所属类别	2023 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2305023

母亲身心健康对婴儿行为和睡眠的研究

摘要

本文研究了母亲的生理指标和心理指标与婴儿的行为特征和睡眠质量之间的关系和变化规律。

对于问题一，本文首先列出变量需满足的潜在条件，并对数据进行预处理，处理了异常值和缺失值，然后通过箱线图和热力图进行可视化分析。根据不同类型的变量，本文选择使用 Pearson **相关系数**、Spearman **相关系数**和 Kendall’s tau-b **相关系数**进行计算并绘制热力图。最终综合分析后得出结论：母亲的生理指标对婴儿行为特征和睡眠质量的影响较小，而母亲的心理指标对婴儿行为特征和睡眠质量的影响较大。因此，应关注母亲的身心健康，以促进婴儿的健康成长。

对于问题二，为提高模型的精确度，首先，本文对分类变量进行了**热编码**处理。在可适用于本题目场景的六套模型中，选择了**随机森林模型**进行模型建立。然后，本文还采取**网格搜索**来对此模型进行参数调优，以得到最优模型。最后，使用最优模型对题目中给出的 20 组数据进行预测，预测结果如表 2 所示。

对于问题三，首先从第一问的结论中得出，产妇心理指标与婴儿行为特征呈**较为显著的正相关**。本文进一步用 **Spearman 线性回归图法**，**三维柱状图**，**三维曲面图法**证明了这一结论。接着，用**偏最小二乘回归**和**逻辑回归**拟合函数，自变量为三个指标，因变量为婴儿行为特征。由于婴儿行为特征为**定类变量**，初步可以判断**逻辑回归拟合程度更好**。但经测试发现这俩个方法求得的三个指标的系数并不全为正，与之前的分析**相违背**，推测是由于数据集过小，离群数据过多导致的。因此本文舍弃了拟合函数的思想，从**均值，中位数，众数**中选择一个能够较好反应某个婴儿行为特征指标的量作为依据，最终选择了**中位数**。最后，根据题目信息构建**微分方程**表达式，使用**非线性规划**进行求解，求出**目标函数**，随后引入**约束条件**，最终得到本题的答案：CBTS、EPDS、HADS 分别下降 10，15，11 和总费用为 46896.7 以及下降 11，16，12 总费用为 50897.36。

对于问题四，本文首先绘制相关性矩阵图和三维曲面图粗略分析三个特征之间的关系。接着，本文采取**因子分析法**中的**因子载荷矩阵**显示出每个原始变量对于提取的潜在因子的贡献程度，根据这个结果，可以列出综合睡眠质量的**评判标准**。然后，本文使用 K-Means **聚类算法**对数据进行聚类分析，并对聚类结果进行**重新排序和赋值**。接着，类似于问题二的处理方法，将数据进行**热编码**，使用**随机森林模型**进行建模和调优，并最终预测题目给出的 20 组数据的综合睡眠质量，预测结果如表 8 所示。

对于问题五，为了有利于分析，本文决定采取变量与变量之间的分析，即直接分析产妇心理指标和睡眠评分标准的关系。首先，用 **Pearson 相关系数**和**箱线图**初步分析得到产妇心理指标和睡眠评分标准皆呈**负相关**且 CBTS 的影响很小。接着，建立**偏最小二乘回归(PLSR)**回归方程，得到 CBTS 的系数为正，因为 CBTS 相关性极弱，所以模型拟合出正系数现象属于合理范畴。最后得到自变量为三个指标，因变量为睡眠评分标准的线性函数，与第三问同理设立**目标函数**和**约束条件**，然后进行**规划问题**求解。最终得到问题五的答案：CBTS、EPDS、HADS 分别下降 0，17，0 总费用为 12730。

关键字：相关性分析，规划问题，随机森林，逻辑回归，偏最小二乘，微分方程

一．问题重述

1.1 问题背景

母亲在婴儿的成长过程中扮演着关键角色。她不仅为婴儿提供物质上的基本保障，更重要的是通过情感支持和建立安全感来促进婴儿的健康发展。然而，如果母亲面临抑郁、焦虑或过度压力等心理困扰，可能会对婴儿产生一定的负面影响，涉及认知、情感以及社会行为等方面。特别是压力过大的母亲极有可能影响婴儿的生理和心理发展，这一影响具体可以体现在婴儿的睡眠质量等方面。所以，关注和支持母亲的心理健康对于婴儿的全面发展至关重要。

1.2 问题提出

问题一：通过研究母亲的身体指标和心理指标对婴儿的行为特征和睡眠质量的影响规律，分析母亲的身心健康是否对婴儿的成长产生影响。

问题二：通过建立婴儿的行为特征与母亲的身体指标与心理指标的关系模型，对题目所给出的 20 组数据，判断他们分别属于什么类型。

问题三：通过建立 CBTS、EPDS 和 HADS 的患病程度与治疗费用的关系模型，并结合问题二，给出 238 号婴儿的行为特征从矛盾型转为中等型所需要的最少治疗费用。

问题四：通过建立婴儿综合睡眠质量与母亲的身体指标、心理指标的关联模型，对题目所给出的 20 组数据，预测他们的综合睡眠质量。

问题五：在问题三的基础上，结合问题四，要让 238 号婴儿的睡眠质量评级为优，判断问题三中的治疗策略是否需要进行调整。如果需要调整，则具体说明如何调整。

二．问题分析

2.1 问题一的分析

根据题意，需要对母亲的身体指标和心理指标对婴儿的行为特征和睡眠质量的影响规律进行研究。已知给出的 390 组数据，为了保证研究的可信性，本文将结合数据需满足的潜在条件进行初步筛查，查看是否存在异常值和缺失值，并根据实际情况进行处理，以减少对后续分析的影响。

为了帮助更好地理解二者之间的关联，本文将清洗后的数据进行预处理并初步可视化，直观地给出母亲的身心健康是否影响婴儿的成长的定性结论。由于题目中给出的用于判断母亲身心健康和婴儿成长指标的参数较多，且有些参数之间关联性不强，定量和定类数据参杂其中，本文将根据实际进行分组选择适当的统计方法分析变量之间的关联性，进一步给出母亲的身心健康是否对婴儿成长产生影响的定量结论。

2.2 问题二的分析

根据题意，需要建立婴儿的行为特征与母亲的身体指标与心理指标的关系模型。首先，进行数据的清理，考虑到问题一中预处理后的所有数据，包括含有离群值的数据行。但因为舍去含有离群值的数据行会导致数据变少反而降低了结果的精确度，并且为了保证数据的完整性和不偏性，本文选择保留所有的数据。

为避免分类变量被错误地当作连续值处理，影响后续算法的学习和预测精度，本文将考虑使用热编码来处理分类变量。通过热编码，可以将分类变量转换为二进制表示，避免了数值大小的误导，并确保算法对这些特征的处理更加准确。为验证猜想的正误，本文还将比较热编码前后数据预测的精确度，选择精确度较高的进行后续的分析。

在建立关系模型方面，本文比较了六个符合条件的模型，并选择精确度最高的模型来进行建模。在模型建立过程中，本文也将运用网格法进行超参数调优，以提高模型性

能和预测准确度。最后，本文将使用优化后的模型对题目所给出的 20 组数据进行预测，得出问题二的答案。

2.3 问题三的分析

根据题意，需要计算出 238 号婴儿的行为特征从矛盾型转为中等型的最小费用。因为身体指标对婴儿的影响较小，本道题直接分析心理指标。首先，绘制图表的形式进行定性分析，以观察这些指标之间的趋势和相关性。然后建立出 CBTS、EPDS 和 HADS 的患病程度与治疗费用之间的函数关系，帮助了解心理指标与治疗费用之间的关联，从而更好地理解问题的本质。考虑到 CBTS、EPDS 和 HADS 都是产妇心理指标，本文合理推测这三者之间存在较强的关联性。

然而，因为不同心理指标的费用可能存在差异，需要进行合理的费用分配。为了得到最小费用，可以尝试采用一系列规划方法，将 CBTS、EPDS 和 HADS 的函数关系拟合成一个总方程式，使得该模型更符合实际情况，并考虑不同指标的权重和影响。最终，通过这个总方程式，可以得到 238 号婴儿行为特征从矛盾型转为中等型和安静型的最小费用。

2.4 问题四的分析

根据题意，需要建立婴儿综合睡眠质量与母亲的身体指标、心理指标的关联模型。已知综合睡眠质量由整晚睡眠时间、睡醒次数和入睡方式三个指标组成。本文考虑采取主成分分析或者因子分析法，将这三个指标整合为一个综合指标，减少指标之间的相关性，得到更为简化和综合的睡眠质量指标。之后再并将其降序分为四份等级：差、中、良和优，帮助更好地理解婴儿的综合睡眠质量水平。

由于问题四的本质类似于问题二，只有婴儿的综合睡眠质量需将三个指标进行综合考虑，接下来的处理步骤可以同理问题二中的处理方法：即将数据进行热编码，使用最优模型进行建模和调优，并最终预测题目给出的 20 组数据的综合睡眠质量，从而得到问题四的答案。

2.5 问题五的分析

根据题意，需要建立产妇心理指标和婴儿的综合睡眠质量的关系模型，并在此基础上分析对 238 号婴儿的综合睡眠质量调为优的治疗策略与第三问是否发生变化，调整之后新的策略是什么。

现已求得婴儿综合睡眠质量的评价标准，可直接得出 238 号婴儿目前的综合睡眠质量等级。为帮助观察指标之间的分布情况和异常值，从而初步了解两者之间的关联，本文将使用箱线图等可视化手段分析产妇心理指标与婴儿综合睡眠质量之间的关系。然后，本文将考虑使用偏最小二乘法或其他合适的拟合方法，来建立产妇心理指标与婴儿综合睡眠质量之间的关系模型。由于睡眠指数是离散定类，不利于模型的建立，本文考虑使用婴儿综合睡眠质量评价标准作为定量指标来进行分析。在分析出关系模型后，根据 238 号婴儿目前的综合睡眠质量等级，结合模型，制定相应的调整策略。

三．定类变量数值含义

自变量	自变量赋值
婚姻状况	1=未婚，2=已婚
教育程度	1=小学，2=初中，3=高中，4=大学，5=研究生
分娩方式	1=自然分娩，2=剖宫产

婴儿性别	1=男性，2=女性
入睡方式	1=哄睡法，2=抚触法，3=安抚奶嘴法，4=环境营造法，5=定时法
行为特征	0=安静型，1=中等型，2=矛盾型
睡眠质量	0=差，1=中，2=良，3=优

四．模型的建立和求解

4.1 问题一模型的建立与求解

4.1.1 数据预处理

根据已给变量并结合变量数值的含义，可得出数据的隐含条件：

(1) 母亲的婚姻状况只有未婚和已婚俩个选项，即数据中只可能存在数字 1 和 2；

(2) 母亲的教育程度只有小学、初中、高中、大学和研究生这五个选项，即数据中只可能存在 1 到 5 这五个数字；

(3) 母亲的分娩方式只有自然分娩和剖宫产俩个选项,即数据中只可能存在数字 1 和 2；

(4) 产妇心理指标 CBTS、EPDS 和 HADS 非负且小于等于 30；

(5) 婴儿的性别只有男性和女性，即数据中只可能存在数字 1 和 2；

(6) 婴儿的整晚睡眠时间要满足时间的表达式；

(7) 婴儿的入睡方式只有哄睡法、抚触法、安抚奶嘴法、环境营造法和定时法这五个选项，即数据中只可能存在 1 到 5 这五个数字；

(8) 母亲的年龄和妊娠时间，以及婴儿的年龄和睡醒次数要求非负。

在程序中，使用编程语言中的条件判断和筛选功能列出以上条件进行初步筛选，将符合条件的数据从 data1 中保存。经观察发现，婴儿的行为特征和整晚睡眠时间这俩项的数据不是具体的数字，为了方便后续进行定量的分析，本文将婴儿的行为特征类似于其他定类数据进行从 0 开始的标号，将婴儿的整晚睡眠时间由表达式改为小数形式。将调整之后的数据保存。

4.1.2 定性研究

数据初步可视化

为了直观感受母亲的身体指标和心理指标对婴儿的行为特征和睡眠质量的影响，本文将经过预处理后的数据进行了初步可视化，得以下图 1 和图 2：

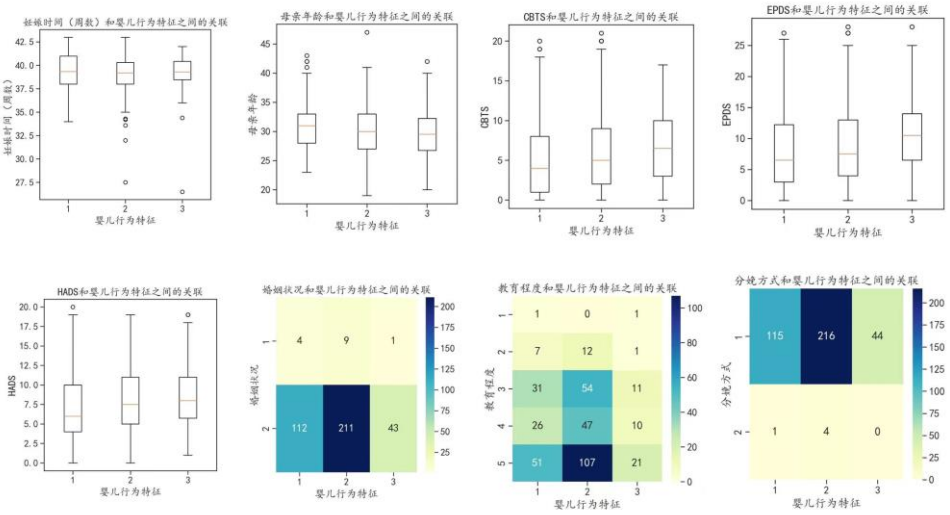


图 1 母亲身心健康与婴儿行为特征之间的关联

从图 1 可以看出，母亲的生理指标包括妊娠时间、年龄、婚姻状况、教育程度和分娩方式与婴儿的行为特征无直接上明显的关联。特别地，由于**婚姻状况为未婚的、分娩方式为剖宫产**的数据**占比过小**，如果扩大数据量以及这俩种情况出现的占比，可能会对婴儿的成长产生影响；在教育程度中上，婴儿行为特征在中等型的占比较高。

另一方面，母亲的**心理指标**与婴儿的行为特征之间**关联性较强**。CBTS、EPDS 和 HADS 的得分越高，即母亲的压力越大，婴儿的情绪反应越强烈。这意味着母亲心理的亚健康状态可能会对婴儿的行为特征产生较为负面的影响。

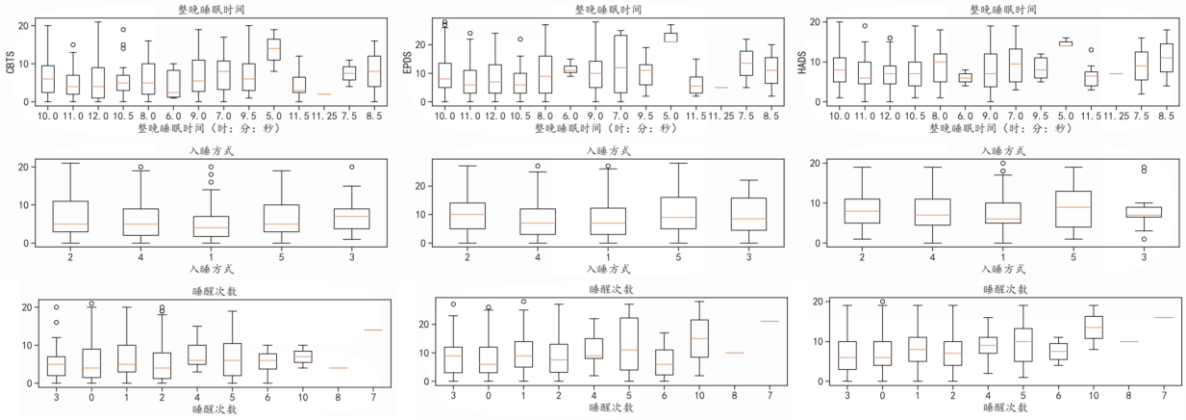


图 2 母亲心理健康与婴儿睡眠质量之间的关联

在这里，将母亲的生理指标与婴儿的睡眠质量的分析结果放在附录 A 中，这些图表可以提供更全面的数据情况，但由于不直接关联主题，因此将其放在附录中以突出更为重要的内容。从图 2 中可以看出，随着 CBTS、EPDS 和 HADS 指标的分数增加，即母亲的压力越大、抑郁焦虑的程度越深，婴儿的睡眠时间和睡醒次数整体上呈递增趋势。这一发现直观地说明**母亲心理的亚健康**会对婴儿的**睡眠质量**造成较为**负面的影响**。

4.1.3 定量分析

4.1.3.1 Pearson 相关系数

根据公式：

$$r=\frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum(X-\bar{X})^2\sum(Y-\bar{Y})^2}}=\frac{l_{XY}}{\sqrt{l_{XXlYY}}}$$

$$l_{XX}=(X-\bar{X})^2=\sum X^2-\frac{(\sum X)^2}{n}$$

$$l_{YY}=(Y-\bar{Y})^2=\sum Y^2-\frac{(\sum Y)^2}{n}$$

$$l_{XY}=\sum(X-\bar{X})(Y-\bar{Y})=\sum XY-\frac{(\sum X)(\sum Y)}{n}$$

其中，样本的相关系数用符号 r 表示，l_{XX}表示 X 的离均差平方和，l_{YY}表示 Y 的离均差平方和，l_{XY}表示 X 与 Y 的离均差平方和。

在本文中，使用 pandas 中的 corr()函数计算了 Pearson 相关系数，并绘制了相应的热力图（图 3）。通过这个过程，可以计算出不同变量之间的 Pearson 相关系数，并使用热力图直观地展示它们之间的关系。

Pearson 相关系数是用来衡量两个连续性变量之间线性相关程度的统计指标。它的取值范围在-1 到 1 之间，其中 1 表示完全正相关，-1 表示完全负相关，0 表示没有线性

相关。通过分析相关系数的取值，可以初步了解不同变量之间的关联程度。

然而，值得注意的是，Pearson 相关系数适用于两个变量都是**连续性变量**的场景。在本文中，虽然 corr()函数将非连续变量之间的相关性也进行了分析，并使用热力图展示出来，但对于这部分的数据可以暂时进行忽略，以确保分析结果的准确性和可靠性，在后续分析中采取更为合适的方法。

