化
- 基于直方图的稀疏特征优化
- 多线程优化[5]

对于训练集有 n 个实体 x_1,x_2,x_3,$\cdots$,x_n，特征维度为 s，每次迭代梯度时，模型中数据变量的损失函数的梯度记录为 g_1,g_2,$\cdots$,g_n，按照最大信息增益，决策树将数据分配至各个节点中，最后通过划分后的方差来测定信息增益。假设 O 表示某个分裂节点的训练集，则分割变量 j 的分割点如下式所示：

$$V_{j|O}(d)=\frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i\in O:x_{ij}\leqslant d\}}g_i\right)^2}{n^j_{l|O}(d)}+\frac{\left(\sum_{\{x_i\in O:x_{ij}>d\}}g_i\right)^2}{n^j_{r|O}(d)}\right) \tag{4}$$

$$n_o=\sum I\left[x_i\in O\right],n^j_{||o}=\sum I\left[x_i\in O:x_i\geq d\right],n^j_{r|o}=\sum I\left[x_i\in O:x_i>d\right] \tag{5}$$

为找到最大的信息增益，遍历每个特征的分割点，并将数据划分为左右节点。以数据集 A 为实例，计算其方差并按照同质数据的梯度降序对训练进行排序，对剩余数据进行随机抽样，生成大小为 b 的数据集 B。该算法的主要步骤如下:

步骤一：模型初始化为常数。

步骤二：使用目标函数对数结构进行打分。

步骤三：枚举不同树的结构，使得目标函数最小即为最优结构的树，加到模型中。

步骤四：重复步骤二和步骤三，直到回归树的数量达到指定的迭代次数。

步骤五：得到最终组合的学习器，即 LightGBM 血糖值预测模型。

步骤六：对多模型的误差进行分析。

采用的误差分析指标为平均绝对误差 MAE，平均绝对百分比误差 MAPE 和百分比均方根误差 CVRMSE。MAE 可以避免计算误差时相互抵消的问题，能够准确反映实际预测误差的大小；MAPE 常用于衡量预测的准确性，越接近 0% 则预测误差越小；CVRMSE 在均方根误差的基础上取了百分比，能够更好的表征测量值和预测值的拟合程度，其值越小，预测精度越高, 其中：

平均绝对误差计算公式为：

$$MAE=\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i-y_i| \tag{6}$$

平均绝对百分比误差计算公式为：

$$MAPE = \frac{100\%}{n}\sum_{i-1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{7}$$

均方根误差计算公式为:

$$CV(RMSE) = \frac{1}{\bar{y}}\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{8}$$

在式 (7) 到式 (9) 中，n 为数据样本量，y_i 为第 i 条样本的真实值，$\hat{y}_l$ 为第 i 条样本的预测值，$\bar{y}$ 为样本真实值的平均值。

5.4 lightGBM 模型的求解

利用 LightGBM 算法分类. 将数据集按照 8：1：1 划分训练集、验证集和测试集
数据集的划分如下表所示:

表 1　数据集的划分

数据集	样本数
训练集	4065
测试集	478
验证集	457

进行 5 折交叉验证. 设置超参数网格进行网格搜索，确定使得测试集分类准确率最高的超参数.

表 2　模型预测各误差对比

num-leaves	n-estimators	learning-rate	准确率
5	9	0.05	0.615
5	10	0.05	0.605
6	9	0.05	0.612
6	10	0.05	0.631

最佳超参数为：larning-rate=0.05，n-estimators=10，num-leaves=6. 测试集准确率为 0.630.，测试集准确率为 0.632，混淆矩阵如下图：

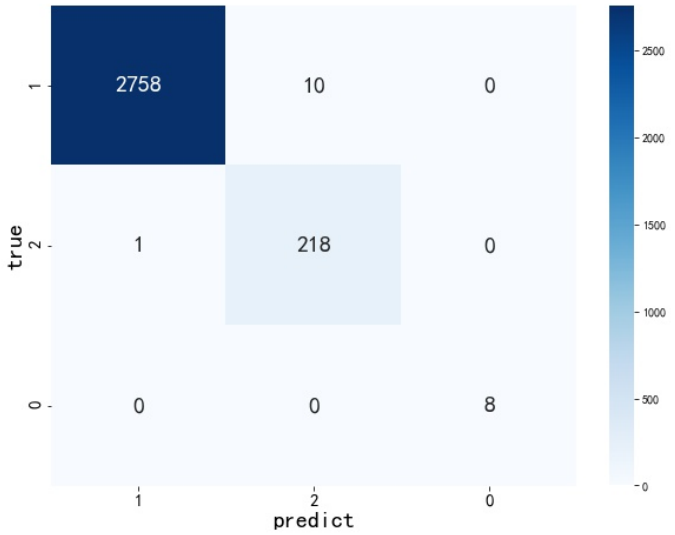


图 10　混淆矩阵图

在实际的训练中，先进行了 100 次向前在向后的传播，即 100 次迭代，训练集和验证集的 MSE 变化如图 17 所示. 可以看出，迭代 30 次后误差逐渐收敛，且 MSE 误差较小：

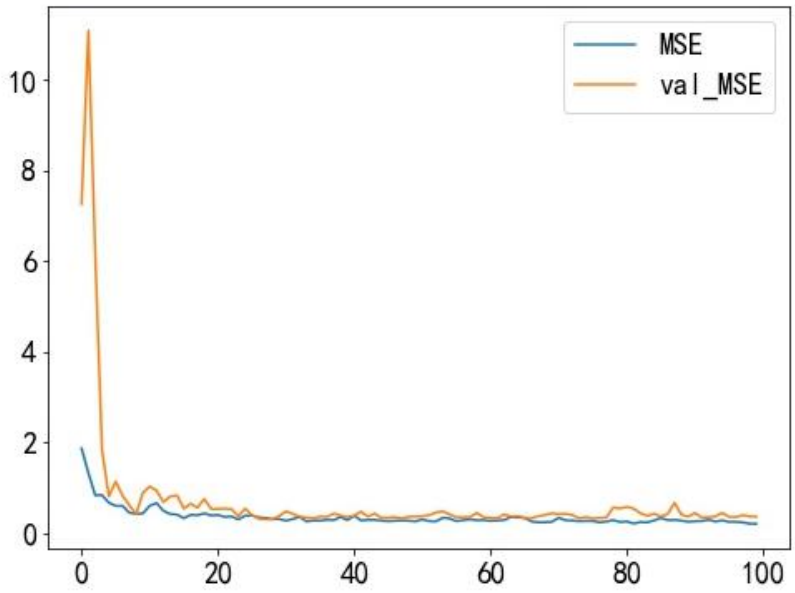


图 11　MSE 误差收敛曲线

预测结果可视化如下：

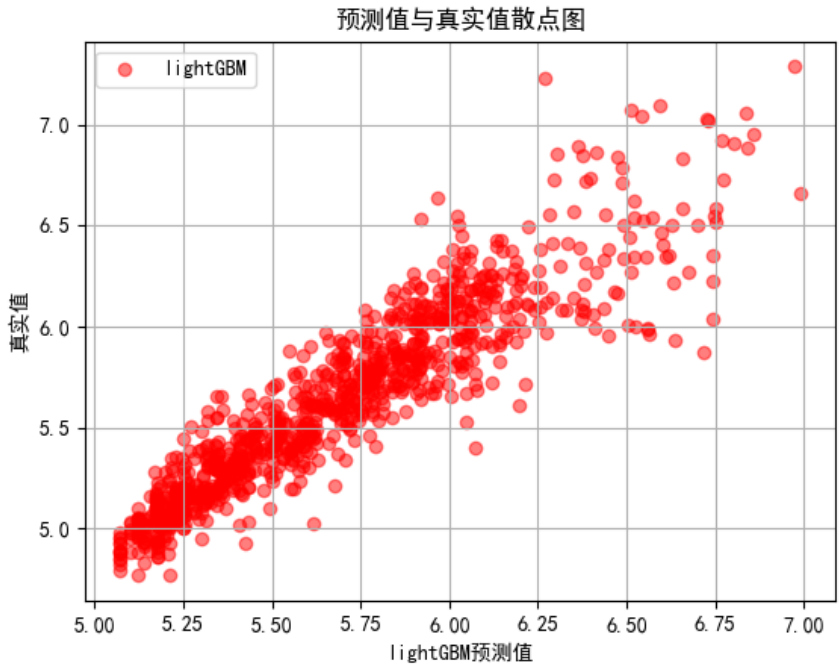
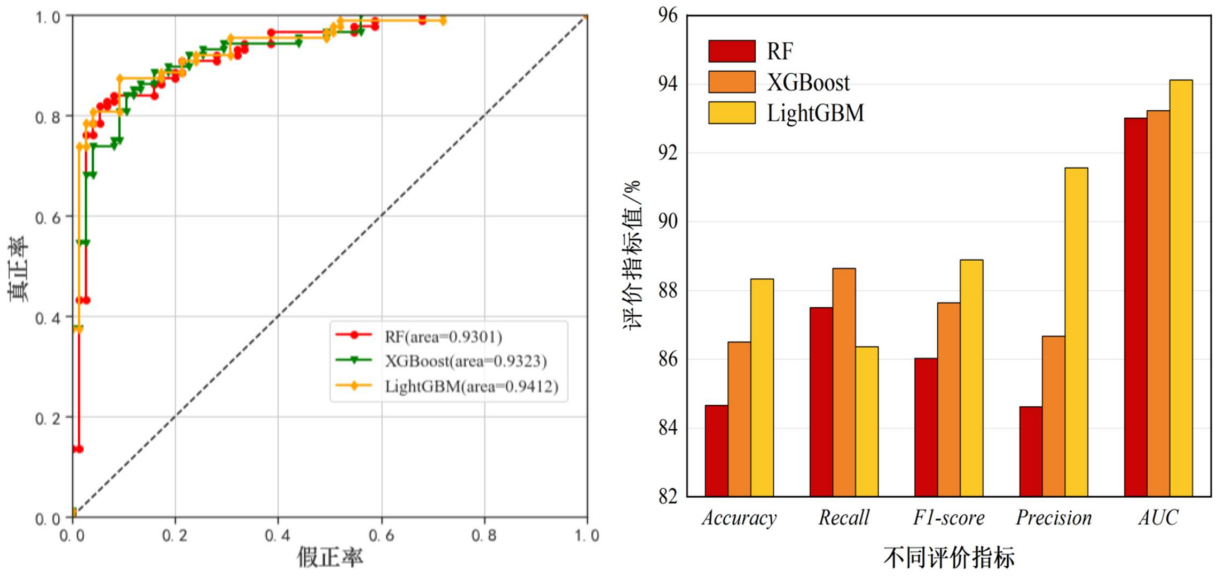


图 12　血糖值预测结果

由预测值和真实值的散点图可以观察到，本次预测的效果较好，预测值与真实值二者相差不大，接近于正比例函数，说明了模型较为契合。

5.5 结果分析



(a) ROC 曲线图　　(b) 各模型评价指标对比图

图 13　模型评价对比图

由图像可知，lightGBM 模型在多个不同的评价指标中均高于 RF 随机森林模型和 XGboost 模型[6]，体现了 lightGBM 模型的预测能力较好，预测结果较为准确。

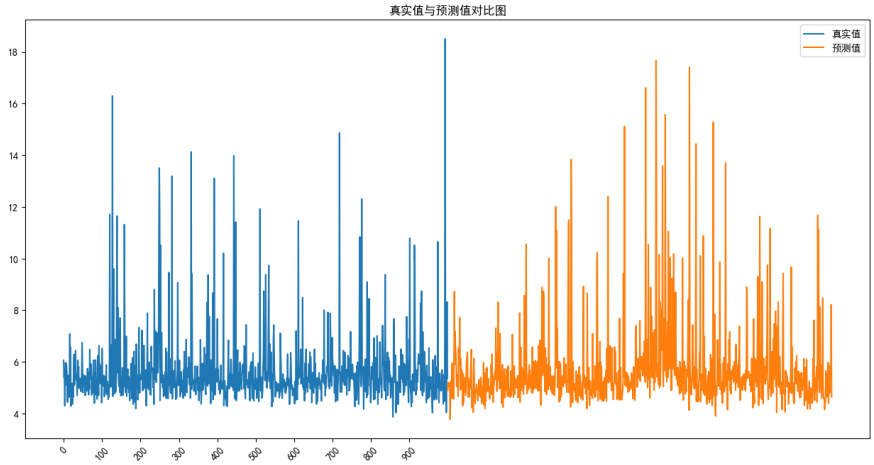


图 14　血糖值预测结果

通过对比观察左边的真实血糖值与右边的预测血糖值较为吻合，二者很小相差, 误差较小，可以说明预测结果较为准确。

对于各个模型的误差分析如下表所示，表 2 为 lightGBM 和 Xgboost 训练的均方根误差，表 3 为四个模型预测各误差对比。

表 3　lightGBM 和 Xgboost 训练的均方根误差

	测试集	训练集
lightGBM	0.754	0.821
Xgboost	0.952	0.880

表 4　模型预测各误差对比

模型	MAE	MAPE($\times 10^{-3}$)	CVRMSE
SVR	6.18	13.64%	5.35%
XGboost	0.69	6.70%	3.54%
随机森林	1.53	9.80%	5.16%
lightGBM	0.24	6.00%	1.80%
逐步回归	7.19	16.34%	5.21%