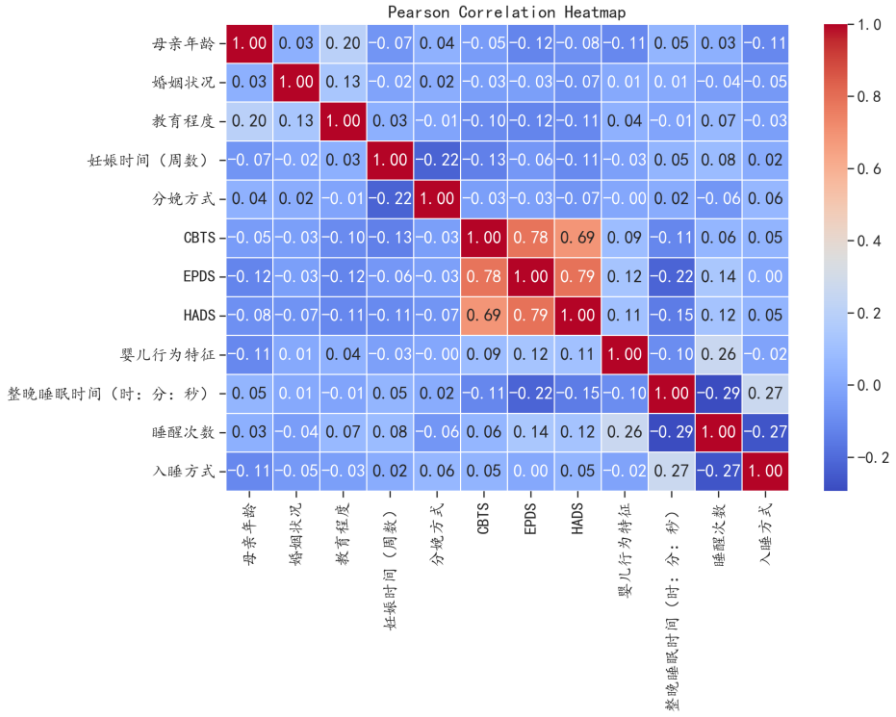


图 3 Pearson 相关系数热力图

从图 3 中可以进行连续性变量之间的关系初步分析：

整晚睡眠时间与母亲的心理指标CBTS、EPDS 和HADS之间存在较显著的负相关性。这意味着随着母亲心理压力和抑郁焦虑程度的增加，婴儿的整晚睡眠时间可能会减少。

整晚睡眠时间与母亲年龄之间存在较小的正相关。这表明母亲的年龄对婴儿的整晚睡眠时间影响较小。然而，尽管相关性较小，年龄仍然可以在整体分析中考虑。

整晚睡眠时间与妊娠时间几乎没有明显的关系。这意味着妊娠时间对婴儿的整晚睡眠时间影响较小。

4.1.3.2 Spearman 相关系数

为了分析**定类变量之间**或**定量变量之间**的关联性，本文选择使用 Spearman 相关系数。Spearman 相关系数是一种非参数统计方法，它将变量 X 和 Y 分别从小到大排序编秩，并用秩次 RX 和 RY 表示。当出现数据相等从而造成秩次相同的现象时，称为相持(tie)，在计算过程中，取其平均秩次作为每个数据的秩次。Spearman 相关系数 rs 的计算公式为：

$$r_s = \frac{\sum(R_X - \overline{R_X})(R_Y - R_Y)}{\sqrt{\sum(R_X - \overline{R_X})^2 \sum(R_Y - \overline{R_Y})^2}} = \frac{\sum R_X R_Y - \frac{(\sum R_X)(\sum R_Y)}{n}}{\sqrt{\left(\sum R_X^2 - \frac{(\sum R_X)^2}{n}\right)\left(\sum R_Y^2 - \frac{(\sum R_Y)^2}{n}\right)}}$$

Spearman 相关系数 rs 的计算公式与 Pearson 相关系数的计算公式相似，只是将

Pearson 相关系数公式中的 X、Y 替换为了 RX、RY。

在本文中，采取使用 Python 中的 spearman()函数进行 Spearman 相关系数的计算，并绘制相应的热力图（图 4）。通过这个过程，可以直观地展示定类变量或定量变量之间的关联性。

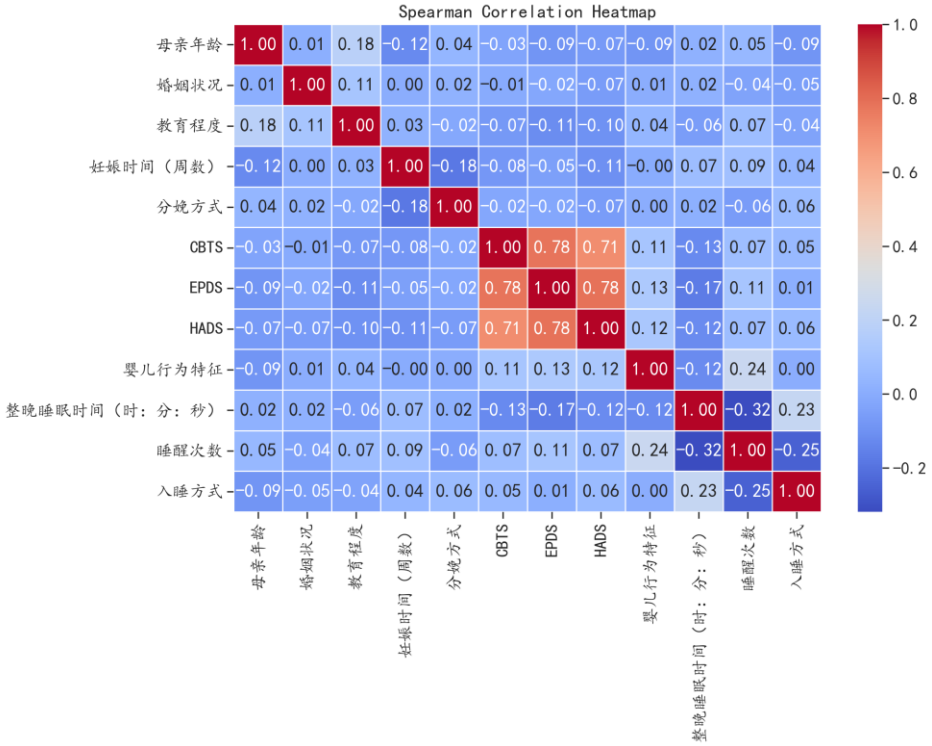


图 4 spearman 相关系数热力图

从图 4 中可以进行离散型变量之间的关系初步分析：婴儿行为特征与入睡方式、睡醒次数以及母亲的婚姻状况、教育程度和分娩方式之间的关联性较弱。这一发现间接证明了母亲的生理指标对婴儿的行为特征和睡眠质量的影响较小。

4.1.3.3 Kendall's tau-b 相关系数τ_b

在本文中，采取使了 Kendall's tau-b 相关系数来进行**定类和定量之间**的相关性分析。Kendall's tau-b 相关系数是一种非参数统计方法，适用于分析有序分类数据之间的关联性。它的计算公式如下：

$$\tau_b = \frac{C - D}{\sqrt{T - T_r}\sqrt{T - T_c}}$$

其中，C 是指两两比较对中协和对的个数，D 是指两两比较对中不协和对的个数。T 是指两两比较的总对数，为 n(n-1)/2，其中 n 为样本量。Tr 是指不变对中，X 值不变的个数，Tc 是指不变对中，Y 值不变的个数。

本文将 Kendall's tau-b 相关系数的计算公式写入程序，并绘制了相应的热力图（图 5）。通过这个过程，可以直观地展示定类变量和定量变量之间的关联性。

从图中可以进行离散型和连续变量之间的关系分析：

婴儿的行为特征与母亲的心理指标 CBTS、EPDS 和 HADS 之间存在较显著的正相关性。这意味着随着母亲心理压力和抑郁焦虑程度的增加，婴儿的行为特征可能会表现得更加突出或明显。

婴儿的行为特征与母亲年龄之间存在较小的正相关。这表明母亲的年龄对婴儿的行为特征影响较小。然而，尽管相关性较小，年龄仍然可以在整体分析中考虑。

婴儿的行为特征与妊娠时间几乎没有明显的关系。这意味着妊娠时间对婴儿的行为特征影响较小。

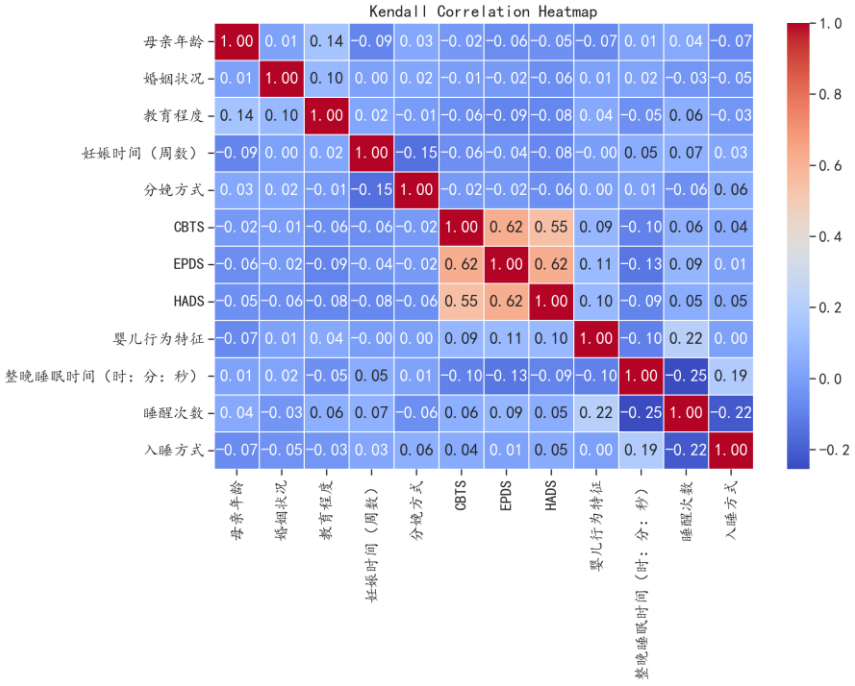


图 5 kendall 相关系数热力图

从三种相关性分析综合来看，婴儿的行为特征与教育程度和产妇心理指标呈正相关，与母亲的年龄呈负相关。婴儿的整晚睡眠时间，入睡与妊娠时间呈正相关，与产妇心理指标呈负相关。婴儿的睡醒次数与母亲的年龄、教育程度、妊娠时间和产妇心理指标呈正相关，与婚姻状况和分娩方式呈负相关。但是妊娠时间，婚姻状况，分娩方式，教育程度与婴儿行为特征和睡眠质量相关性几乎为 0。

4.1.4 总结

根据上述分析和计算，本文得出了重要的结论：母亲的**心理指标**对婴儿的行为特征和睡眠质量的影响较大，而且母亲心理抑郁焦虑程度越深，对婴儿的成长产生的负面影响越明显。与此同时，母亲的身体指标对婴儿的行为特征和睡眠质量也有一定程度的影响，但相对较小。

经翻阅文献，发现许多国内外学者也认为母亲的抑郁、焦虑情绪与婴儿的睡眠问题有密切关联[1]。这种影响可能是因为母亲的抑郁、焦虑情绪会影响亲子交流的质量和积极性，从而降低婴儿的安全感。同时，母亲的心理状况也会影响她对婴儿睡眠行为的敏感性和积极性[2]。为更好地促进婴儿的健康成长，应重视产妇心理健康的问题，提供

必要的支持和帮助。

需要指出的是，本研究存在一些局限性。首先，在样本的选择和数据采集可能存在一定的偏差。其次，本文只考虑了母亲的身体指标和心理指标对婴儿的影响，而其他因素可能也会对婴儿成长产生影响。在未来的研究中可以进一步拓展样本规模和纳入更多因素，以得到更加全面和准确的结论。

4.2 问题二模型的建立与求解

4.2.1 数据预处理

因为变量中含有一些定类变量，为了提高模型的精确度，本文将对数据采取两种不同的特征对他们处理方式，一种是目标编码，另一种是热编码。然后使用 XGBoost 分类器进行训练和预测，并输出分类报告。另外，本文还计算了直接分类的准确率并输出分类报告。从而分析哪一种方式准确率更高。

4.2.1.1 目标编码：

本文使用 ce.TargetEncoder 将分类特征：婚姻状况、教育程度和分娩方式进行目标编码。然后将数据集拆分为训练集和测试集。接着初始化 XGBoost 分类器，将模型训练在训练数据上，并在测试数据上进行预测。最后输出目标编码分类报告（图 6）。

目标编码分类报告：

	precision	recall	f1-score	support
0	0.36	0.33	0.35	24
1	0.53	0.67	0.59	36
2	0.00	0.00	0.00	12
accuracy			0.44	72
macro avg	0.30	0.33	0.31	72
weighted avg	0.39	0.44	0.41	72

图 6 目标编码分类报告

4.2.1.2 热编码：

本文使用 pd.get_dummies()对分类特征：婚姻状况、教育程度和分娩方式进行热编码。然后将数据集拆分为训练集和测试集。接着初始化 XGBoost 分类器，将模型拟合在训练数据上，并在测试数据上进行预测。最后输出热编码分类报告（图 7）。

热编码分类报告：

	precision	recall	f1-score	support
0	0.36	0.19	0.25	26
1	0.54	0.80	0.65	40
2	0.33	0.10	0.15	10
accuracy			0.50	76
macro avg	0.41	0.36	0.35	76
weighted avg	0.45	0.50	0.45	76

图 7 热编码分类报告

4.2.1.3 直接分类：

这部分与前两部分类似，不过在测试数据上进行预测后，计算了直接分类的准确率，并输出直接分类报告（图 8）。

直接分类报告：

	precision	recall	f1-score	support
0	0.36	0.19	0.25	26
1	0.52	0.78	0.62	40
2	0.00	0.00	0.00	10
accuracy			0.47	76
macro avg	0.29	0.32	0.29	76
weighted avg	0.39	0.47	0.41	76

图 8 直接分类报告

根据报告展示的三种不同特征处理方式的模型性能，发现使用热编码可以避免将分类变量误解为连续值，消除分类之间的顺序关系，帮助算法更好地理解特征之间的独立性，从而提高预测的准确性。所以，在之后的模型建立时使用的数据都是经过热编码处理之后的数据。

4.2.1 模型对比

为了提高结果的精确度，本文初步筛选出适用于该场景使用的六个模型：XGBoost 模型、决策树模型、随机森林模型、LightGBM 模型、BP 神经网络模型和 SVM 模型。对分类特征：婚姻状况、教育程度和分娩方式进行热编码，并将数据集拆分为训练集和测试集的基础上进行模型拟合和预测。同时，绘制出目标变量类别分布、混淆矩阵（热图）、特征重要性条形图，并输出相应的分类报告。本文将其他不重要数据图暂放于附录 A，分类报告稍作加工如表 1 所示：

表 1 模型数据对比

	accuracy	macro avg	weighted avg
XGboost	0.5	0.35	0.45
决策树	0.42	0.35	0.42
随机森林	0.54	0.32	0.46
LightGbm	0.47	0.33	0.43
DP	0.46	0.32	0.41
SVM	0.53	0.23	0.36

根据数据显示，使用随机森林模型的精确度更高。

4.2.2 模型建立

在本文中，使用了 Python 中现有的随机森林库进行分类任务。随机森林是一种集成学习算法，它由多个决策树组成，通过对每个决策树的预测结果进行投票或平均来得到最终的分类结果。其表达式如下：

$$y = \frac{1}{k}[t_1 \cdot \mathrm{prob}(x) + t_2 \cdot \mathrm{prob}(x) + \cdots + t_k \cdot \mathrm{prob}(x)]$$

其中，t_i表示决策树，$t_i \cdot \mathrm{prob}(x)$表示第 i 棵树对 x 的预测，输出为各个类别的预测概率（行向量），k 表示森林的规模。自变量 x 为母亲的年龄、婚姻状况、教育程度、妊娠时间、分娩方式、CBTS、EPDS 和 HADS 这些特征。这些特征用来预测婴儿的行为特征。因变量 y 表示婴儿的行为特征，即本文希望通过这些自变量预测出婴儿的行为特征的分类结果。

4.2.3 参数调优

为了对新数据进行准确的预测，提供更可靠的模型性能评估和预测结果，本文使用了网格搜索来对随机森林模型进行参数调优，以找到最佳的参数组合，进而提高模型的性能。

首先，在 param_grid 中指定了一组候选参数，这些参数包括随机森林的树的数量、每棵树的最大深度、每个节点最少样本数等。然后，使用 GridSearchCV 对随机森林模型进行网格搜索，以找到最佳的参数组合。在网格搜索过程中，通过交叉验证（cv=3），对每种参数组合进行模型训练和评估。交叉验证将训练数据分成若干份，每次取其中一份作为验证集，其余作为训练集，然后对模型进行训练和评估。这样可以避免过拟合，并且能够更准确地评估模型的性能。

为了加快训练过程，本文置 n_jobs=-1，表示使用所有可用的 CPU 核心来并行处理网格搜索。最终，网格搜索会输出最佳的参数组合和对应的最佳精度。接下来，使用网格搜索得到的最佳模型对测试数据进行预测，并输出分类报告。分类报告包含了模型在测试集上的精确度、召回率、F1-score 等分类指标，这些指标可以帮助评估模型的性能。

然后，本文将使用之前调优过的最佳模型对新数据进行预测。预测结果会被添加到新数据中。最后，保存新数据的婴儿行为特征的预测结果，从而得到模型对新数据的预测结果（表 2）。

表 2 问题二预测结果

母亲年龄	婚姻状况	教育程度	妊娠时间（周数）	分娩方式	CBTS	EPDS	HADS	婴儿行为特征
29	2	4	40	1	7	15	12	安静型
29	2	3	42	1	9	14	12	安静型
23	2	2	38.5	1	7	12	7	安静型
27	2	3	36.3	1	8	4	5	安静型
36	2	4	39	1	6	6	8	安静型
30	2	5	41.2	1	5	8	5	安静型
28	2	2	40.6	1	8	11	9	安静型
32	2	5	37	1	3	6	7	安静型
28	2	5	38	1	7	11	5	安静型
31	2	4	42	1	4	5	8	安静型
25	2	2	40.5	1	16	22	15	安静型
27	2	5	40.4	1	4	6	10	安静型
33	2	5	39	1	6	6	4	安静型
25	2	3	39	1	0	4	5	安静型
28	2	2	41	1	9	6	5	安静型
31	2	3	39.5	1	1	4	4	中等型
26	2	2	37	1	4	9	14	安静型
26	2	5	39	1	0	3	3	安静型
27	2	5	41.2	1	0	0	4	中等型

31	2	5	38	1	3	7	7	安静型

4.3 问题三模型的建立与求解

4.3.1 求解初步函数关系

根据题意——“CBTS、EPDS 和 HADS 的治疗费用相对于患病程度的变化率均与治疗费用呈正比”，可以推断出最终的总方程式一定是指数形式的函数，且根据$\frac{d_y}{d_x}=ky$得出该函数应是形如$y=ce^{k}$的微分方程。