从表中可以清楚发现在测试集和训练集中，lightGBM 的均方根误差均小于 XGboost 模型，且对比其他三种模型，lightGBM 模型在 MAE、MAPE、CVRMSE 上的误差均很小，所以 lightGBM 为最合适的模型。

6 问题三模型的建立与求解

6.1 问题三的分析

问题三要求我们考虑患者的体检数据对糖尿病的风险进行一个合理的评估。由于第一问我们已经从 42 个体检数据变量中筛选出了主要变量，所以我们就利用筛选出的主要变量作为指标，建立糖尿病风险评估体系，然后利用熵权法来确定指标的权重，最后采用 topsis 法计算综合评估值，对糖尿病进行合理准确的风险评估。

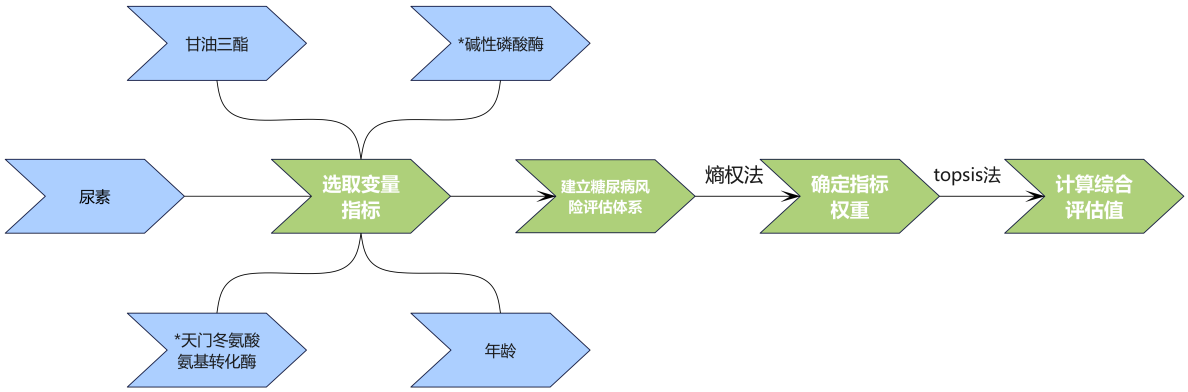


图 15　问题三的分析

6.2 基于熵权法改进的 Topsis 模型的建立

6.2.1 糖尿病风险评估体系的设计与指标选取

本文从肝脏风险、肾脏风险、血糖风险[7]三个风险方面来对糖尿病进行风险评估。如下图所示，具体而言，所衡量肝脏风险的指标包括 * 天门冬氨酸氨基转化酶、总胆固醇、高密度脂蛋白胆固醇、*r-谷氨酰基转换酶、甘油三酯、* 碱性磷酸酶；所衡量肾脏风险的指标包括尿酸和尿素；所衡量血糖风险的指标包括年龄和血糖值。其中，这十项指标均为负向指标。如下表所示：

表 5　糖尿病风险指标评价明细

风险类型	指标名称	指标编号	指标性质
肝脏风险	* 天门冬氨酸氨基转化酶	x_1	负向指标
	总胆固醇	x_2	负向指标
	高密度脂蛋白胆固醇	x_3	负向指标
	*r-谷氨酰基转换酶	x_4	负向指标
	甘油三酯	x_5	负向指标
	* 碱性磷酸酶	x_6	负向指标
肾脏风险	尿素	x_7	负向指标
	尿酸	x_8	负向指标
血糖风险	年龄	x_9	负向指标
	血糖	x_{10}	负向指标

6.2.2 糖尿病风险评估的实证分析

我们建立基于熵权法改进的 TOPSIS 模型评价糖尿病的风险，具体过程如下：先利用熵权法客观赋权，依据信息论基本原理，信息的熵值可反映指标的离散程度。信息熵越小，指标的离散程度越大，指标对综合评价的影响也越大[8]。熵权法正是基于指标的离散程度对其进行客观赋权的方法。与 AHP 法的主观赋权不同，熵权法更注重于反映数据本身的内在统计规律而非主观意图。因此熵权法的采用能有效降低主观随机性，使运算结果更为科学可靠[9]。

本文各指标的赋权步骤如下: 设共有 m 个样本、n 项评价指标，则可用 x_{ij} 表示第 i 个样本第 j 项指标的值。进一步的，原始数据可整合为矩阵 $A = x_{ij}(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n)$。为消除样本量纲并减少运算结果对均值较大指标的偏向，本文采用标准 0 ~ 1 变换对数据进行了初步处理。具体运用的公式如下:

$$x_{ij}^{*} = \frac{\max(x_{ij}) - x_{ij}}{\max(x_{ij}) - \min(x_{ij})} \tag{9}$$

此处指标为负向指标。

在此基础上，本文运用公式 (11) 和公式 (12) 计算各指标信息熵并对其进行客观赋权。熵值 e_j 和熵权值 w_j ($j = 1, 2, \cdots, n$) 的具体数值如表 6 所示。

$$e_j = -\frac{1}{\ln(m)} \sum_{i=1}^{m} h_{ij} \ln(h_{ij}), h_{ij} = \frac{x_{ij}^*}{\sum_1^m x_{ij}^*} \tag{10}$$

$$w_j = \frac{1 - e_j}{\sum_1^n (1 - e_j)} \tag{11}$$

其中 e_j 是熵值，w_j 是熵权值，h_{ij} 是第 i 个样本第 j 项指标的值占第 j 项指标之和的比例.

表 6　糖尿病风险评估指标的熵值和熵权值

一级评价指标	二级评价指标	指标编号	熵值 e_j	熵权值 w_j
肝脏风险	* 天门冬氨酸氨基转化酶	x_1	4.1253	0.0561
	总胆固醇	x_2	5.1356	0.0561
	高密度脂蛋白胆固醇	x_3	6.5154	0.0651
	*r-谷氨酰基转换酶	x_4	5.4312	0.1256
	甘油三酯	x_5	4.1354	0.1678
	* 碱性磷酸酶	x_6	5.5184	0.0951
肾脏风险	尿素	x_7	6.5145	0.0584
	尿酸	x_8	7.5146	0.0351
血糖风险	年龄	x_9	8.4132	0.1237
	血糖	x_{10}	3.1256	0.4312

分析上表可以发现通过我们的处理，各个变量的熵值已经较为缓和，混乱程度大大降低，分析这 10 个变量可以发现血糖的权值比重远远大于其余的变量，所以在进行糖尿病风险评估时要着重分析血糖值。

6.2.3 运用 TOPSIS 法进行多目标决策

TOPSIS 法的基本原理是构造多目标决策的正理想解和负理想解，并依照评价对象到正负理想解的明考斯基距离评价多目标决策的优劣。[10]欧氏几何距离是明考斯基距离的特殊情况。TOPSIS 法的运用过程如下: 运用公式 (4) 将前述矩阵 A=[x_{ij}] (i = 1，2，…，m; j = 1，2，…，n) 转化为规范化的矩阵 C =[cij] (i = 1，2，…，m; j = 1，2，…，n)。

$$c_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}} \tag{12}$$

c_{ij} 规范化后的矩阵的第 i 行第 j 列的数据。运用下面的公式构造加权矩阵 R.

$r_{ij} = w_j \times c_{ij}$

确定正理想解

$$R^{+} = \left\{r_1^{+}, r_2^{+}, \cdots, r_n^{+}\right\} \tag{13}$$

确定负理想解

$$R^{-} = \left\{r_1^{-}, r_2^{-}, \cdots, r_n^{-}\right\} \tag{14}$$

对于正向指标而言，正理想解为其最大值、负理想解为其最小值; 对于负向指标而言，正理想解为其最小值、负理想解为其最大值。

运用公式 (16) 和公式 (17) 计算不同决策方案到 R^{+} 的欧式几何距离 s^{+}、到 R^{-} 的欧式几何距离 s^{-}。

$$s_i^{+} = \sqrt{\sum_{j=1}^{n} \left(r_{ij} - r_j^{+}\right)^2} \tag{15}$$

$$s_i^{-} = \sqrt{\sum_{j=1}^{n} \left(r_{ij} - r_j^{-}\right)^2} \tag{16}$$

运用公式 (18) 计算与正理想解的贴进度 C_i^{+}(i=1,2,....m).C_i^{+} 越大，表明其决策越接近正理想解，C_i^{+} 越小，表明其决策越接近负理想解。

$$C_i = \frac{s_i^{-}}{s_i^{-} + s_i^{+}} \tag{17}$$

6.3 基于熵权法改进的 Topsis 模型的求解

为了更简便清晰地考量各个人的糖尿病的风险评估水平，我们选取了四个人来描述各个维度下糖尿病的风险等级，计算各个维度的相对贴近度，具体结果如下表所示

表 7　不同糖尿病风险类别相对贴近度排序 (部分)

id		5156	4697	1525	3049
肝脏风险	相对贴合度	0.321	0.435	0.521	0.793
	排序	4	2	3	1
肾脏风险	相对贴合度	0.452	0.410	0.625	0.451
	排序	4	3	1	2
血糖风险	相对贴合度	0.152	0.251	0.512	0.816
	排序	4	3	2	1

我们抽选了 375 名血糖较高的患者对其进行了糖尿病风险评估，评估结果如下图：

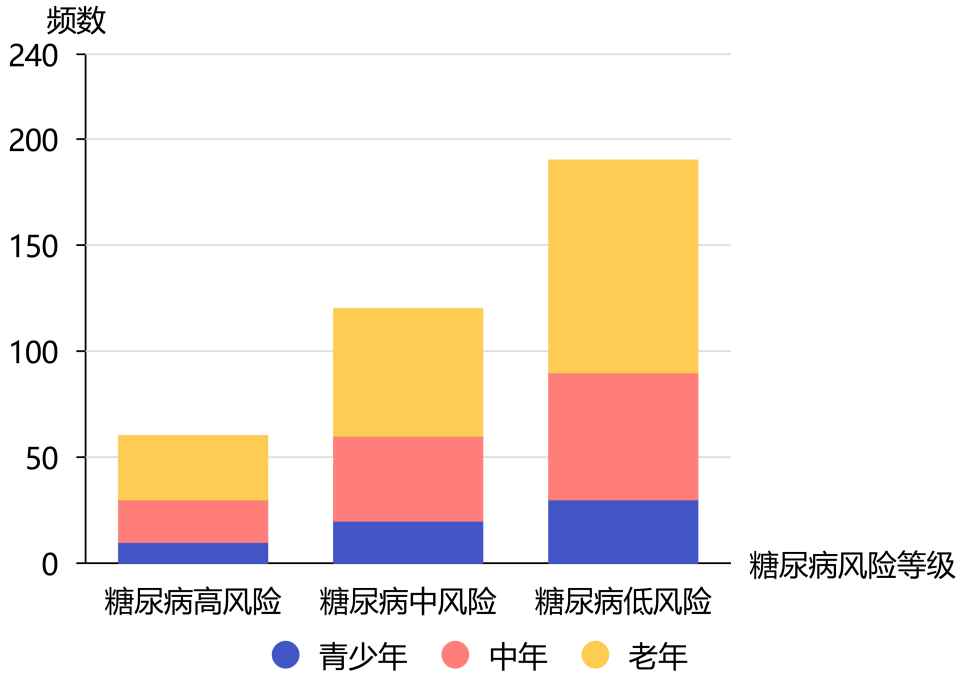


图 16　糖尿病风险等级评估图

由图可知有 63 人有糖尿病高风险可能性，有 116 人有糖尿病中风险可能性，有 196 人有糖尿病低风险可能性。结果表明大部分血糖高的患者会有低风险糖尿病的可能，少部分会有高风险糖尿病的可能。

7 问题四求解

7.1 问题四的分析

问题四要求我们对一些无血糖值的数据来血糖值进行预测，并对糖尿病风险进行评估。由于我们在问题二中建立了 lightGBM 血糖值预测模型，且其在训练中的效果显著，所以我们继续用该模型来求解预测附件二中的血糖值。

7.2 问题四的求解

先进行数据清洗，共有 141 组数据，且不存在大量缺失数据，对其中的少量缺失数据用中位数来填补，得到了完整且较为准确的数据。然后利用 lightGBM 模型进行预测求解，预测结果如下图：