依据这一结论进行数学建模：

令$x=[x_1,x_2,x_3]$ 为患病程度；

$y=[y_1,y_2,y_3]$ 为治疗费；

$D=[A,B,C]$ 为患者对应的治疗费；

$\frac{dy}{dx}$ 为治疗费相对于患病程度的变化率。

解方程组$\int \frac{1}{y}dy=\int k\cdot dx$，得到 $y=ce^{kx}$。

将题目中表格 1 的数据进行代入：

$$\begin{bmatrix}x_1\\y_1\end{bmatrix} \text{ 代入 } \begin{bmatrix}0\\200\end{bmatrix},\begin{bmatrix}3\\2812\end{bmatrix}$$
$$\begin{bmatrix}x_2\\y_2\end{bmatrix} \text{ 代入 } \begin{bmatrix}0\\500\end{bmatrix},\begin{bmatrix}2\\1890\end{bmatrix}$$
$$\begin{bmatrix}x_3\\y_3\end{bmatrix} \text{ 代入 } \begin{bmatrix}0\\5\end{bmatrix},\begin{bmatrix}300\\12500\end{bmatrix}$$

得到$\begin{bmatrix}C\\k\end{bmatrix}=\begin{bmatrix}200,500,300\\0.88,0.66,0.75\end{bmatrix}$，即患者医疗费用方程式为：

$$\begin{cases}w=(A-y_1)+(B-y_2)+(c-y_3)\\y_1=200e^{0.88x}\\y_2=500e^{0.66x}\\y_3=300e^{0.75x}\end{cases}$$

设线性回归相关系数$Q=[a、b、c]$，截距为D，则预测结果为$Z=QX^{T}+D$。

最终得到目标函数：

$$\min W=200e^{0.88*15}-200e^{0.88*x_1}+500e^{0.66*22}-500e^{0.66*x_2}+300e^{0.75*18}-300e^{0.75*x_3}$$

4.3.2 建立约束条件

4.3.2.1 逻辑回归模型预测

逻辑回归模型预测中得出以下表格：

表 3 逻辑回归模型预测

似然比卡方值	P	AIC	BIC
697.558	0.243	713.558	745.079

从表格中可以看出，显著性 P 值为 0.243，水平上不呈现显著性，不能拒绝原假设，因而模型是无效的，并且三个指标应该都为正相关，而 CBTS，HADS 均为负相关，与之前的论证相违背，所以本文将考虑使用其他预测模型。

4.3.2.2 偏最小二乘模型

偏最小二乘预测中得出以下表格：

表 4 偏最小二乘模型预测

常数	CBTS	EPDS	HADS
0.0001	0.706	0.262	0.607

从表格数据可以看出，显著性 P 值水平上不呈现显著性, 不能拒绝回归系数为 0 的原假设。

表 5 偏最小二乘模型预测

	常数	CBTS	EPDS	HADS
婴儿行为特征	0.691	-0.004	0.01	0.006
婴儿行为特征(标准化)	0	-0.032	0.111	0.044

从表格数据中可以得出模型公式为： 婴儿行为特征 = 0.691-0.004 * CBTS+0.01 * EPDS+0.006 * HADS。但是 CBTS 依旧为负相关，模型无效。

4.3.2.3 均值、中位数、众数

由于上述俩个模型都无法解决实际的问题，本文考虑从均值、中位数和众数出发，

从中选择一个能够较好反应某个婴儿行为特征指标的量作为依据。首先，绘制出三者的关系图，如图 9 所示：

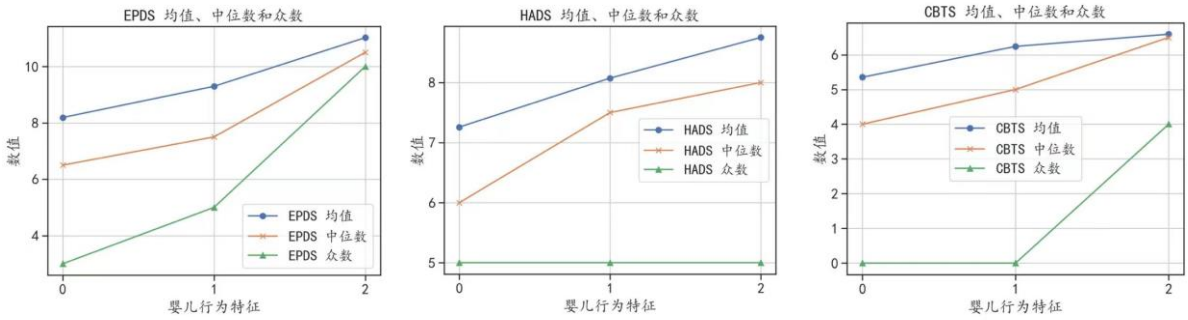


图 9 孕妇心理指标的均值、中位数和众数

从图中看出，中位数最能体现婴儿行为特征的标准。于是本文将根据图 10 来判断约束条件：

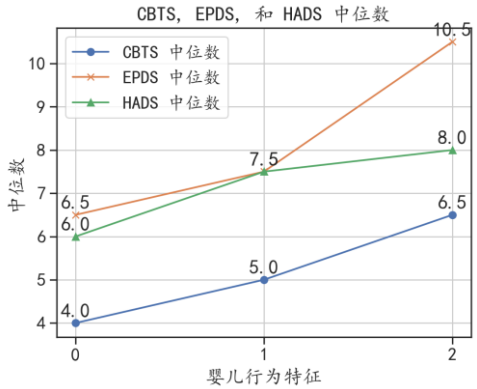


图 10 孕妇心理指标的中位数

从图中可以看出当婴儿的行为特征为中等型时，三者的指标分别为 5.0，7.5，7.5；安静型时为 4.0，6.5，6.0。因此，可以得出两种约束条件：

$$\text{S.T.}\begin{cases} 15-x_1 \le 4 \\ 22-x_2 \le 6.5 \\ 18-x_3 \le 6 \\ x_1,x_2,x_3 \in [0,30] \\ x_1,x_2,x_3 \in \text{int} \end{cases} \qquad \text{S.T.}\begin{cases} 15-x_1 \le 5 \\ 22-x_2 \le 7.5 \\ 18-x_3 \le 7.5 \\ x_1,x_2,x_3 \in [0,30] \\ x_1,x_2,x_3 \in \text{int} \end{cases}$$

即总目标方程为：

$$\min W = 200e^{0.88*15} - 200e^{0.88*1} + 500e^{0.66*22} - 500e^{0.66*2} + 300e^{0.75*18} - 300e^{0.75*3}$$

4.3.3 结果

综上所述，本题将使用 python 的 scipy 库求得最后结果为：

变成中等型，CBTS、EPDS、HADS 分别下降 10，15，11 总费用为 46896.7。

变成安静型，CBTS、EPDS、HADS 分别下降 11，16，12 总费用为 50897.36

4.4 问题四模型的建立与求解

4.4.1 综合睡眠质量

4.4.1.1 绘制相关性矩阵图和三维曲面图

本文使用 Seaborn 库绘制整晚睡眠时间、睡醒次数与入睡方式这三个要素的相关性矩阵图，通过相关性矩阵图可以了解数据中特征之间的相关性程度；同时使用 Matplotlib 和 mpl_toolkits 库绘制其三维曲面图，通过三维曲面图可以直观地观察三个特征之间的关系。如图 11 所示：

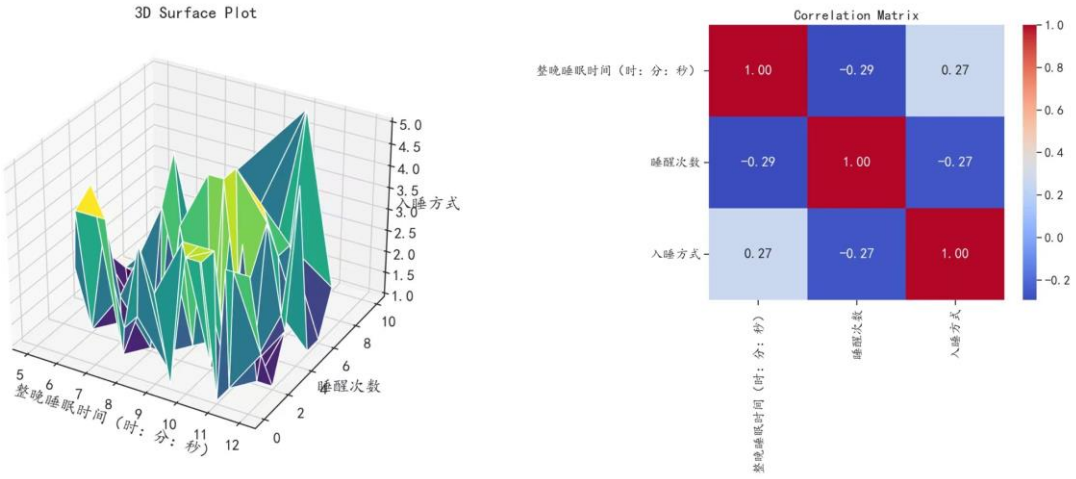


图 11 三要素相关性矩阵图与三维曲面图

从图中可以看出，这三个要素之间存在俩个正相关、一个负相关：睡醒次数与其他俩要素之间呈负相关，整晚睡眠时间和入睡方式之间呈正相关。

4.4.1.2 因子分析法

本文在采取因子分析法分析之前，也考虑过使用主成分分析（PCA）。查找相关资料发现：如果目标是降维和去除冗余特征，且关注的是数据的主要变异，那么主成分分析是一个较好的选择；而如果目标是探索数据背后的潜在结构和因素，或者变量之间存在复杂的因果关系，那么因子分析法可能更适合。根据题目要求，降维的目的更侧重于探讨三要素背后复杂的因果关系，所以采用因子分析法更为合适。

因子载荷矩阵：

因子载荷矩阵是因子分析的核心结果之一，它显示了原始变量与提取的潜在因子之间的相关性。可以通过观察载荷矩阵来理解原始变量与潜在因子之间的关系以及每个变量对于潜在因子的重要性。

首先，本文从原始数据（data3)中选择三列数据：整晚睡眠时间、睡醒次数和入睡方式用于因子分析。然后，创建一个因子分析模型并指定因子数为 1，即希望提取一个主要的潜在因子。接着，使用拟合方法对选择的数据进行因子分析，以提取潜在因子。其次，获取因子载荷矩阵（成分矩阵），显示了原始变量与潜在因子之间的关系强度。最后，打印出因子载荷矩阵，显示了每个原始变量对于提取的潜在因子的贡献程度，如表 6 所示：

表 6 因子载荷矩阵（成分矩阵）表

整晚睡眠时间	-0.540604
睡醒次数	0.543433
入睡方式	-0.490766

计算评分标准新列：

根据因子载荷矩阵的结果，可以列出一个评分标准的计算公式，用于加权计算婴儿的睡眠质量，计算公式如下：

综合睡眠质量 = 0.54 * 整晚睡眠时间 - 0.54 * 睡醒次数 + 0.49 * 入睡方式

本文使用给定的计算式对每 data2 中的每一行数据进行计算，并将结果进行四舍五入处理并存储在新的名为评分标准的列中。将更新后的数据保存为名为 data7 的新 CSV 文件中。

4.4.1.3 分类处理

KMeans 聚类分析

本文将使用 KMeans 聚类算法对数据进行聚类分析，并可视化聚类结果。以下是具体的实现方法：

首先，本文将 data7 中评分标准这一列作为聚类的数据，并使用 values.reshape(-1, 1)对数据进行格式转换，以便适应 KMeans 算法。接着，使用 KMeans 算法进行聚类分析，设置聚类个数为 4，将聚类结果存储在'睡眠质量'中并添加到 DataFrame 中作为一个新的'睡眠质量'列，然后将包含聚类结果的保存。其次，使用散点图将聚类结果可视化，横轴为数据的索引，纵轴为评分标准，不同聚类结果使用不同颜色进行区分，如图 12 所示。最后，使用 groupby 对聚类结果进行汇总，计算每个聚类中的数据个数、平均值和标准差，并打印结果。

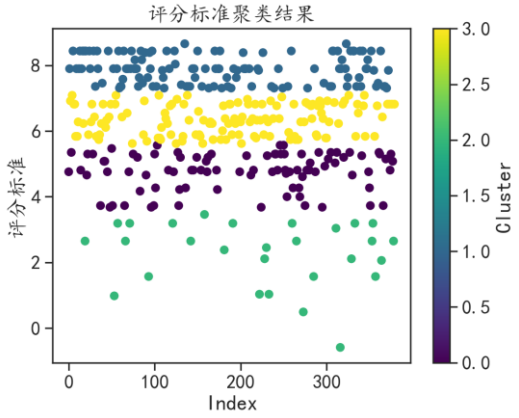


图 12 评分标准聚类结果

重新排序和赋值

为了更好地给综合睡眠质量划分层次，本文对聚类结果进行了重新排序和重新赋值，使得聚类标签 0 对应的平均值最小，聚类标签 3 对应的平均值最大。以下是具体操作流程：

首先，本文使用 groupby 对聚类结果进行汇总，计算每个聚类中的数据个数、平均值和标准差，并将结果存储。然后，根据聚类标签的平均值对聚类进行升序排序，得到排序后的聚类标签，并存储在 sorted_clusters 中。接着，使用 map 函数，将新的聚类标签重新赋值给 DataFrame 中的 Cluster 列，使得聚类标签 0 对应的平均值最小，聚类标签 3 对应的平均值最大。最后，重新计算聚类汇总并输出重新赋值后的聚类汇总，按照聚类标签顺序进行分组，输出结果如表 7 所示：

表 7 重新赋值后地聚类汇总

Cluster	count	mean	std
0	28	2.267500	0.990644

1	91	4.782418	0.552536
2	134	6.359179	0.451750
3	127	7.885748	0.424463

4.4.2 使用随机森林模型进行分类

类似于问题二中求解婴儿的行为特征与母亲的身体指标与心理指标的关系模型。在本问题中，自变量仍是母亲的年龄、婚姻状况、教育程度、妊娠时间、分娩方式和产妇心理指标，将原本的因变量——婴儿的行为特征换为婴儿的睡眠质量，并对婚姻状况、教育程度、分娩方式进行热编码处理，将分类变量转换成哑变量。

同样的操作过程如下：

首先，使用 train_test_split 函数将数据划分为训练集和测试集，测试集占比为80%。接着，初始化一个随机森林分类器模型进行拟合然后预测。再然后，绘制出目标变量睡眠质量的类别分布情况。其次，计算预测结果与实际标签的混淆矩阵，并使用热图形式绘制，方便查看预测结果的正确与错误情况。最后输出分类报告，并根据随机森林模型学习到的特征重要性，绘制特征重要性的条形图，显示哪些特征对模型预测最重要。输出结果如下图 13 所示：

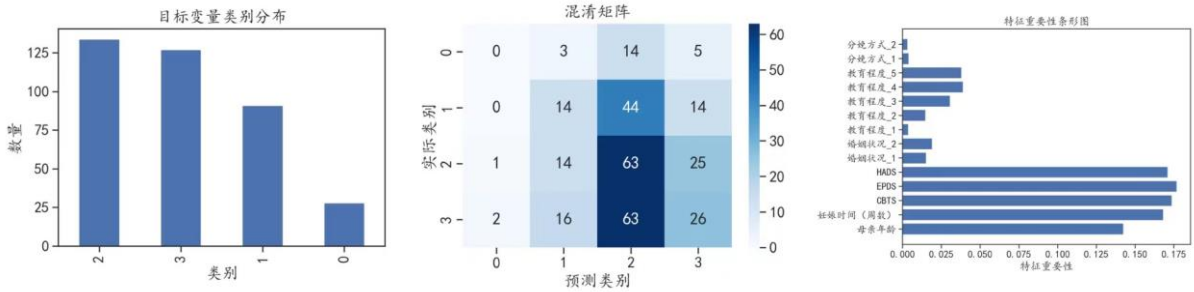


图 13 输出结果

同理，在本题中依旧选择使用网格搜索来对随机森林模型进行超参数调优（图 14）。进行调优后，选择表现最好的参数组合的模型对题目所给出的 20 组数据进行预测，并将预测结果添加到原数据中的最后一列。

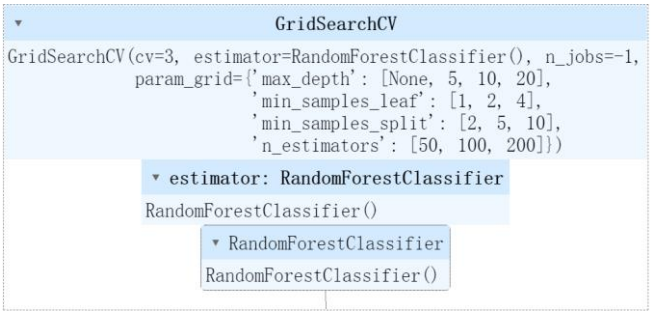


图 14 参数调优

4.4.3 结果

至此，可以得出问题四的答案，如表 8 所示：

表 8 问题四预测结果

母亲年龄	婚姻状况	教育程度	妊娠时间（周数）	分娩方式	CBTS	EPDS	HADS	综合睡眠质量

29	2	4	40	1	7	15	12	良
29	2	3	42	1	9	14	12	优
23	2	2	38.5	1	7	12	7	中
27	2	3	36.3	1	8	4	5	优
36	2	4	39	1	6	6	8	良
30	2	5	41.2	1	5	8	5	良
28	2	2	40.6	1	8	11	9	优
32	2	5	37	1	3	6	7	优
28	2	5	38	1	7	11	5	中
31	2	4	42	1	4	5	8	优
25	2	2	40.5	1	16	22	15	良
27	2	5	40.4	1	4	6	10	良
33	2	5	39	1	6	6	4	良
25	2	3	39	1	0	4	5	优
28	2	2	41	1	9	6	5	优
31	2	3	39.5	1	1	4	4	优
26	2	2	37	1	4	9	14	良
26	2	5	39	1	0	3	3	优
27	2	5	41.2	1	0	0	4	优
31	2	5	38	1	3	7	7	优

4.5 问题五模型的建立与求解

4.5.1 数据初步可视化

本文定义了一个函数 caseplot(name)，用于绘制特定变量与睡眠质量之间的箱线图，从而帮助观察不同睡眠质量类别下特定变量的分布情况，从而了解这些变量与睡眠质量之间是否存在关联，如下图 15 所示：

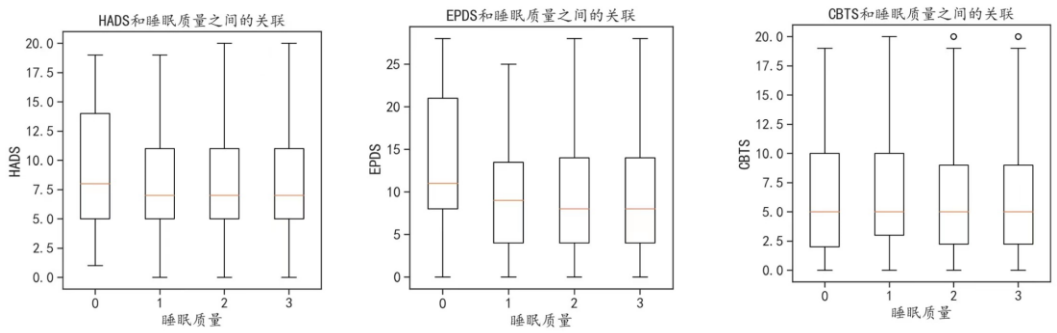


图 15 孕妇心理指标与睡眠质量之间的关联

由图可知：EPDS 和 CBTS 与睡眠质量有一定的负相关。但是 CBTS 非常不明显，几乎没有关联。

4.5.2 数据关联性分析

本文对上面的初步结论使用热力图进行定量分析，分析结果如图 16。

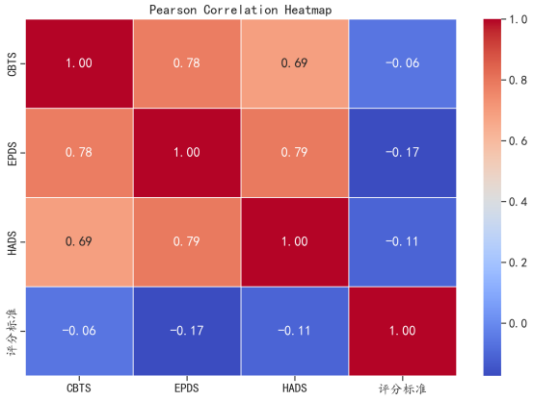


图 16 数据关联性分析

由热力图可知，CBTS 负相关并很不显著，也证实了之前初步的分析。

4.5.3 模型建立

本体使用偏最小二乘-判别分析（PLS-DA）建立模型，将自变量 CBTS、EPDS 和 HADS 与因变量评分标准进行建模。具体实现过程如下：

首先，本文将自变量 X 设置为 CBTS、EPDS 和 HADS 列的值，将因变量 y 设置为评分标准列的值，并计算自变量的均值和标准差，对其进行标准化处理。然后，使用偏最小二乘-判别分析建立模型：创建一个 PLSRegression 模型，并通过 fit 方法对标准化后的自变量 X_normalized 和因变量 y 进行建模。最后，输出建模后的偏最小二乘-判别分析模型的回归系数和截距。

回归系数分别为：0.33810148、-0.54102068 和-0.01006329，截距为 6.19028947。此时发现，CBTS 的系数为正，因为 CBTS 相关性极弱，所以模型拟合出正系数现象属于合理范畴，接下来得到自变量为三个指标，因变量为睡眠评分标准的线性函数，和第三问同理设立目标函数和约束条件，然后进行规划问题求解。

至此，可以得出本体所要的数学规划模型如下：

$$\min W = 200e^{0.88*15} - 200e^{0.88*x_1} + 500e^{0.66*22} - 500e^{0.66*x_2} + 300e^{0.75*18} - 300e^{0.75*x_3}$$

设线性回归相关系数 $Q = [a、b、c]$ =$[0.34、 -0.54、 -0.016]$

截距为D=6.2

预测结果： $Z = QX^{\mathrm{T}} + D$

其中，x_1、x_2和x_3与问题三中所表示含义一致。

约束条件S.T. $\begin{cases} Z \geq 7.89 \\ x \in [0,30] \\ x \in \text{int} \end{cases}$

经查找，238 号婴儿的睡眠评分标准是 5.25，综合睡眠质量为 1，即中等水平。根据题意要求，要将 238 号婴儿的综合睡眠质量从中等水平提升到优秀水平。而根据问题四的分析，优秀的评分标准是 7.88±0.42。

4.5.4 求解方法

本文使用了 Python 中的 pulp 库[3]进行线性规划，规划目标是最小化所需治疗费用。具体实现过程如下：

首先，创建线性规划问题。接着，定义了 6 个决策变量：x_1、x_2、x_3并确定这些变量的取值范围。然后，将目标函数定义为 w，最后，引入约束条件求得该线性规划问题。

4.5.4 结果

至此，可以得出问题五的答案：CBTS、EPDS、HADS 分别下降 0，17，0 总费用为 12730.0

五．模型的检验

本文的问题二和四采用了随机森林模型，问题五采用了偏最小二乘模型，下面对模型进行进一步的分析。

5.1 随机森林的模型检验：

经测试，随机森林在第二问中的精确度如下：

表 9 随机森林精确度

	准确率	召回率	精确率	F1
训练集	0.857	0.857	0.883	0.842
测试集	0.632	0.632	0.749	0.532

第四问中的随机森林模型采用了交叉验证和数据洗牌，准确率如下：

表 10 随机森林精确度

	准确率	召回率	精确率	F1
训练集	0.836	0.836	0.843	0.834
交叉验证集	0.316	0.316	0.297	0.295
测试集	0.329	0.329	0.425	0.325

由图可知，使用随机森林在第二问中的准确率还算中等，但是在第四问的准确率不算太高。

5.2 偏最小二乘线性回归方程的检验

模型检验的输出结果为：

表 11 偏最小二乘线性回归显著性

R-squared	0.04360005954115442
MSE	2.728142075806079
RMSE	1.65170883505722
P-value	0.000999000999000999

检验的结果分析可以得到，显著性 P 值为 0.001，水平上呈现显著性，拒绝回归系数为 0 的原假设，说明模型可行。

本文通过预测值和真实值的散点图来定性分析模型的准确率：

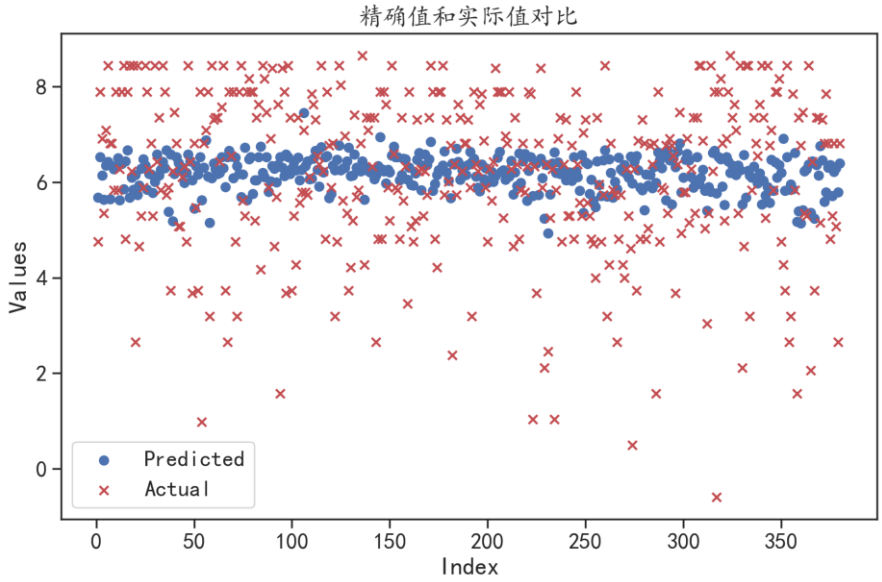


图 17 精确值和实际值对比

从图中可知，模型基本满足要求，但是预测值和实际值还是有一定差距。

六．模型的评价、改进与推广

6.1 模型的评价

6.1.1 模型的优点

高准确率：随机森林在处理复杂数据集时通常能够取得较高的准确率，它能够克服单个决策树容易过拟合的问题。

抗过拟合：随机森林采用了 Bagging（自助采样）的策略，通过随机选择数据子集来训练不同的决策树，再将它们集成，从而减少过拟合的风险。

处理大量特征：随机森林能够处理具有大量特征的数据集，并且在训练过程中可以自动选择重要特征，这使得它在处理高维数据时表现优秀。

对缺失值的鲁棒性：随机森林对于缺失值的处理相对较好，能够在保持准确性的同时处理包含缺失值的数据。

适合并行化处理：随机森林中的决策树可以独立并行地构建，这使得它在大规模数据集上的训练速度较快。

6.1.2 模型的缺点

预测解释性较差：随机森林由多个决策树组成，导致模型的预测过程相对复杂，不

如单个决策树那样容易解释。

内存消耗较大：由于随机森林包含多个决策树，因此它在内存占用方面相对较大，特别是在树的数量较多时。

训练时间较长：相比于简单的线性模型，随机森林需要构建多个决策树，因此在训练过程中会消耗更多时间。

不适合处理高度不平衡的数据：当数据集存在明显的类别不平衡问题时，随机森林的性能可能会下降，需要采取一些特殊的调整方法来处理。

模型的准确度表现并不出色:初步分析并非模型选择问题，而是数据集数据过小，数据与数据之间的关联度不够等等。但在实际应用中，由于数据的不完整性，模型的预测结果可能会受到一定程度的影响。

应用范围可能受到一定的限制。由于模型的特定结构和假设，该模型可能更适用于特定类型的问题，而在其他类型的问题上可能表现不佳。这需要在实际应用中进行进一步的验证和评估。

6.2 模型的改进

数据预处理：确保对数据进行充分的清洗、归一化和标准化，处理缺失值和异常值，以及解决数据不平衡问题。优质的数据预处理有助于提高模型的准确性和鲁棒性。

特征工程：精心选择和构建特征可以提升模型的性能。特征工程包括选择最相关的特征、创建新特征、进行特征转换和降维等。

超参数调优：随机森林等模型通常有一些超参数需要设置，通过交叉验证等技术来搜索最佳的超参数组合，可以显著提高模型的性能。

集成学习：考虑使用不同的集成学习方法，如梯度提升树（Gradient Boosting Tree）等，它们在一些情况下可能比随机森林表现更好。

模型融合：如果有多个模型，可以考虑使用模型融合技术，如投票（Voting）、堆叠（Stacking）等，将它们的预测结果结合起来，以获得更好的综合性能。

数据增强：对于样本较少的情况，可以使用数据增强技术来生成新的训练样本，增加数据的多样性，从而改进模型的泛化能力。

6.3 模型的推广

个性化预测：每个宝宝都是独特的，模型会根据不同的母亲身心指标，量身定制预测结果，帮助您更好地了解宝宝的个性和需求。

早期干预：宝宝的行为特征和睡眠质量在早期形成，模型可以提前预测，让您有充足的时间做好准备，为宝宝提供早期干预和关爱。

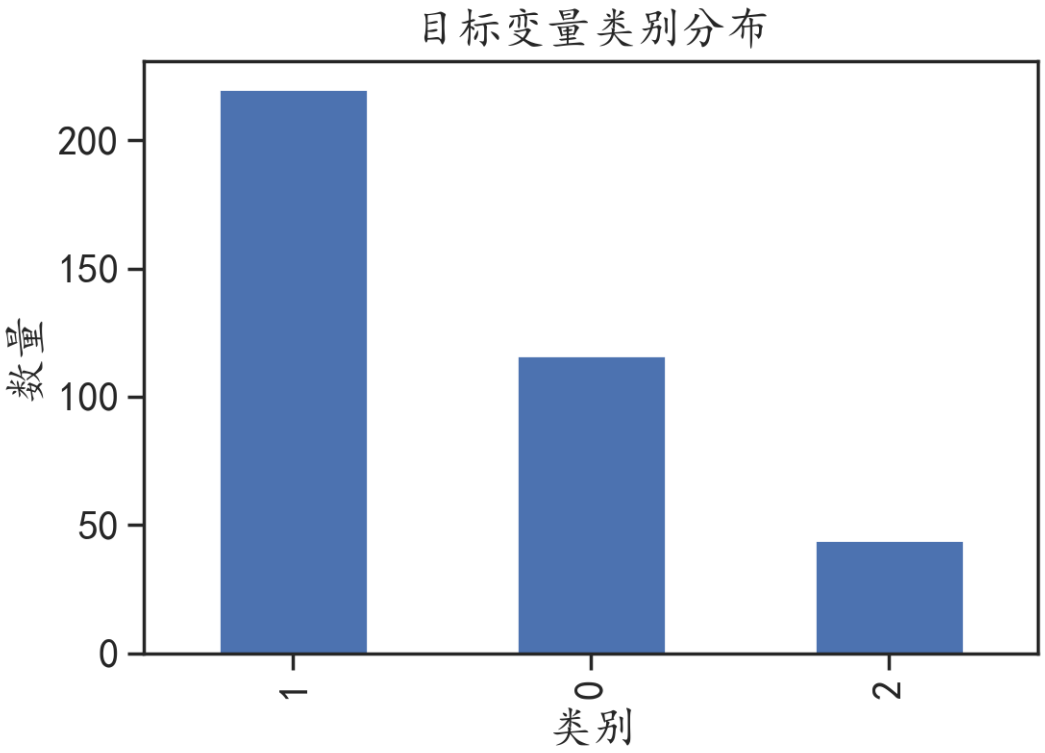
科学育儿指南：模型输出不仅包含预测结果，还提供了科学育儿指南，帮助您优化宝宝的生活方式和睡眠环境，促进宝宝的全面健康成长。

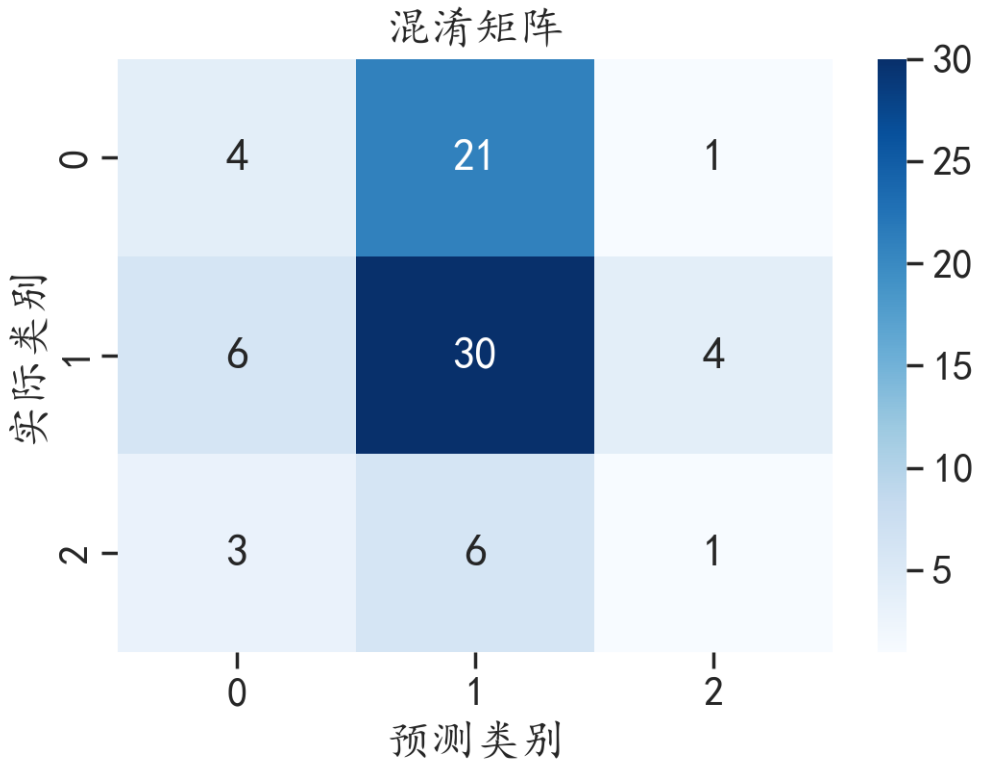
本文可与其他研究团队或领域专家进行合作，共同推广和改进模型。通过与其他研究者的合作，本文可以获得更多一线的反馈和建议，进一步改进模型的性能和应用。这种合作还可以促进模型的交流和共享，推动相关领域的发展。

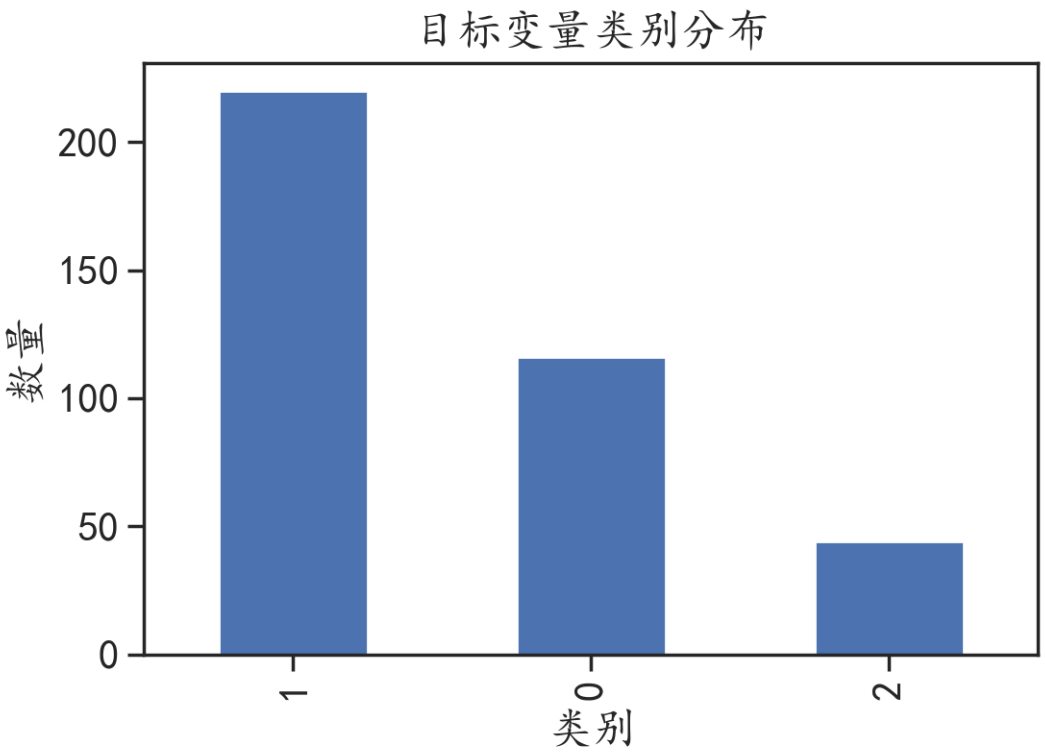
参考文献

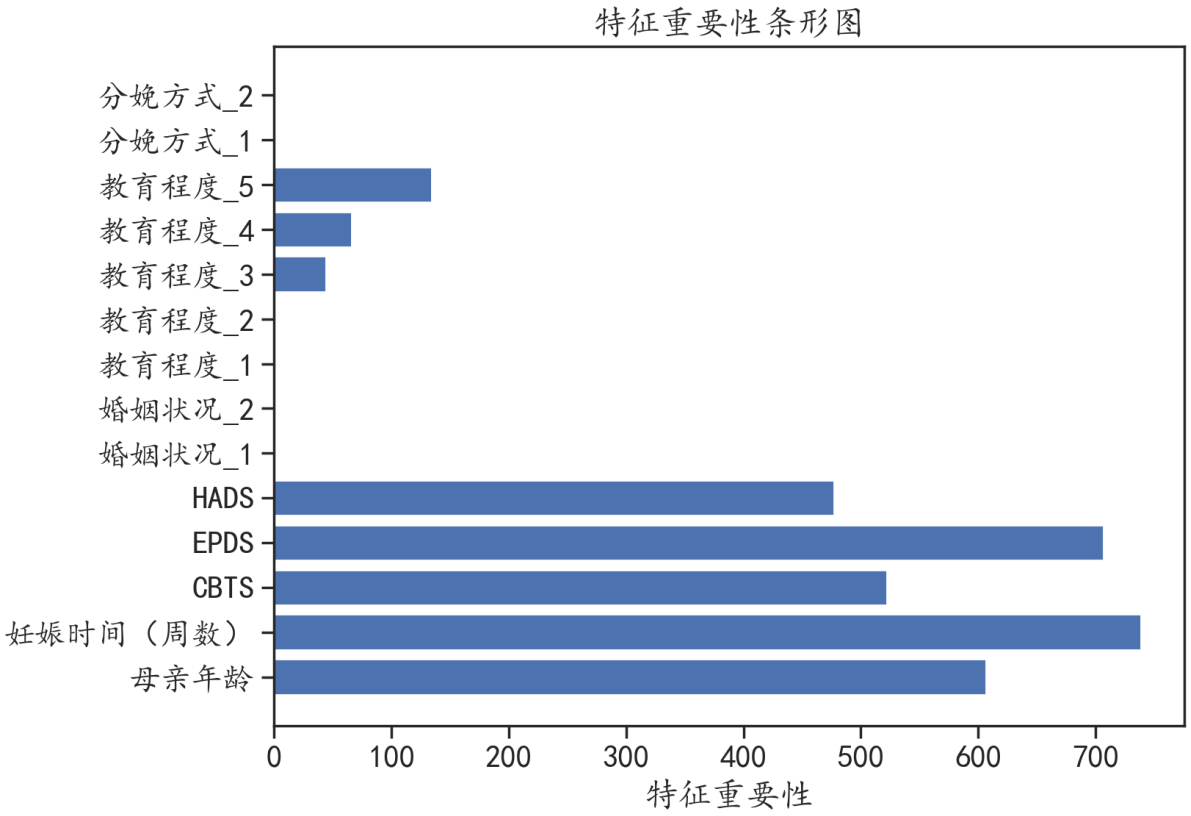
[1]AdairRH，BauchnerH．Sleepproblemsinchildhood[J]． CurrProhcPediatr，23(1)：147-170，1993．
[2]潘玲琳、黄春香、喻忠、杨碧云，婴儿睡眠问题相关因素研究，中国儿童保健杂志[J]，17(1)：43-45，2009。
[3] 司守奎，孙玺菁, Python 数学实验与建模. SCIENCE PRESS, 2020.