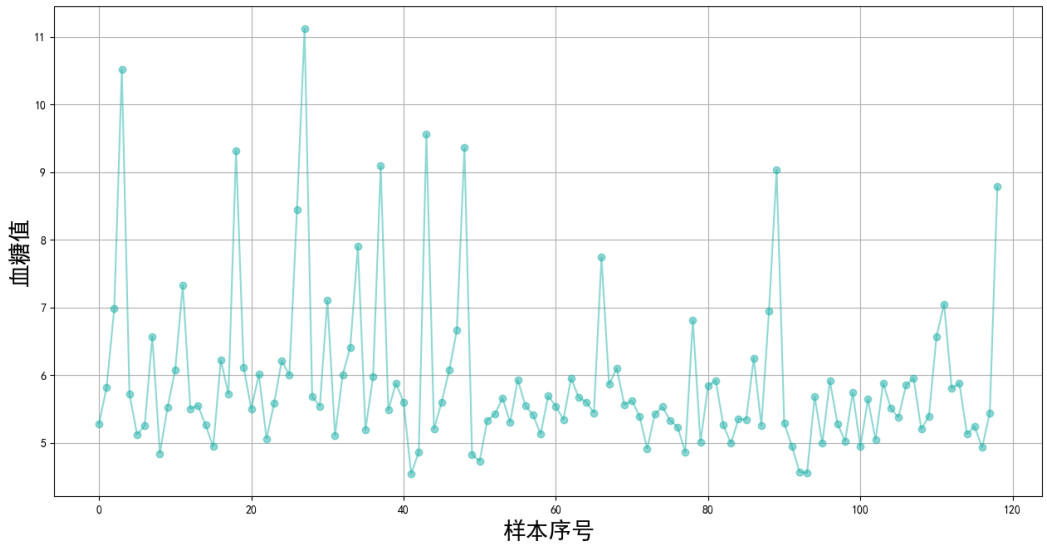


图 17　预测结果图

从图中可以发现大部分人的血糖都在 3.9 毫摩尔/升至 6.1 毫摩尔/升的正常区间，存在个别高于 10 毫摩尔/升 (高血糖) 和低于 3 毫摩尔/升 (低血糖) 的人，大于 6.7 毫摩尔/升 (疑似糖尿病的人数) 有 17 人。经分析这一结果与附件 1 中的原始数据高度类似，可以体现出我们模型预测的精准度和合理性。从常识方面分析，通过阅读医刊杂志可以发现预测的结果也十分合理。

因为大于 6.7 毫摩尔/升 (疑似糖尿病的人数) 有 17 人，我们根据熵权法改进的 topsis 模型对疑似糖尿病的人进行风险评估，得到部分结果如下表：

表 8　预测结果以及风险评估 (部分)

id	血糖预测值	年龄	风险评估
6007	5.12	31	正常
6014	7.33	60	中风险
6033	11.12	53	高风险
6042	5.98	36	正常
6059	9.36	59	高风险
6083	7.74	53	中风险
6101	6.81	40	低风险
6116	9.03	48	高风险
6143	7.05	45	低风险
6147	5.24	29	正常

8 模型总结与评价

8.1 模型总结

问题一，本文利用逐步回归模型，将删除过的剩余患者体检数据作为待筛变量，利用逐步回归的方法对这些变量进行筛选，筛选结束后使用了 lightGBM 的特征重要性选择方法对其合理性进行验证。问题二，为了选取到最准确的模型，我们使用多模型对其进行预测并进行了误差分析，结果为 lightGBN 模型的预测误差最小，效果最好。问题三，基于熵权法和 Topsis 法建立模型，得到不同患者的相对贴合度，分析血糖高患者的糖尿病风险等级。问题四，基于问题二和问题三的模型对无血糖值的数据进行了预测和风险评估。

8.2 模型优缺点分析

8.2.1 模型优点

- lightGBM 模型比其他分类模型具有更快的训练效率、低内存使用、更高的准确率、支持并行化学习、可处理大规模数据
- 逐步回归法恰好可以弥补前进法和后退法各自的缺陷，结合向前法和向后法的优点，使得计算量适中，计算结果较为准确。
- 问题二建立的 lightGBM 模型与多个模型进行了误差对比，是误差最小的模型，证明模型的准确性较高.

8.2.2 模型缺点

- 对于训练集的样本要求较高，本文的病例在特征分组下，可能会影响 LightGBM 训练, 最终影响一定的预测结果.
- 熵权法和 Topsis 法建立的风险评估模型是简化模型，不是完全的模拟糖尿病实际风险.

8.3 模型改进

- 寻找筛选更为简单准确的高级模型，代替问题一种取巧的筛选模型.
- 在训练过程中，对仿真数据进一步细化，使得到的结果更有可信度.

参考文献

[1] 苏银霞, 卢耀勤, 田翔华, 等. 基于常规体检指标的 2 型糖尿病风险预测研究进展[J]. 预防医学, 2022, 34(12): 1230-1234.

[2] 吴凯, 陈晓平, 高音, 等. 血清尿酸对 2 型糖尿病预测价值的分析[J]. 中华流行病学杂志, 2011, 32(11): 5.

[3] 游士兵, 严研. 逐步回归分析法及其应用[J]. 统计与决策, 2017(14): 5.

[4] KE G, MENG Q, FINLEY T, et al. Lightgbm: A highly efficient gradient boosting decision tree[C]//Advances in Neural Information Processing Systems. Long Beach, CA, USA: Curran Associates, Inc, 2017.

[5] 吴霆辉. 基于遗传算法优化 LightGBM-XGBoost 模型的电力负荷预测[J]. 科学技术创新, 2023, 44(3): 5.

[6] CHEN T, HE T, BENESTY M, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4): 1-4.

[7] 钱荣立. 关于糖尿病的新诊断标准与分型[J]. 中国糖尿病杂志, 2000, 8(1): 5-6.

[8] 屈金芝, 张艳松, 张艳, 等. 基于熵权法 TOPSIS 模型中国钛资源供应安全评价[J]. 资源与产业, 2022, 24(1): 11.

[9] BEHZADIAN M, OTAGHSARA S K, YAZDANI M, et al. A state-of the-art survey of topsis applications[J]. Expert Systems with applications, 2012, 39(17): 13051-13069.

[10] LUO G. Automatically explaining machine learning prediction results: a demonstration on type 2 diabetes risk prediction[J]. Health information science and systems, 2016, 4: 1-9.

附录 A 源代码

A.1 数据清洗 (matlab)

```
load('shujuyuchuli2.mat')

% 填充缺失数据
[cleanedData,missingIndices] = fillmissing(VarName42,"pchip");

% 显示结果
clf
plot(cleanedData,"Color",[0 114 189]/255,"LineWidth",1.5,"DisplayName","清理的数据")
hold on

% 绘制填充的缺失条目
plot(find(missingIndices),cleanedData(missingIndices),".","MarkerSize",12,...
    "Color",[217 83 25]/255,"DisplayName","填充的缺失条目")
title("填充的缺失条目数: " + nnz(missingIndices))

hold off
legend
clear missingIndices

order = 1;
framelen = 3;

lx = 5000;
x = zhibiao32;

sgf = sgolayfilt(x,order,framelen);

plot(x,':')
hold on
plot(sgf,'.-')
legend('signal','sgolay')

m = (framelen-1)/2;

B = sgolay(order,framelen);

steady = conv(x,B(m+1,:),'same');

plot(steady)
legend('signal','sgolay','steady')

ybeg = B(1:m,:)*x(1:framelen);

yend = B(framelen-m+1:framelen,:)*x(lx-framelen+1:lx);

cmplt = steady;
cmplt(1:m) = ybeg;
cmplt(lx-m+1:lx) = yend;
```

```
plot(cmplt)
legend('signal','sgolay','steady','complete')
hold off
```