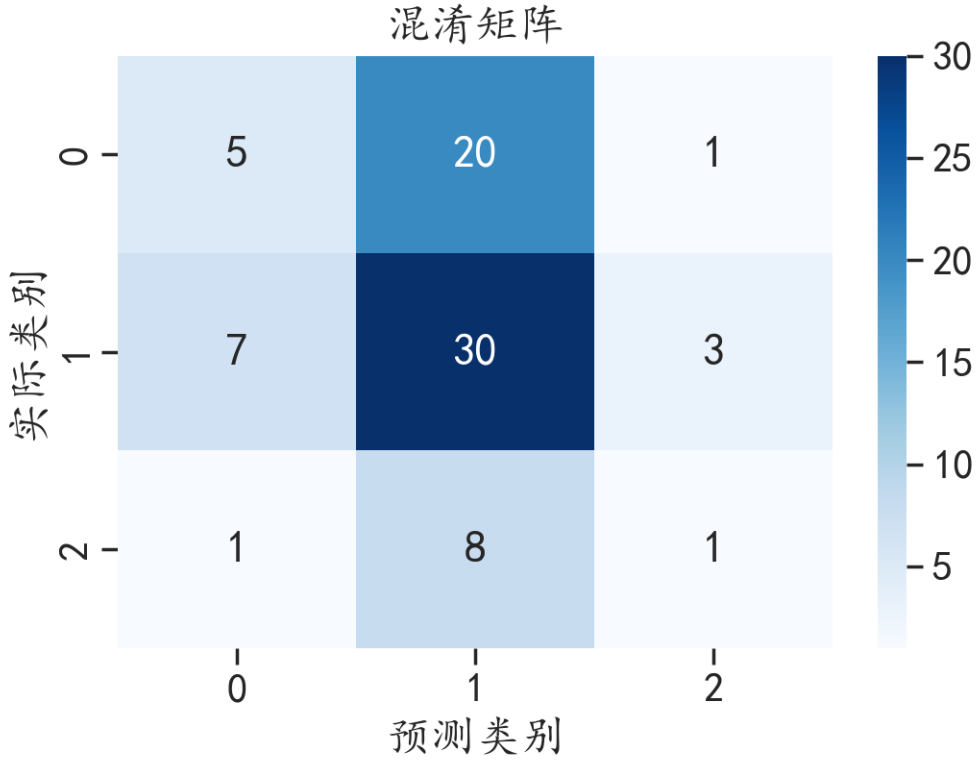
附录 A

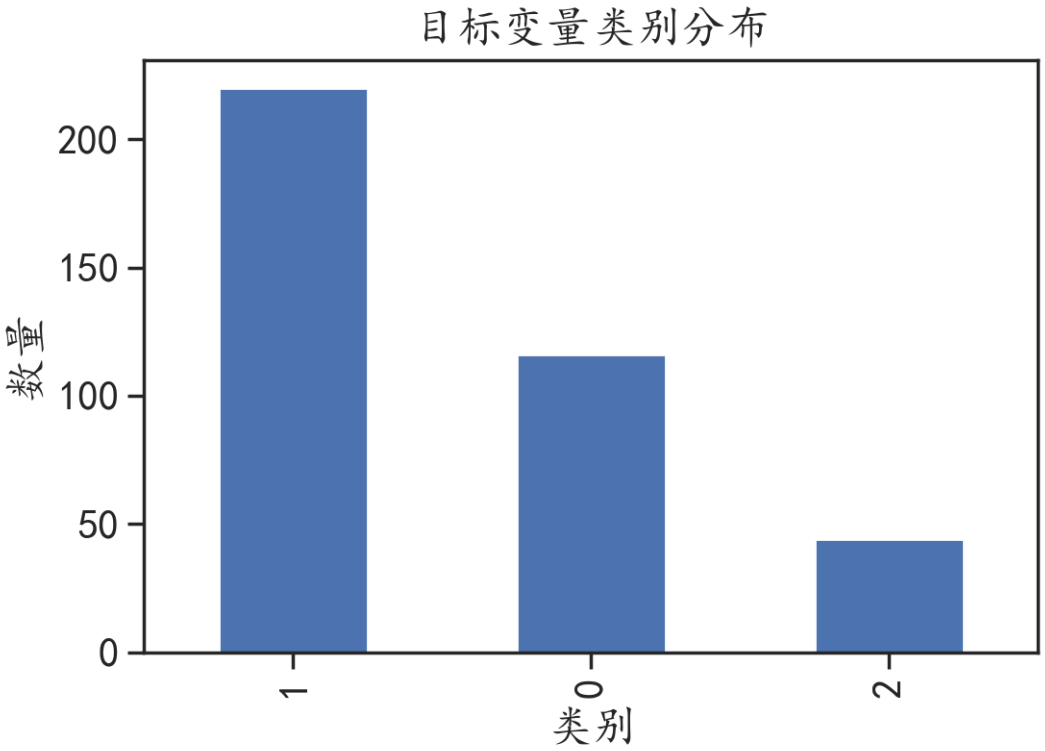

目标变量类别分布
数量
200
150
100
50
0
1
0
2
类别

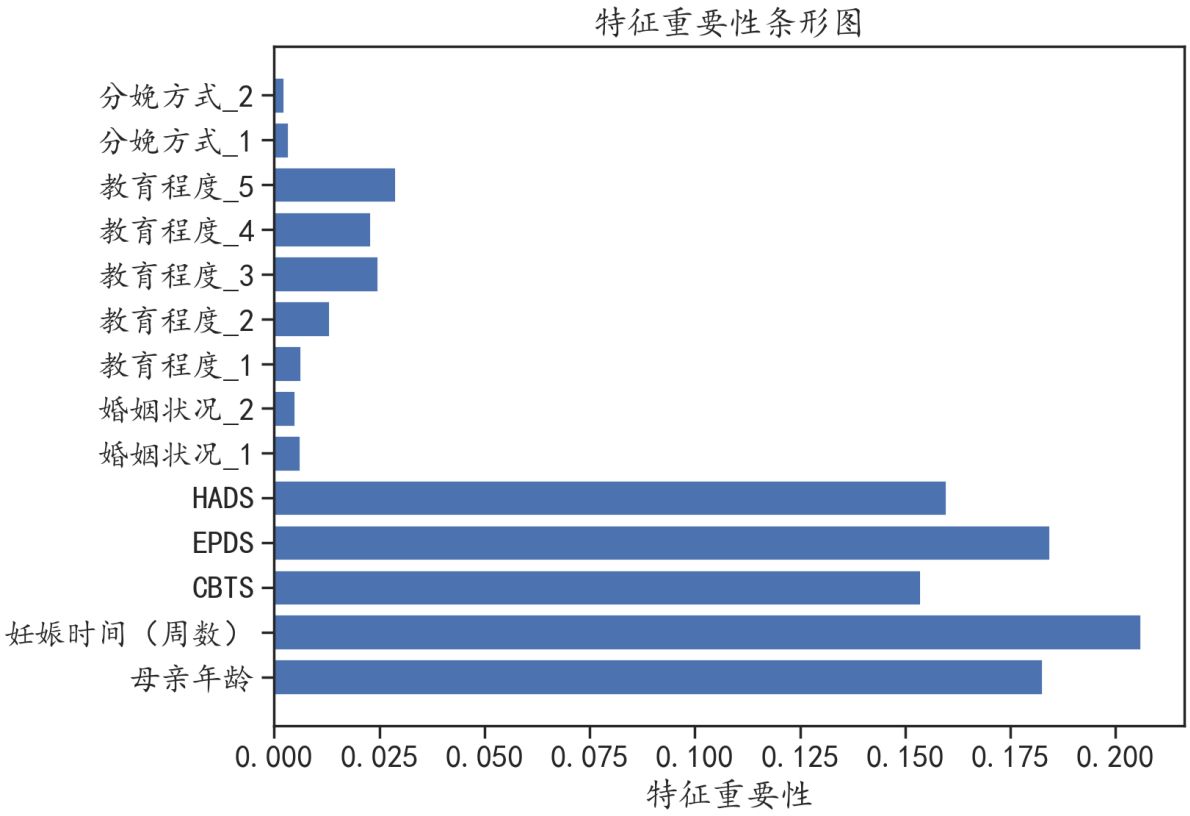

混淆矩阵
实际类别
0
1
2
4
21
1
6
30
4
3
6
1
0
1
2
预测类别
30
25
20
15
10
5

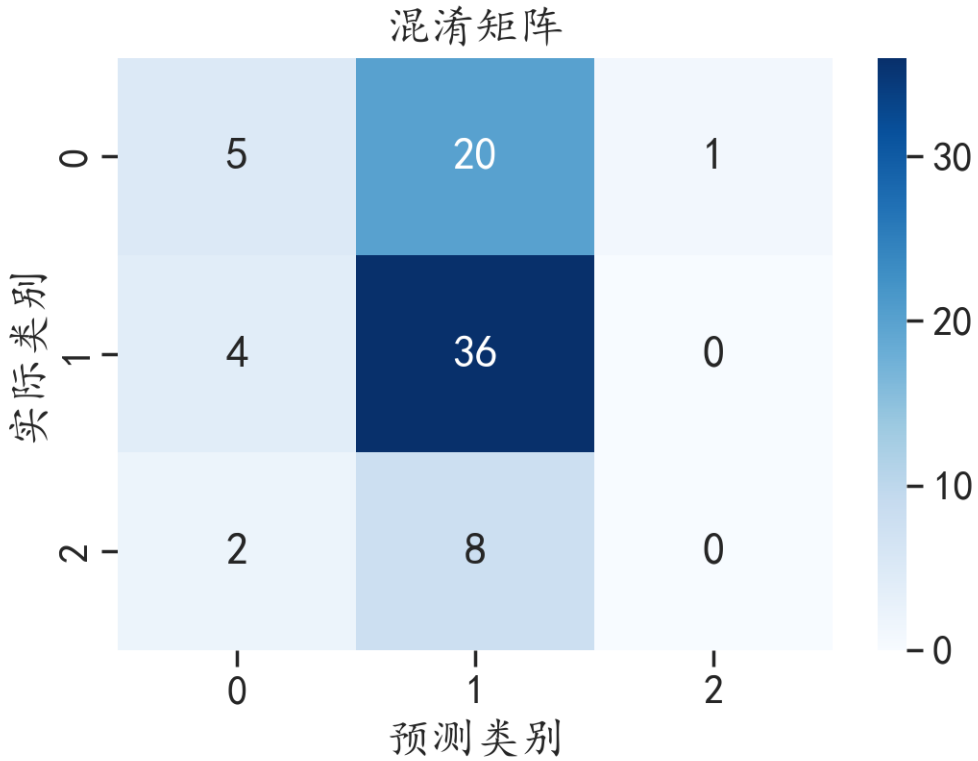

目标变量类别分布
数量
200
150
100
50
0
1
0
2
类别

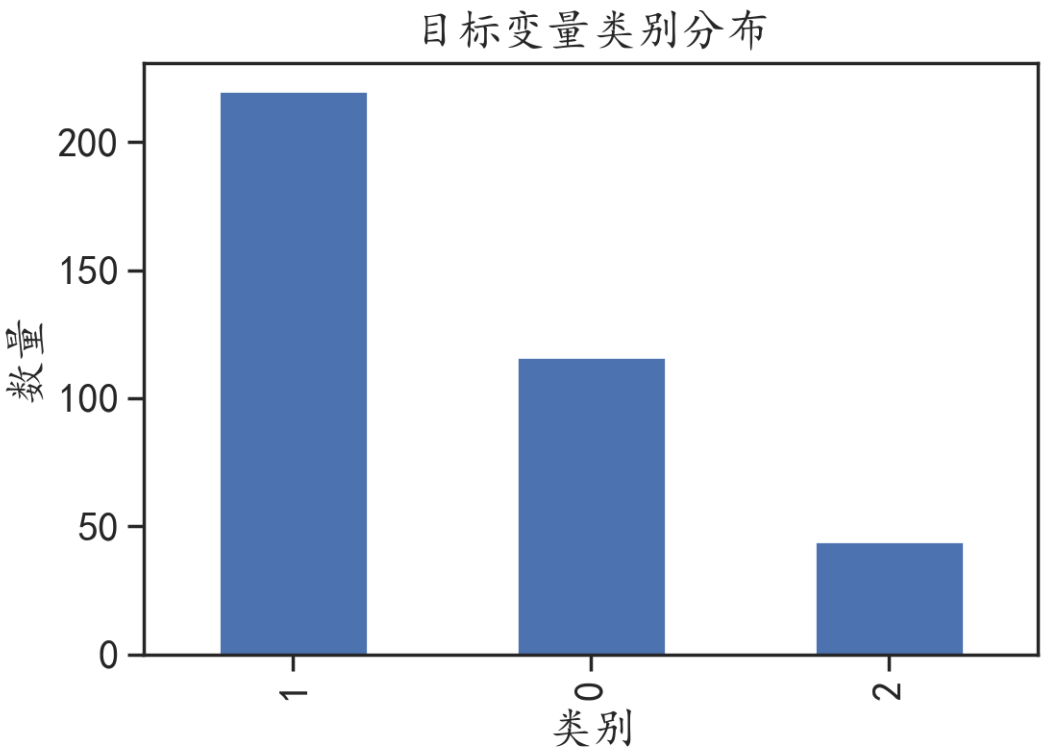

特征重要性条形图
分娩方式_2
分娩方式_1
教育程度_5
教育程度_4
教育程度_3
教育程度_2
教育程度_1
婚姻状况_2
婚姻状况_1
HADS
EPDS
CBTS
妊娠时间（周数）
母亲年龄
0
100
200
300
400
500
600
700
特征重要性