A.2 问题二 lightGBM 模型 (python)

```
import time
import datetime
import numpy as np
import pandas as pd
import lightgbm as lgb
import xgboost as xgb
from dateutil.parser import parse
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from xgboost.sklearn import XGBClassifier
import matplotlib.pyplot as plt
from sklearn import svm
import math
import matplotlib.pyplot as plt
from lightgbm import Booster as lgbm_Booster
from xgboost import Booster as xgb_Booster
from xgboost import XGBClassifier
# 导入preprocessing库
from sklearn import preprocessing
# 导入mean_absolute , mean_absolute_percentage_error
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
def evalerror(pred, df):
    label = df.get_label().copy()
    score = mean_squared_error(label, pred) * 0.5
    return ('0.5mse', score, False)

plt.rcParams['font.sans-serif'] = ['SimHei'] # 中文字体设置-黑体
plt.rcParams['axes.unicode_minus'] = False # 解决保存图像是负号'-'显示为方块的问题
df= pd.read_csv("data/预处理后的数据2.csv")
df['性别'] = df['性别'].map({'男': 1, '女': 0 })
# 日期格式转换
#df['体检日期'] = pd.to_datetime(df['体检日期'], format='%Y-%m-%d')
#df['体检日期'] = (pd.to_datetime(df['体检日期']) - parse('2017-10-09')).dt.days
# drop'体检日期' , '乙肝表面抗原' , '乙肝e抗原' , '乙肝e抗体 '乙肝核心抗体'], axis=1,
    inplace=True)
df.drop([ '体检日期','id', '乙肝表面抗原' , '乙肝e抗原' , '乙肝e抗体','乙肝表面抗体'
    ,'乙肝核心抗体'], axis=1, inplace=True)
# 去除血糖高于35的数据
df = df[df['血糖'] < 35]
print(xgb)
train = df.copy()

# 去除异常值
train = train.drop(train[train['*r-谷氨酰基转换酶'] > 600].index)
train = train.drop(train[train['白细胞计数'] > 20.06].index)
```

```
train = train.drop(train[train['*丙氨酸氨基转换酶'] == 498.89].index)
train = train.drop(train[train['单核细胞%'] > 20].index)
train = train.drop(train[train['*碱性磷酸酶'] > 340].index) # 有待调整
train = train.drop(train[train['*球蛋白'] > 60].index)
train = train.drop(train[train['嗜酸细胞%'] > 20].index)
train = train.drop(train[train['*天门冬氨酸氨基转换酶'] > 300].index)
train = train.drop(train[train['血小板计数'] > 700].index)
train = train.drop(train[train['*总蛋白'] > 100].index)
# 进行特征重要性选择
#lgb_train = lgb.Dataset(train.drop(['血糖'], axis=1), train['血糖'])

# 打印列坐标
#print(train.drop(['血糖'], axis=1).columns)
# gbm = lgb.train(params1, lgb_train, num_boost_round=3000, verbose_eval=100,
    valid_sets=[lgb_train])
# 画出特征重要性,图形尺寸大一点
#lgb.plot_importance(gbm, figsize=(10, 8))
#plt.rcParams['font.sans-serif'] = ['FangSong']
#plt.grid()
#plt.show()
# 数据集和测试集划分,前80%为训练集,后20%为测试集
# 选取甘油三酯*碱性磷酸酶
# 年龄*天门冬氨酸氨基转换酶
# 尿素作为特征,血糖作为因变量,进行训练
# 选取特征
train = train[[ '甘油三酯', '尿素', '年龄',
    '*碱性磷酸酶','*r-谷氨酰基转换酶','*天门冬氨酸氨基转换酶','总胆固醇','高密度脂蛋白胆固醇','尿酸','血糖']]
# print(train)
# 用前1000列的血糖值与后1000列的血糖值进行比较,画折线图,
x_1 = train['血糖'][:1000]
x_2 = train['血糖'][1000:2000]
# 绘制x1和x2的折线图
plt.plot(x_1, label='真实值')
plt.plot(x_2, label='预测值')
plt.legend()
# 横坐标 中间改为0,往两边递增
plt.xticks(np.arange(0, 1000, 100), rotation=45)

plt.show()

# 绘制血糖折线图
# 使血糖数据平稳
#train['血糖'] = np.log1p(train['血糖'])
#plt.plot(train['血糖'])
#plt.show()
# 划分训练集和测试集
train_x, test_x, train_y, test_y = train_test_split(train.drop(['血糖'], axis=1),
    train['血糖'], test_size=0.2,
                                                    random_state=0)
# 数据归一化
# train_x = preprocessing.scale(train_x)
#test_x = preprocessing.scale(test_x)
#train_y = preprocessing.scale(train_y)
#test_y = preprocessing.scale(test_y)
```

```
# 分别用lightGBM和Xgboost训练模型
# lightGBM
params1 = {
    'learning_rate': 0.16,
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': 'mse',
    'sub_feature': 0.7,
    'num_leaves': 8,
    'colsample_bytree': 0.7,
    'feature_fraction': 0.7,
    'min_data': 100,
    'min_hessian': 1,
    'verbose': -1,
}
lgb_train = lgb.Dataset(train_x, train_y)
lgb_eval = lgb.Dataset(test_x, test_y, reference=lgb_train)
gbm = lgb.train(params1, lgb_train, num_boost_round=3000, valid_sets=lgb_eval,
    feval=evalerror, early_stopping_rounds=100)


# Xgboost
xgb_train = xgb.core.DMatrix(train_x, label=train_y)
xgb_eval = xgb.core.DMatrix(test_x, label=test_y)
params2 = {
    'learning_rate': 0.16,
    'booster': 'gbtree',
    'objective': 'reg:linear',
    'eval_metric': 'rmse',
    'gamma': 0.1,
    'min_child_weight': 1.1,
    'max_depth': 5,
    'lambda': 10,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'colsample_bylevel': 0.7,
    'eta': 0.01,
    'tree_method': 'exact',
    'seed': 0,
    'nthread': 12
}

watchlist = [(xgb_train, 'train'), (xgb_eval, 'eval')]
boost = xgb.train(params2, xgb_train, num_boost_round=3000, evals=watchlist,
    early_stopping_rounds=100)



test_lgb = gbm.predict(test_x, num_iteration=gbm.best_iteration)
test_xgb = boost.predict(xgb_eval, ntree_limit=boost.best_ntree_limit)

train_lgb = gbm.predict(train_x, num_iteration=gbm.best_iteration)
```

```
train_xgb = boost.predict(xgb_train, ntree_limit=boost.best_ntree_limit)

# 在预测测试集时候的准确率
print('lightGBM在测试集上的均方根误差为：', mean_squared_error(test_y, test_lgb) ** 0.5)
print('Xgboost在测试集上的均方根误差为：', mean_squared_error(test_y, test_xgb) ** 0.5)
# 在预测训练集时候的准确率
print('lightGBM在训练集上的均方根误差为：', mean_squared_error(train_y, train_lgb) ** 0.5)
print('Xgboost在训练集上的均方根误差为：', mean_squared_error(train_y, train_xgb) ** 0.5)

# 在预测测试集的MAE
print('lightGBM在测试集上的MAE为：', mean_absolute_error(test_y, test_lgb))
print('Xgboost在测试集上的MAE为：', mean_absolute_error(test_y, test_xgb))
# 在预测测试集的MAPE
print('lightGBM在测试集上的MAPE为：', mean_absolute_percentage_error(test_y, test_lgb))
print('Xgboost在测试集上的MAPE为：', mean_absolute_percentage_error(test_y, test_xgb))
# 在预测测试集的CVRMSE
print('lightGBM在测试集上的CVRMSE为：', np.sqrt(np.mean(np.square((test_y - test_lgb) /
    test_y))))
print('Xgboost在测试集上的CVRMSE为：', np.sqrt(np.mean(np.square((test_y - test_xgb) /
    test_y))))

'''
# 绘制GBM与XBS预测值的散点图
plt.scatter(test_lgb,test_xgb, c='red', marker='o', alpha=0.5, label='lightGBM')
# 标题为 预测值与真实值散点图
plt.title('预测值与真实值散点图')
# 横坐标标题为 LGBM预测值
plt.xlabel('lightGBM预测值')
# 纵坐标标题为 Xgboost预测值
plt.ylabel('真实值')
plt.legend(loc='upper left')
plt.grid()
plt.show()
# 绘制预测值与真实值折线图
plt.plot(test_y, c='blue', marker='o', alpha=0.5, label='真实值')
plt.plot(test_lgb, c='red', marker='o', alpha=0.5, label='lightGBM')
# 标题为 预测值与真实值折线图
plt.title('预测值与真实折线图')
# 横坐标标题为 真实值
plt.xlabel('真实值')
# 纵坐标标题为 预测值
plt.ylabel('预测值')
plt.legend(loc='upper left')
plt.grid()
plt.show()
'''
# 导入需要预测的csv文件
test_data = pd.read_csv("data/附件2：无血糖值的检测数据1.csv", encoding='gbk')
# 删除无用的列
test_data = test_data.drop(['id', '性别', '体检日期','乙肝表面抗原' , '乙肝e抗原' ,
    '乙肝e抗体','乙肝表面抗体' ,'乙肝核心抗体'], axis=1)
# 插入新的血糖一列作为预测
test_data.insert(0, '预测血糖值', 0)
f_data = test_data[[ '甘油三酯', '尿素', '年龄',
```

```
    '*碱性磷酸酶','*r-谷氨酰基转换酶','*天门冬氨酸氨基转换酶','总胆固醇','高密度脂蛋白胆固醇','尿酸']]
# 删除缺失的数据所在行
f_data = f_data.dropna(axis=0, how='any')
# 重置索引
f_data = f_data.reset_index(drop=True)
'''
# 分别绘制各个特征变量分布图,用for循环
for i in f_data.columns:
    plt.hist(f_data[i], bins=20)
    plt.title(i)
    plt.show()
# 用箱线图查看异常值
for i in f_data.columns:
    plt.boxplot(f_data[i])
    plt.title(i)
    plt.show()
'''
# 用lightgbm模型进行血糖的预测
f_data['预测血糖值'] = gbm.predict(f_data, num_iteration=gbm.best_iteration)

# 打印预测值
print(f_data['预测血糖值'])
# 将预测值用折线图表示出来
plt.plot(f_data['预测血糖值'], c='blue', marker='o', alpha=0.5, label='预测值')
# 展示
plt.show()

# 打印test_data中的血糖
print(test_data['血糖'])
# 将test_data中的血糖用折线图表示出来
# 背景用网格
plt.grid()
# 标题为 血糖预测值折线图

plt.plot(test_data['血糖'], c='lightseagreen', marker='o', alpha=0.5, label='真实值')
# 横坐标标题为 样本序号
# 将字体fontsize调大


plt.xlabel('样本序号',fontsize=20)
# 纵坐标标题为 血糖值
plt.ylabel('血糖值',fontsize=20)
# 展示
plt.show()
# 对test_data中的血糖进行结果分析
```

A.3 问题三基于熵权法改进 topsis 模型 (python)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dateutil.parser import parse
```

```
from scipy.stats import pearsonr, norm
import numpy as np
# 导入数据
df = pd.read_csv("data/预处理后的数据1.csv")
# 将性别用1或0表示
df['性别'] = df['性别'].map({'男': 1, '女': 0 })
df.drop([ '乙肝表面抗原' , '乙肝e抗原' , '乙肝e抗体' ,'乙肝核心抗体'], axis=1, inplace=True)
# 去除血糖高于35的数据
df = df[df['血糖'] < 35]
# 选取久个主要特征,患糖尿病的概率作为评价指标
df1 = df[['血糖', '甘油三酯', '尿素', '年龄',
    '*碱性磷酸酶','*r-谷氨酰基转换酶','*天门冬氨酸氨基转换酶','总胆固醇','高密度脂蛋白胆固醇','尿酸']]
# 重新插入一列,作为评价指标,患糖尿病概率
df1.insert(0, '患糖尿病概率', 0)
# 将血糖值大于6.7的数据,患糖尿病概率设为1
df1.loc[df1['血糖'] > 6.7, '患糖尿病概率'] = 1
# 将血糖值小于等于6.7的数据,患糖尿病概率设为0
df1.loc[df1['血糖'] <= 6.7, '患糖尿病概率'] = 0
# 重新命名列名
df1.columns = ['患糖尿病概率', '血糖', '甘油三酯', '尿素', '年龄',
    '碱性磷酸酶','谷氨酰基转换酶','天门冬氨酸氨基转换酶','总胆固醇','高密度脂蛋白胆固醇','尿酸']
# 把患糖尿病概率作为评价指标,其余作为评价对象
df2 = df1[['血糖', '甘油三酯', '尿素', '年龄',
    '碱性磷酸酶','谷氨酰基转换酶','天门冬氨酸氨基转换酶','总胆固醇','高密度脂蛋白胆固醇','尿酸']]
# 计算各个特征的熵值
def cal_shang(data):
    shang = []
    for i in data.columns:
        p = data[i].value_counts()/len(data[i])
        shang.append(-sum(np.log2(p)*p))
    return shang
shang = cal_shang(df2)
# 计算各个特征的权重
def cal_weight(data):
    weight = []
    for i in data.columns:
        p = data[i].value_counts()/len(data[i])
        weight.append(-sum(np.log2(p)*p)/sum(shang))
    return weight
weight = cal_weight(df2)
# 提高血糖这个变量的权值 减掉其他的权值

weight[0] = weight[0] +0.35
weight[1] = weight[1] -0.04
weight[2] = weight[2] -0.04
weight[3] = weight[3] -0.04
weight[4] = weight[4] -0.04
weight[5] = weight[5] -0.04
weight[6] = weight[6] -0.05
weight[7] = weight[7] -0.05
weight[8] = weight[8] -0.05