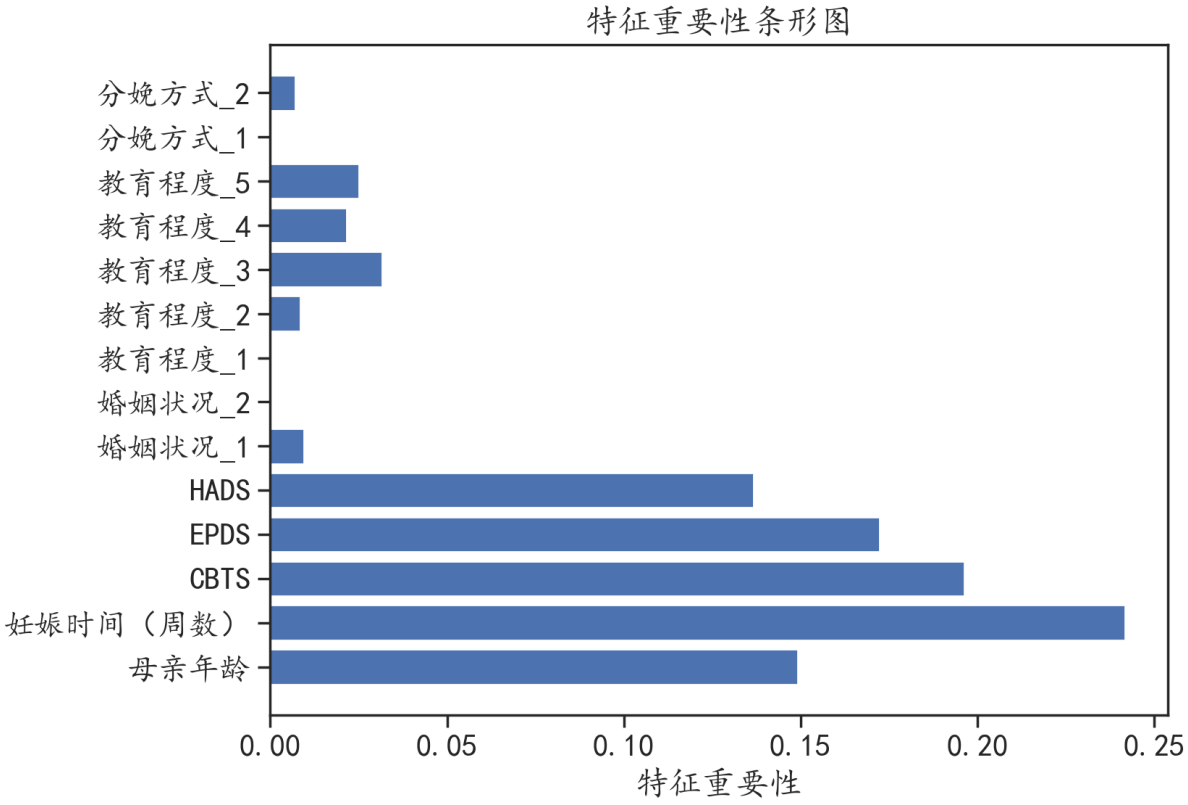

混淆矩阵
实际类别
预测类别
0
1
2
5
20
1
7
30
3
1
8
1
30
25
20
15
10
5

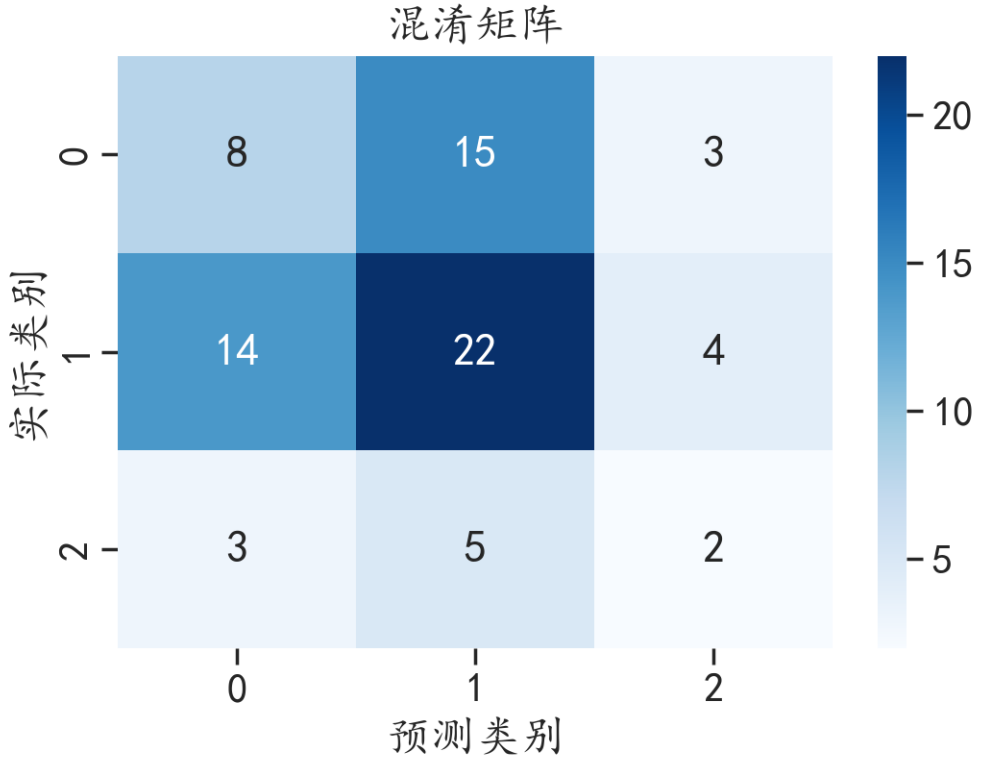

目标变量类别分布
数量
类别
200
150
100
50
0
1
0
2

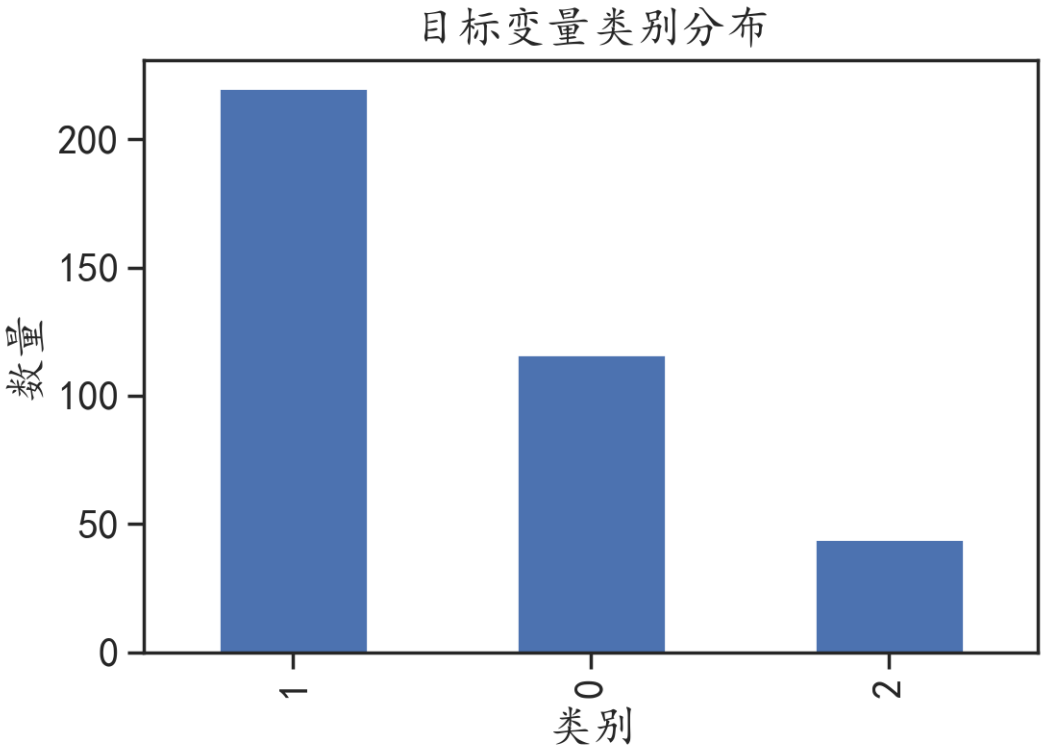

特征重要性条形图
分娩方式_2
分娩方式_1
教育程度_5
教育程度_4
教育程度_3
教育程度_2
教育程度_1
婚姻状况_2
婚姻状况_1
HADS
EPDS
CBTS
妊娠时间（周数）
母亲年龄
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
特征重要性

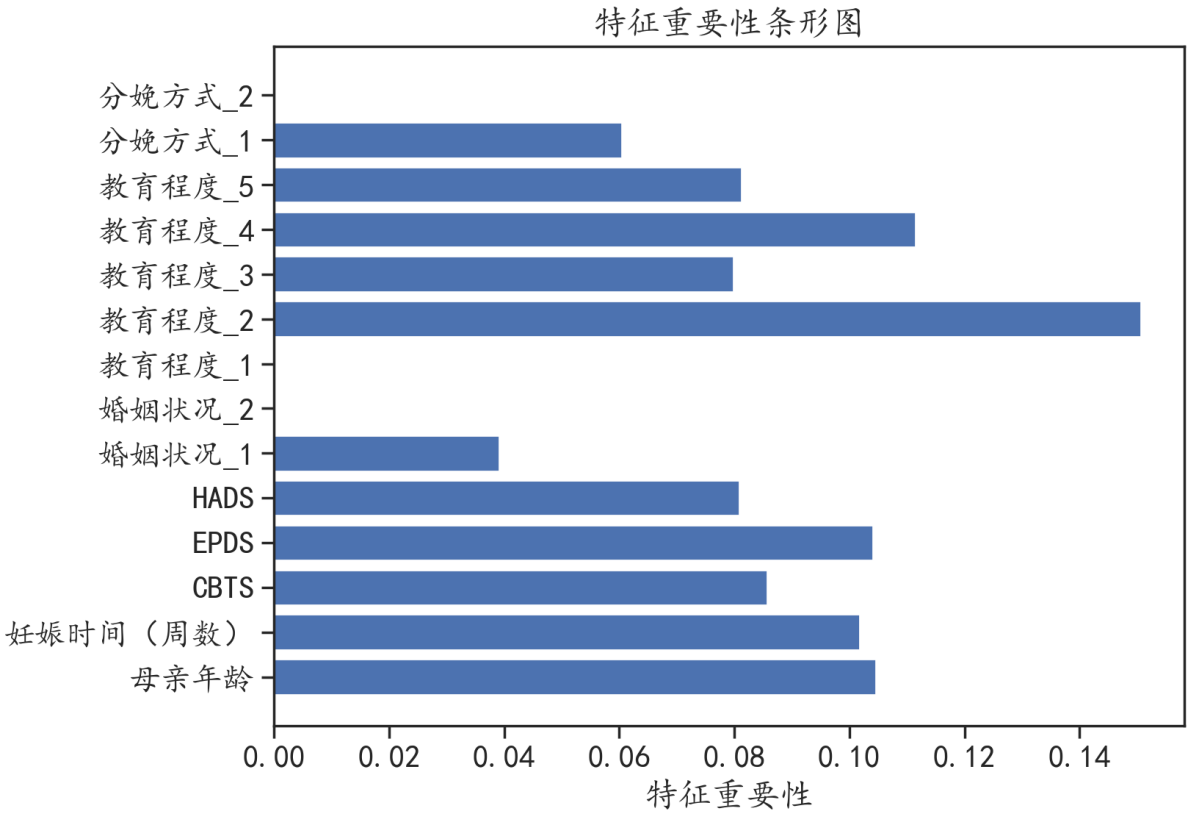

混淆矩阵
实际类别
预测类别
0
1
2
5
20
1
4
36
0
2
8
0
30
20
10
0

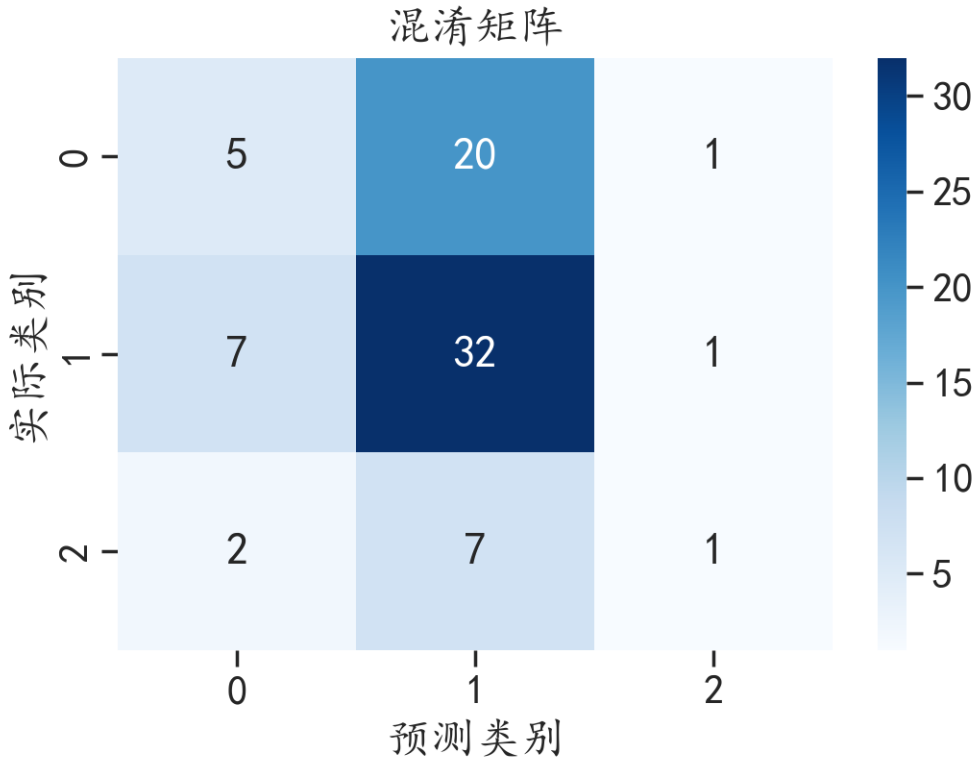

目标变量类别分布
数量
200
150
100
50
0
1
0
2
类别

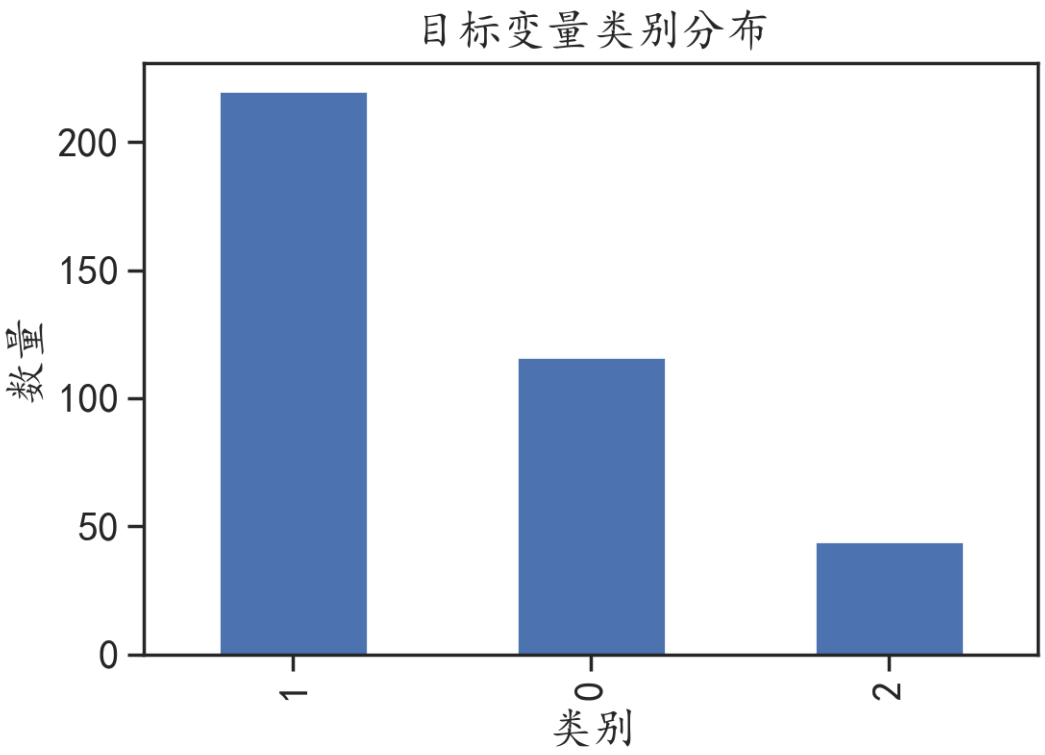

特征重要性条形图
分娩方式_2
分娩方式_1
教育程度_5
教育程度_4
教育程度_3
教育程度_2
教育程度_1
婚姻状况_2
婚姻状况_1
HADS
EPDS
CBTS
妊娠时间（周数）
母亲年龄
0.00
0.05
0.10
0.15
0.20
0.25
特征重要性

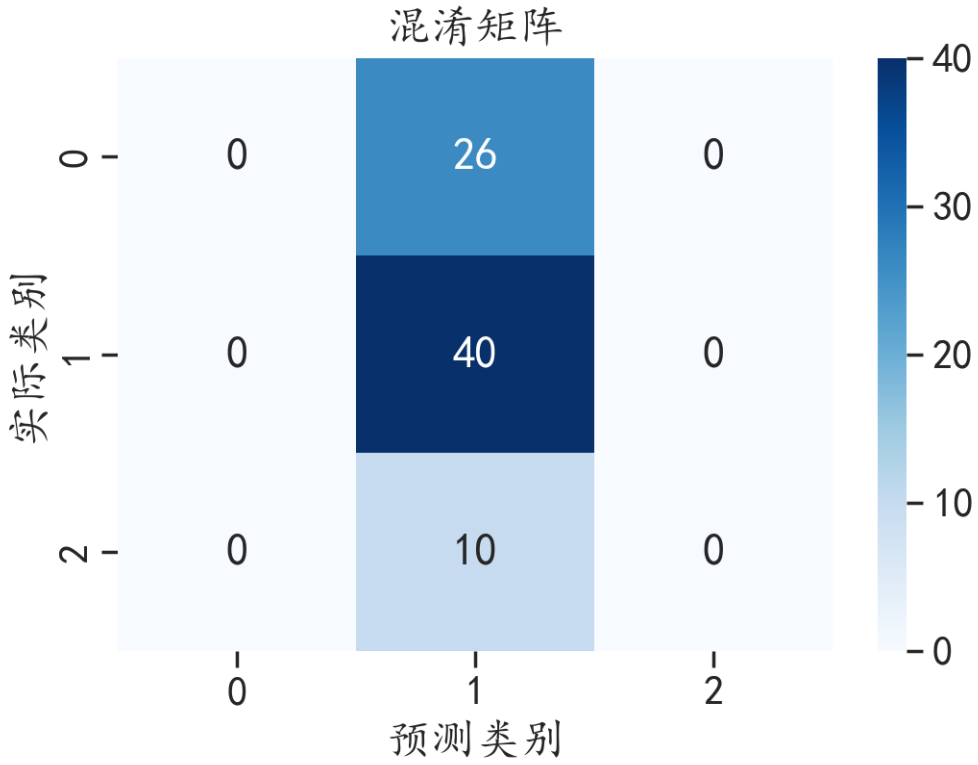

混淆矩阵
8
15
3
14
22
4
3
5
2
0
1
2
实际类别
0
1
2
预测类别
20
15
10
5

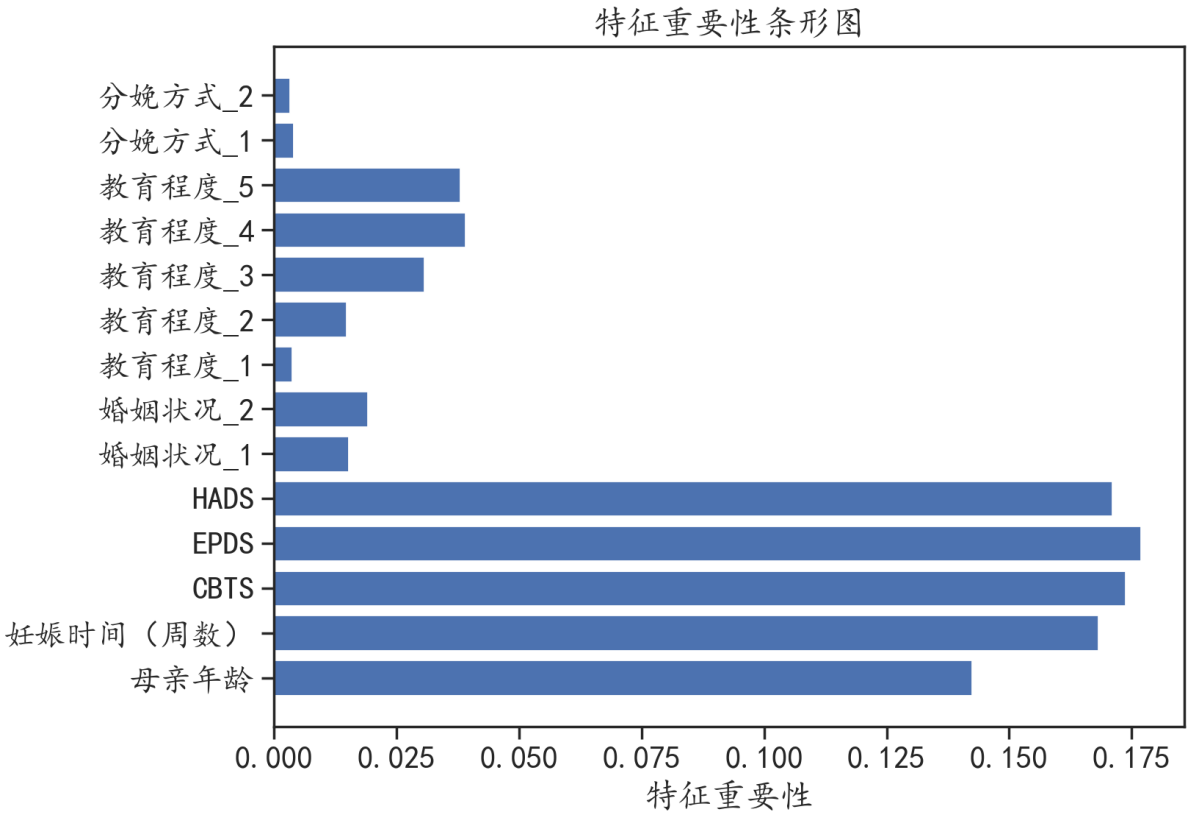

目标变量类别分布
数量
200
150
100
50
0
1
0
2
类别

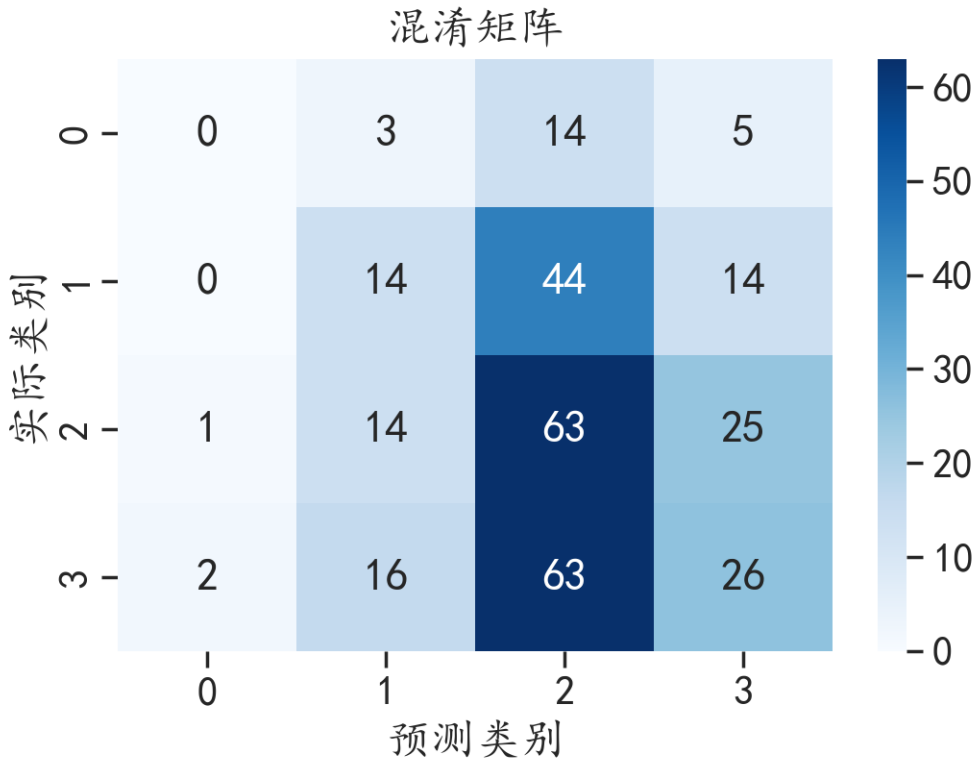

特征重要性条形图
分娩方式_2
分娩方式_1
教育程度_5
教育程度_4
教育程度_3
教育程度_2
教育程度_1
婚姻状况_2
婚姻状况_1
HADS
EPDS
CBTS
妊娠时间（周数）
母亲年龄
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
特征重要性

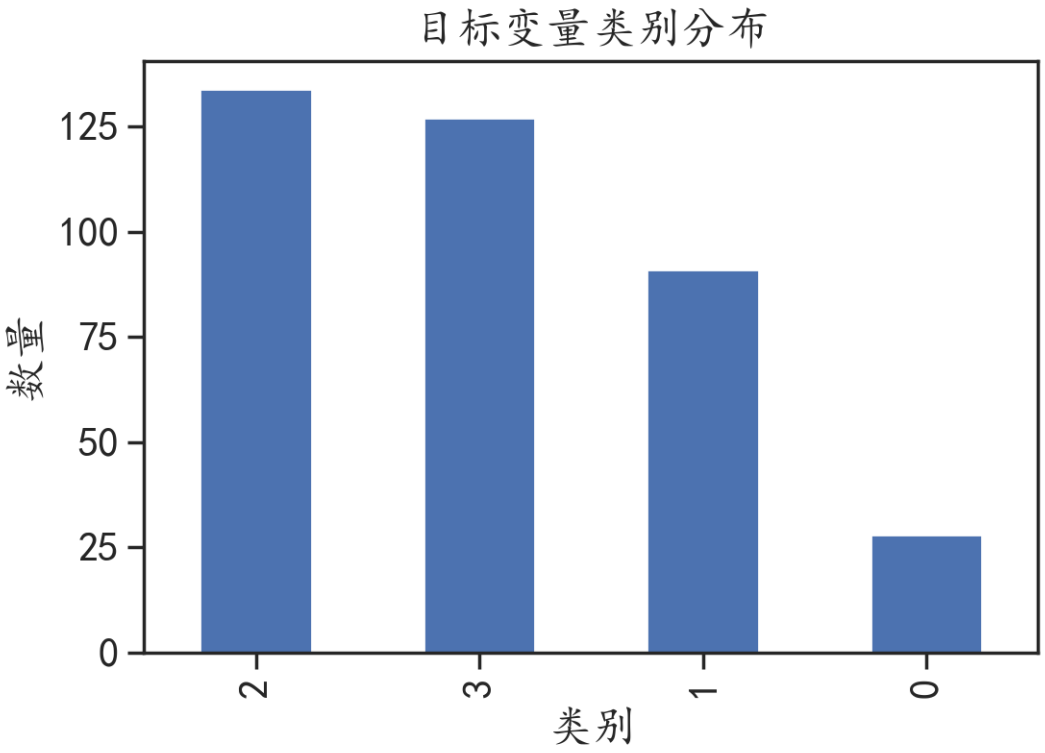

混淆矩阵
实际类别
预测类别
0
1
2
5
20
1
7
32
1
2
7
1
30
25
20
15
10
5

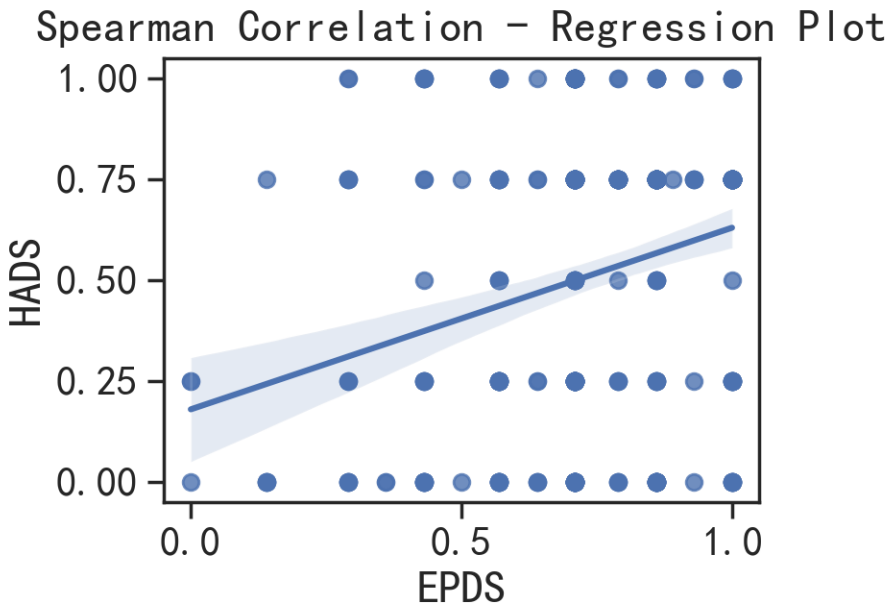

目标变量类别分布
200
150
100
50
0
数量
1
0
2
类别


混淆矩阵
0
26
0
0
40
0
0
10
0
0
1
2
实际类别
0
1
2
预测类别
40
30
20
10
0

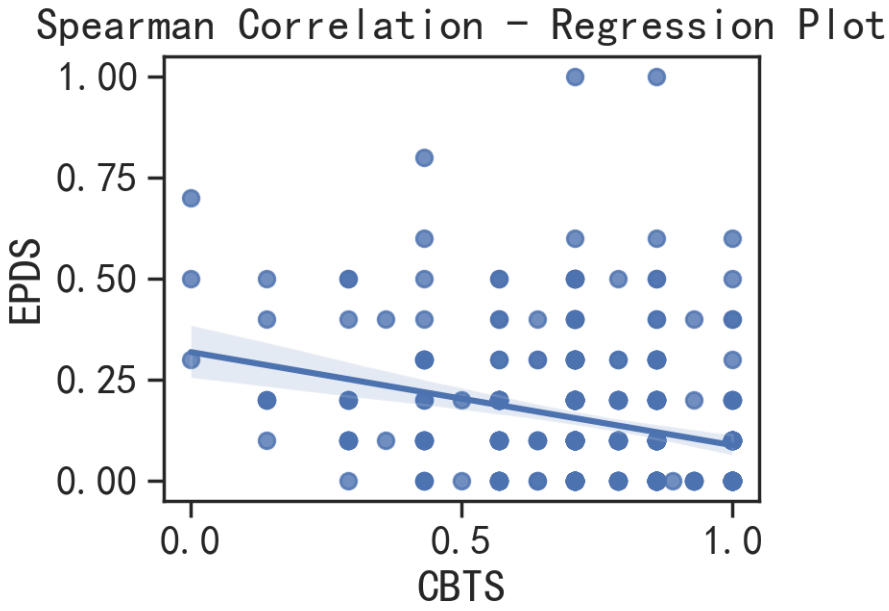

特征重要性条形图
分娩方式_2
分娩方式_1
教育程度_5
教育程度_4
教育程度_3
教育程度_2
教育程度_1
婚姻状况_2
婚姻状况_1
HADS
EPDS
CBTS
妊娠时间（周数）
母亲年龄
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
特征重要性

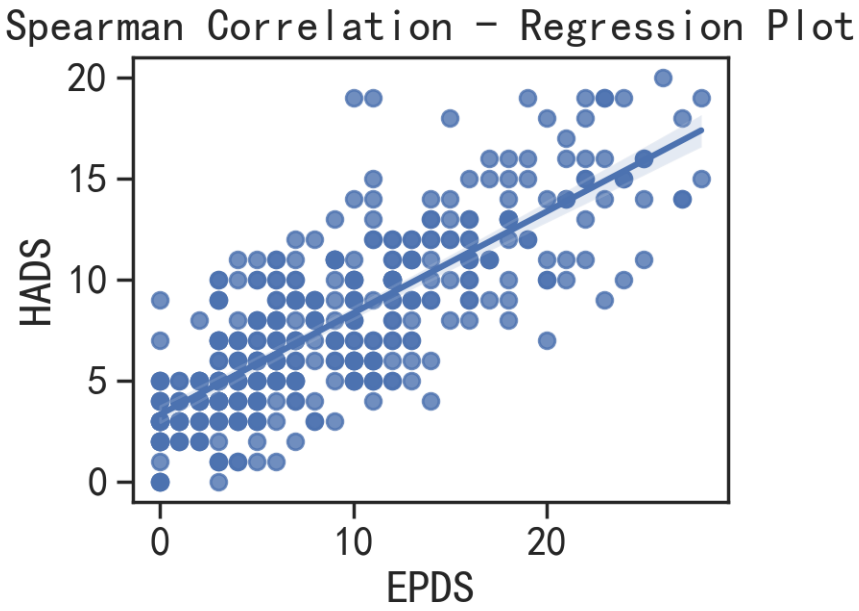

混淆矩阵
实际类别
预测类别
0
1
2
3
0 3 14 5
0 14 44 14
1 14 63 25
2 16 63 26
60
50
40
30
20
10
0

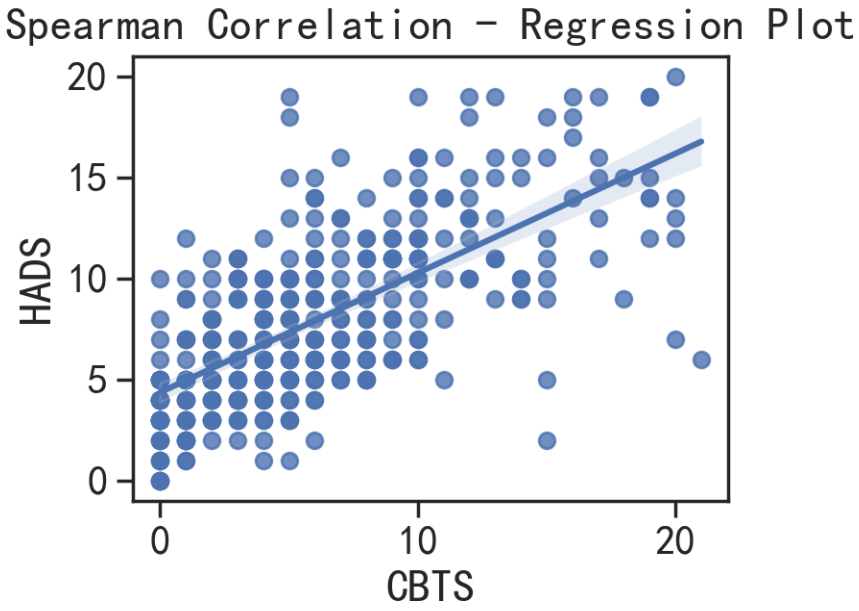

目标变量类别分布
125
100
75
50
25
0
数量
2
3
1
0
类别

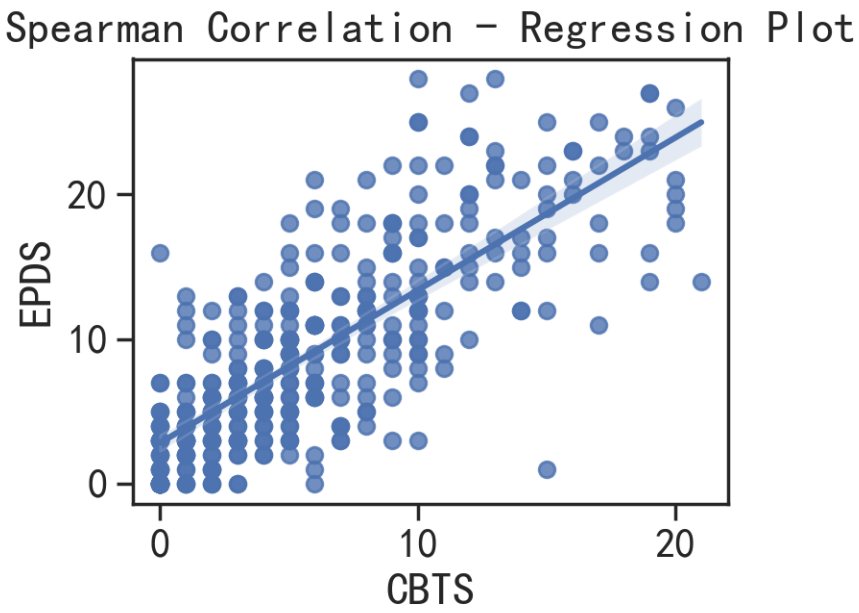

Spearman Correlation - Regression Plot
1.00
0.75
0.50
0.25
0.00
HADS
0.0
0.5
1.0
EPDS


Spearman Correlation - Regression Plot
1.0
0.5
0.0
HADS
0.0
0.5
1.0
CBTS


Spearman Correlation - Regression Plot
1.00
0.75
0.50
0.25
0.00
EPDS
0.0
0.5
1.0
CBTS


Spearman Correlation - Regression Plot
HADS
EPDS
0
5
10
15
20
0
10
20


Spearman Correlation - Regression Plot
HADS
CBTS
0
5
10
15
20
0
10
20


Spearman Correlation - Regression Plot
EPDS
CBTS
0
10
20
0
10
20

附录 B

4.2.1.1 中逻辑回归模型预测补充结果

婴儿行为特征 = 0.0	回归系数	标准误差	Wald	df	P	OR	OR 值 95%置信区间	
上限	下限							
常数	-0.293	0.239	1.504	6	0.220	0.746	0.467	1.192
CBTS	-0.02	0.038	0.277	6	0.599	0.98	0.909	1.057
EPDS	0.003	0.034	0.01	6	0.921	1.003	0.939	1.072
HADS	-0.034	0.045	0.556	6	0.456	0.967	0.884	1.057
婴儿行为特征 = 2.0	回归系数	标准误差	Wald	df	P	OR	OR 值 95%置信区间	
上限	下限							
常数	-1.876	0.361	26.951	6	0.0001	0.153	0.075	0.311
CBTS	-0.062	0.054	1.324	6	0.250	0.94	0.846	1.044
EPDS	0.077	0.046	2.768	6	0.096	1.08	0.986	1.182
HADS	-0.014	0.063	0.047	6	0.828	0.986	0.871	1.117