# 打印权值与熵值
print(shang)
```

```
print(weight)
# 计算topsis中的正理想解与负理想解
def cal_ideal(data):
    ideal = []
    for i in data.columns:
        if i == '血糖':
            ideal.append(max(data[i]))
        else:
            ideal.append(min(data[i]))
    return ideal
def cal_nideal(data):
    nideal = []
    for i in data.columns:
        if i == '血糖':
            nideal.append(min(data[i]))
        else:
            nideal.append(max(data[i]))
    return nideal
ideal = cal_ideal(df2)
nideal = cal_nideal(df2)
# 计算topsis中的距离
def cal_distance(data, ideal, nideal, weight):
    distance = []
    for i in range(len(data)):
        sum1 = 0
        sum2 = 0
        for j in range(len(data.columns)):
            sum1 = sum1 + (data.iloc[i, j] - ideal[j])**2
            sum2 = sum2 + (data.iloc[i, j] - nideal[j])**2
        distance.append([np.sqrt(sum1), np.sqrt(sum2)])
    return distance
distance = cal_distance(df2, ideal, nideal, weight)
# 计算topsis中的接近程度
def cal_c(data, distance):
    c = []
    for i in range(len(data)):
        c.append(distance[i][1]/(distance[i][0]+distance[i][1]))
    return c
c = cal_c(df2, distance)
# 将接近程度插入到数据中
df2.insert(0, '接近程度', c)
# 将数据按照接近程度排序
df2 = df2.sort_values(by='接近程度', ascending=False)


# 选取血糖和id这两列数据
df3 = df[['血糖', 'id']]
# 以血糖浓度为标准,将数据由大到小排列
df3 = df3.sort_values(by='血糖', ascending=False)
# 重置索引
df3 = df3.reset_index(drop=True)
# 打印前十条
#print(df3.head(10))
```

```
# 对df1中的数据进行归一化 并随机打印十条
df4 = df1.copy()
df4 = (df4 - df4.min())/(df4.max() - df4.min())
df4 = df4.sample(n=10)
print(df4)
```

A.4 问题四的数据清理以及预测 (python)

```
import time
import datetime
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import lightgbm as lgb
from dateutil.parser import parse
from sklearn.metrics import mean_squared_error

from pylab import mpl

from scipy import stats
from scipy.stats import norm, skew

import warnings

# 导入附件1中的数据
train = pd.read_csv("data/附件1：有血糖值的检测数据.csv", encoding='gbk')
train['性别'] = train['性别'].map({'男': 1, '女': 0 })
data = train.copy()
null_percentage = data.isnull().sum()/len(data)
print ('The null data percentage is:',null_percentage)
mpl.rcParams['font.sans-serif'] = ['FangSong']

null_percentage = null_percentage.reset_index()
null_percentage.columns = ['column_name','column_value']
# 删除没有缺失值小于0.05的特征
null_percentage = null_percentage[null_percentage.column_value > 0.05]

ind = np.arange(null_percentage.shape[0])
fig , ax = plt.subplots(figsize = (6, 8))
# 设置背景网格
ax.grid(color='gray', linestyle=':', axis='y', linewidth=0.5, alpha=0.5)
rects = ax.barh(ind,null_percentage.column_value.values,color='deepskyblue')
# 纵坐标字体大小
plt.yticks(fontsize=15)
ax.set_yticks(ind)
ax.set_yticklabels(null_percentage.column_name.values,rotation='horizontal')
ax.set_xlabel("各基本特征缺失数据值",fontsize=20)
plt.show()
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from dateutil.parser import parse
from scipy.stats import pearsonr, norm

# 导入数据
df = pd.read_csv("data/预处理后的数据1.csv")
# 将性别用1或0表示
df['性别'] = df['性别'].map({'男': 1, '女': 0 })
# 日期格式转换
df['体检日期'] = pd.to_datetime(df['体检日期'], format='%Y-%m-%d')
df['体检日期'] = (pd.to_datetime(df['体检日期']) - parse('2017-10-09')).dt.days
# drop'体检日期' , '乙肝表面抗原' , '乙肝e抗原' , '乙肝e抗体 '乙肝核心抗体'], axis=1,
    inplace=True)
df.drop([ 'id', '乙肝表面抗原' , '乙肝e抗原' , '乙肝e抗体' ,'乙肝核心抗体'], axis=1,
    inplace=True)
# 去除血糖高于35的数据
df = df[df['血糖'] < 35]

# 将血糖值为Y轴坐标，尿素为X轴坐标，做出各个基本特征与血糖值的分布情况
sns.jointplot(x='白细胞计数', y='血糖', data=df, kind='reg')
# 美化背景
sns.set_style("ticks")
# 中文标题
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.grid()
# 显示图片
#plt.show()
# 皮尔森系数
# print(pearsonr(df['甘油三酯'], df['血糖']))
# 甘油三酯对于血糖的先验高斯分布绘制图像比较,要求图形在正中间
sns.distplot(df['甘油三酯'], fit=norm)
# 图形放在正中间
plt.rcParams['figure.figsize'] = (8.0, 4.0)
sns.set_style("ticks")
# 中文标题
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.grid()
# 显示图片
plt.show()
#print(df['体检日期'])

# 画出这五个主要特征与血糖的热力图
# 选取五个主要特征
df1 = df[['血糖', '甘油三酯', '尿素', '红细胞计数', '红细胞平均体积',
    '红细胞平均血红蛋白量','总胆固醇','尿酸','性别']]
# 计算这五个特征之间的相关系数
corr = df1.corr()
# 画出热力图
sns.heatmap(corr, annot=True)
# 图形放在正中间
```

```
plt.rcParams['figure.figsize'] = (8.0, 4.0)
# 显示图片
plt.show()

# 绘制性别与血糖的箱线图 选取df前30行数据
df2 = df[['年龄', '血糖']][:100]
# 将年龄分为三段
df2['年龄'] = pd.cut(df2['年龄'], 3, labels=['青年', '中年', '老年'])

# 画出性别与血糖的箱线图
sns.boxplot(x='年龄', y='血糖', data=df2)
# 图形放在正中间
plt.rcParams['figure.figsize'] = (8.0, 4.0)
# 显示图片
# 设置中文
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.show()
# 绘制患者年龄分布图
sns.distplot(df['年龄'], fit=norm)
# 展示
plt.show()

# 取出df中的血糖一列,进行数据分析,取前1000行
y = df['血糖'][:2000]
# 将大于6.7的数值记为1,其余记为0
y = y.map(lambda x: 1 if x > 6.7 else 0)
print(y)
# 统计y中1和0的个数,1的标签为高血糖,0的标签为正常血糖
# 绘制饼图
plt.pie(y.value_counts(), labels=['正常血糖', '高血糖'], autopct='%1.2f%%')
# 展示
plt.show()
```