附录 C

第一题代码:

```
import pandas as pd
import re

# 读取数据
df = pd.read_csv('data1.csv')

invalid_data = pd.DataFrame()

# 1. 婚姻状况
invalid_data = invalid_data.append(df[~df['婚姻状况'].isin([1, 2])])

# 2. 教育程度
invalid_data = invalid_data.append(df[~df['教育程度'].isin([1, 2, 3, 4, 5])])

# 3. 分娩方式
invalid_data = invalid_data.append(df[~df['分娩方式'].isin([1, 2])])

# 4. 婴儿性别
invalid_data = invalid_data.append(df[~df['婴儿性别'].isin([1, 2])])

# 5. 入睡方式
invalid_data = invalid_data.append(df[~df['入睡方式'].isin([1, 2, 3, 4, 5])])

# 6. 婴儿年龄
invalid_data = invalid_data.append(df[~df['婴儿年龄（月）'].isin([1, 2, 3])])

# 7. 睡眠时间
def is_valid_time(time_str):
    pattern = r'^(?:[01]?[0-9]|2[0-3]):[0-5][0-9]$'
    return bool(re.match(pattern, time_str))

invalid_data = invalid_data.append(df[~df['整晚睡眠时间（时：分：秒）'].apply(is_valid_time)])

# 8. 母亲年龄
invalid_data = invalid_data.append(df[df['母亲年龄'] < 0])

# 9. 睡醒次数
invalid_data = invalid_data.append(df[df['睡醒次数'] < 0])

# 补充
```

```
print("不符合要求的数据:")
print(invalid_data)
import pandas as pd
import re

# 读取数据
df = pd.read_csv('data1.csv')

# # 1. 婚姻状况
df = df[df['婚姻状况'].isin([1, 2])]

# 2. 教育程度
df = df[df['教育程度'].isin([1, 2, 3, 4, 5])]
# 3. 分娩方式
df = df[df['分娩方式'].isin([1, 2])]
# 4. 婴儿性别
df = df[df['婴儿性别'].isin([1, 2])]
# 5. 入睡方式
df = df[df['入睡方式'].isin([1, 2, 3, 4, 5])]
# 6. 婴儿年龄
df = df[df['婴儿年龄（月）'].isin([1, 2, 3])]
# 7. 睡眠时间
def is_valid_time(time_str):
    pattern = r'^(?:[01]?[0-9]|2[0-3]):[0-5][0-9]$'
    return bool(re.match(pattern, time_str))
df = df[df['整晚睡眠时间（时：分：秒）'].apply(is_valid_time)]
# 8. 母亲年龄
df = df[df['母亲年龄'] >= 0]
# 9. 睡醒次数
df = df[df['睡醒次数'] >= 0]
# 10. 重新编号
df['编号'] = range(1, len(df) + 1)
# 将婴儿行为特征映射为数字表达
behavior_mapping = {
    '安静型': 1,
    '中等型': 2,
    '矛盾型': 3
}
df['婴儿行为特征'] = df['婴儿行为特征'].map(behavior_mapping)
# 将整晚睡眠时间（时：分：秒）转换成小时表示
def time_to_hours(time_str):
    hours, minutes = map(int, time_str.split(':'))
    return hours + minutes / 60
```

```
df['整晚睡眠时间（时：分：秒）'] = df['整晚睡眠时间（时：分：秒）'].apply(time_to_hours)

# 输出修改后的 DataFrame
print(df['整晚睡眠时间（时：分：秒）'])
print(df['婴儿行为特征'])
# 保存修改后的 DataFrame 到新文件
df.to_csv('data2.csv', index=False)
import pandas as pd
import re

# 读取数据
df = pd.read_csv('data1.csv')

# # 1. 婚姻状况
df = df[df['婚姻状况'].isin([1, 2])]

# 2. 教育程度
df = df[df['教育程度'].isin([1, 2, 3, 4, 5])]

# 3. 分娩方式
df = df[df['分娩方式'].isin([1, 2])]

# 4. 婴儿性别
df = df[df['婴儿性别'].isin([1, 2])]
# 5. 入睡方式
df = df[df['入睡方式'].isin([1, 2, 3, 4, 5])]

# 6. 婴儿年龄
df = df[df['婴儿年龄（月）'].isin([1, 2, 3])]

# 7. 睡眠时间
def is_valid_time(time_str):
    pattern = r'^(?:[01]?[0-9]|2[0-3]):[0-5][0-9]$'
    return bool(re.match(pattern, time_str))

df = df[df['整晚睡眠时间（时：分：秒）'].apply(is_valid_time)]

# 8. 母亲年龄
df = df[df['母亲年龄'] >= 0]

# 9. 睡醒次数
df = df[df['睡醒次数'] >= 0]
# 10. 重新编号
df['编号'] = range(1, len(df) + 1)
```

```
# 将婴儿行为特征映射为数字表达
behavior_mapping = {
    '安静型': 0,
    '中等型': 1,
    '矛盾型': 2
}
df['婴儿行为特征'] = df['婴儿行为特征'].map(behavior_mapping)

# 将整晚睡眠时间（时：分：秒）转换成小时表示
def time_to_hours(time_str):
    hours, minutes = map(int, time_str.split(':'))
    return hours + minutes / 60
df['整晚睡眠时间（时：分：秒）'] = df['整晚睡眠时间（时：分：秒）'].apply(time_to_hours)
# 输出修改后的 DataFrame
print(df['整晚睡眠时间（时：分：秒）'])
print(df['婴儿行为特征'])
# 保存修改后的 DataFrame 到新文件
df.to_csv('data2.csv', index=False)
import pandas as pd
import re
# 读取数据
df = pd.read_csv('pred.csv')
# # 1. 婚姻状况
df = df[df['婚姻状况'].isin([1, 2])]
# 2. 教育程度
df = df[df['教育程度'].isin([1, 2, 3, 4, 5])]
# 3. 分娩方式
df = df[df['分娩方式'].isin([1, 2])]
# 8. 母亲年龄
df = df[df['母亲年龄'] >= 0]
# 10. 重新编号
df['编号'] = range(1, len(df) + 1)
# 保存修改后的 DataFrame 到新文件
df.to_csv('pred.csv', index=False)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 读取数据
data = pd.read_csv('data3.csv')

# 进行 Pearson 相关性分析
correlation_matrix = data.corr(method='pearson')
```

```
# 绘制热力图
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Pearson Correlation Heatmap")
plt.show()

# 输出相关性矩阵表
print("\nCorrelation Matrix Table:")
print(correlation_matrix)
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt

def caseplot(name):
    # 读取数据
    data = pd.read_csv('data4.csv')
    plt.figure(figsize=(5, 5))
    # 绘制箱线图
    plt.boxplot([data[data['婴儿行为特征'] == 0][name],
                 data[data['婴儿行为特征'] == 1][name],
                 data[data['婴儿行为特征'] == 2][name]],
                 labels=['0', '1', '2'])
    plt.xlabel('婴儿行为特征')
    plt.ylabel(name)
      plt.title(f'{name}和婴儿行为特征之间的关联')
    plt.show()
caseplot('母亲年龄')
caseplot('妊娠时间（周数）')
caseplot('CBTS')
caseplot('EPDS')
caseplot('HADS')
def crossplot(name):

    data = pd.read_csv('data3.csv')
    # 创建交叉表
    cross_table = pd.crosstab(data[name], data['婴儿行为特征'])
```

```
    # 绘制热力图
    plt.figure(figsize=(5, 5))
    sns.heatmap(cross_table, annot=True, cmap='YlGnBu', fmt='d')
    plt.xlabel('婴儿行为特征')
    plt.ylabel(name)
    plt.title(f'{name}和婴儿行为特征之间的关联')
    plt.show()
crossplot('婚姻状况')
crossplot('教育程度')
crossplot('分娩方式')
import pandas as pd
import matplotlib.pyplot as plt
def plot_boxplot(name):
        # 读取 CSV 文件
    data = pd.read_csv('data3.csv')

    # 提取母亲年龄和婴儿睡眠质量数据
    list1 = data[name]
    list2 = data['整晚睡眠时间（时：分：秒）']
    list3 = data['睡醒次数']
    list4 = data['入睡方式']

    # 创建一个箱线图
    plt.figure(figsize=(16, 6))
    # 母亲年龄和整晚睡眠时间（时：分：秒）的箱线图
    plt.subplot(2, 2, 1)
    plt.boxplot([list1[list2 == duration] for duration in list2.unique()])
    plt.xticks(range(1, len(list2.unique()) + 1), list2.unique())
    plt.xlabel('整晚睡眠时间（时：分：秒）')
    plt.ylabel(name)
    plt.title('整晚睡眠时间')

    # 母亲年龄和睡醒次数的箱线图
    plt.subplot(2, 2, 2)
    plt.boxplot([list1[list3 == count] for count in list3.unique()])
    plt.xticks(range(1, len(list3.unique()) + 1), list3.unique())
    plt.xlabel('睡醒次数')
     plt.title('睡醒次数')

    # 母亲年龄和入睡方式的箱线图
    plt.subplot(2, 2, 3)
    plt.boxplot([list1[list4 == method] for method in list4.unique()])
    plt.xticks(range(1, len(list4.unique()) + 1), list4.unique())
    plt.xlabel('入睡方式')
```

```
    plt.title('入睡方式')

    # 调整子图之间的间距
    plt.tight_layout()

    # 显示图形
    plt.show()
    plot_boxplot('母亲年龄')
plot_boxplot('妊娠时间（周数）')
plot_boxplot('CBTS')
plot_boxplot('EPDS')
plot_boxplot('HADS')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def crossplot2(name):
    data = pd.read_csv('data3.csv')

    # 创建交叉表
    cross_table1 = pd.crosstab(data[name], data['整晚睡眠时间（时：分：秒）'])
    cross_table2 = pd.crosstab(data[name], data['睡醒次数'])
    cross_table3 = pd.crosstab(data[name], data['入睡方式'])

    # 绘制热力图
    plt.figure(figsize=(8, 5))  # 调整图的大小
     plt.subplot(1, 3, 1)  # 创建第一个子图，用于绘制第一个热力图
    sns.heatmap(cross_table1, cmap='YlGnBu', cbar=False)  # 移除 'annot' 参数
    plt.xlabel('整晚睡眠时间（时：分：秒）')
    plt.ylabel(name)


    plt.subplot(1, 3, 2)  # 创建第二个子图，用于绘制第二个热力图
    sns.heatmap(cross_table2, cmap='YlGnBu', cbar=False)  # 移除 'annot' 参数
    plt.xlabel('睡醒次数')
    plt.ylabel(name)


    plt.subplot(1, 3, 3)  # 创建第三个子图，用于绘制第三个热力图
    sns.heatmap(cross_table3, cmap='YlGnBu', cbar=False)  # 移除 'annot' 参数
    plt.xlabel('入睡方式')
    plt.ylabel(name)
     # 调整子图之间的间距
    plt.tight_layout()
    plt.show()
```

```
crossplot2('婚姻状况')
crossplot2('教育程度')
crossplot2('分娩方式')
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 读取数据
data = pd.read_csv('data3.csv')

# 进行Pearson 相关性分析
correlation_matrix = data.corr(method='pearson')

# 绘制热力图
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Pearson Correlation Heatmap")
plt.show()
from scipy.stats import spearmanr

# 进行Spearman 相关性分析
correlation_matrix2, _ = spearmanr(data)

# 绘制热力图
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix2, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, xticklabels=data.
columns, yticklabels=data.columns)
plt.title("Spearman Correlation Heatmap")
plt.show()

from scipy.stats import kendalltau

# 进行Kendall 一致性检验
correlation_matrix3 = np.zeros((len(data.columns), len(data.columns)))
p_values = np.zeros((len(data.columns), len(data.columns)))
```

```
for i, col1 in enumerate(data.columns):
    for j, col2 in enumerate(data.columns):
        corr, p_value = kendalltau(data[col1], data[col2])
        correlation_matrix3[i, j] = corr
        p_values[i, j] = p_value

# 绘制热力图
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix3, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, xticklabels=data.
columns, yticklabels=data.columns)
plt.title("Kendall Correlation Heatmap")
plt.show()
print(correlation_matrix)
print(correlation_matrix2)
print(correlation_matrix3)
```

第二题代码:

```
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import pandas as pd
import category_encoders as ce
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score
# 加载数据
data = pd.read_csv('data2.csv', encoding='utf-8')
# 分割数据为 X（特征）和 y（目标变量）
X = data[["母亲年龄", "婚姻状况", "教育程度", "妊娠时间（周数）", "分娩方式", "CBTS", "EPDS", "
HADS"]]
y = data["婴儿行为特征"]

# 目标编码（Target Encoding）用于处理分类特征
cat_features = ["婚姻状况", "教育程度", "分娩方式"]
encoder = ce.TargetEncoder(cols=cat_features)
X_encoded = encoder.fit_transform(X, y)
```

```
# 将数据拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 初始化 XGBoost 分类器
model = XGBClassifier()
# 在训练数据上训练模型
model.fit(X_train, y_train)

# 在测试数据上进行预测
y_pred = model.predict(X_test)

# 输出分类报告
print("目标编码分类报告：")
print(classification_report(y_test, y_pred))
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立 XGBoost 模型
model = xgb.XGBClassifier(objective='multi:softmax', num_class=len(y.unique()))

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 输出分类报告
print("热编码分类报告：")
print(classification_report(y_test, y_pred))
# 绘制特征重要性条形图
# 将数据拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 初始化 XGBoost 分类器
model = XGBClassifier()

# 在训练数据上训练模型
model.fit(X_train, y_train)

# 在测试数据上进行预测
y_pred = model.predict(X_test)
```

```
# 计算准确率
accuracy = accuracy_score(y_test, y_pred)
print("准确率：", accuracy)
# 输出分类报告
print("直接分类报告：")
print(classification_report(y_test, y_pred))
# 因此热编码精度更高
import pandas as pd
import xgboost as xgb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立 XGBoost 模型
model = xgb.XGBClassifier(objective='multi:softmax', num_class=len(y.unique()))

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)
# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()
```

```
# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()
# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
plt.title('特征重要性条形图')
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立决策树模型
model = DecisionTreeClassifier()
```

```
# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)
# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
plt.title('特征重要性条形图')
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

```
# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立随机森林模型
model = RandomForestClassifier()
# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)
# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))
```

```
# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
plt.title('特征重要性条形图')
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from lightgbm import LGBMClassifier
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立 LightGBM 模型
model = LGBMClassifier()

# 拟合模型
model.fit(X_train, y_train)
# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()
```

```
# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
plt.title('特征重要性条形图')
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']

# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)
```

```
# 建立 BP 神经网络模型
model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000)

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 注意：BP 神经网络没有内置特征重要性属性
# 不过你可以通过其他方法来评估特征的重要性
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')
```

```
# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']

# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立 SVM 模型
model = SVC()

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))
```

```
# 注意：SVM 没有内置特征重要性属性
# 不过你可以通过其他方法来评估特征的重要性
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data2.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['婴儿行为特征']

# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# 建立随机森林模型
model = RandomForestClassifier()

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
```

```
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
plt.title('特征重要性条形图')
plt.show()
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1)
grid_search.fit(X_train, y_train)
print("Best parameters found: ", grid_search.best_params_)
print("Best accuracy found: {:.2f}".format(grid_search.best_score_))
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))
import pandas as pd
```

```
new_data = pd.read_csv('pred.csv')
# 假设新数据的特征与训练数据相同，直接选取相同的特征列
X_new = new_data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS',
 'HADS']]
# 对婚姻状况、教育程度、分娩方式进行热编码
X_new_encoded = pd.get_dummies(X_new, columns=['婚姻状况', '教育程度', '分娩方式'])
# 使用之前调优过的最佳模型
# # 对新数据进行预测
y_pred_new = best_model.predict(X_new_encoded)
#y_pred_new = model.predict(X_new_encoded)
new_data_with_predictions = new_data.copy()
new_data_with_predictions['Predicted_婴儿行为特征'] = y_pred_new
print(new_data_with_predictions.tail(20))
```

第三题代码:

```
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 读取 CSV 文件
data = pd.read_csv('data5.csv')

# 计算 Spearman 相关系数
spearman_corr = data.corr(method='spearman')

# 绘制热力图
plt.figure(figsize=(10, 8))
sns.heatmap(spearman_corr, annot=True, cmap='coolwarm', center=0, fmt='.2f')
plt.title('Spearman Correlation Heatmap')
plt.show()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 读取 CSV 文件
data = pd.read_csv('data5.csv')
```

```
# 计算 Spearman 相关系数
spearman_corr = data.corr(method='spearman')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='CBTS', y='EPDS', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('CBTS')
plt.ylabel('EPDS')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='CBTS', y='HADS', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('CBTS')
plt.ylabel('HADS')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='EPDS', y='HADS', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('EPDS')
plt.ylabel('HADS')

plt.show()
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# 读取数据
data = pd.read_csv('data5.csv')

# 准备数据
X = data['CBTS']
Y = data['EPDS']
Z = data['HADS']

# 绘制三维曲面图
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

ax.plot_trisurf(X, Y, Z, cmap='viridis')

ax.set_xlabel('CBTS')
```

```
ax.set_ylabel('EPDS')
ax.set_zlabel('HADS')

plt.title('3D Surface Plot')
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 读取数据
data = pd.read_csv('data5.csv')

# 准备数据
X = data['CBTS']
Y = data['EPDS']
Z = data['HADS']

# 创建画布
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# 绘制 3D 柱状图
ax.bar3d(X, Y, 0, 0.5, 0.5, Z, shade=True)

ax.set_xlabel('CBTS')
ax.set_ylabel('EPDS')
ax.set_zlabel('HADS')

plt.title('3D Bar Plot')
plt.show()
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 读取 CSV 数据
data = pd.read_csv('data5.csv')

# 提取三个变量的数据
X = data[['CBTS', 'EPDS', 'HADS']]

# 对数据进行标准化处理（PCA 对数据的尺度敏感，需要进行标准化）
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# 创建 PCA 模型，指定主成分数量为 1
pca = PCA(n_components=1)

# 对数据进行 PCA 转换
X_pca = pca.fit_transform(X_scaled)

print(X_pca)
# 将 PCA 转换后的数据添加为新的一列并保留小数点后 3 位
data['PCA'] = X_pca.round(3)

# 将带有新列的数据保存到原始数据文件中
data.to_csv('data7.csv', index=False)
import pandas as pd
import matplotlib.pyplot as plt

def caseplot(name):
    # 读取数据
    data = pd.read_csv('data7.csv')
    plt.figure(figsize=(5, 5))
    # 绘制箱线图
    plt.boxplot([data[data['婴儿行为特征'] == 0][name],
                 data[data['婴儿行为特征'] == 1][name],
                 data[data['婴儿行为特征'] == 2][name]],
                 labels=['0', '1', '2'])
    plt.xlabel('婴儿行为特征')
    plt.ylabel(name)
    plt.title(f'{name}和婴儿行为特征之间的关联')
    plt.show()
caseplot('PCA')
import pandas as pd
from scipy import stats

# 读取数据
data = pd.read_csv("data7.csv")

# 将数据分成"婴儿行为特征"组别的子集
behavior_0 = data[data["婴儿行为特征"] == 0]["PCA"]
behavior_1 = data[data["婴儿行为特征"] == 1]["PCA"]
behavior_2 = data[data["婴儿行为特征"] == 2]["PCA"]

# 执行独立样本 t 检验
t_statistic_0_vs_1, p_value_0_vs_1 = stats.ttest_ind(behavior_0, behavior_1)
t_statistic_0_vs_2, p_value_0_vs_2 = stats.ttest_ind(behavior_0, behavior_2)
t_statistic_1_vs_2, p_value_1_vs_2 = stats.ttest_ind(behavior_1, behavior_2)
```

```
# 输出结果
print("0 vs 1: t-statistic =", t_statistic_0_vs_1, "p-value =", p_value_0_vs_1)
print("0 vs 2: t-statistic =", t_statistic_0_vs_2, "p-value =", p_value_0_vs_2)
print("1 vs 2: t-statistic =", t_statistic_1_vs_2, "p-value =", p_value_1_vs_2)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 读取 CSV 文件
data = pd.read_csv("data7.csv")

# 使用小提琴图可视化
plt.figure(figsize=(8, 6))
sns.violinplot(x='婴儿行为特征', y='PCA', data=data)
plt.xlabel('婴儿行为特征')
plt.ylabel('PCA')
plt.title('婴儿行为特征与 PCA 的关系')
plt.show()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def spearman_correlation_heatmap(csv_file):
    # 读取 CSV 文件并转换为 DataFrame
    df = pd.read_csv(csv_file)

    # 使用 Spearman 方法计算相关性矩阵
    correlation_matrix = df.corr(method='spearman')

    # 绘制热力图
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
    plt.title('Spearman Correlation Heatmap')
    plt.show()

if __name__ == "__main__":

    csv_file_path = "data7.csv"
    spearman_correlation_heatmap(csv_file_path)
    # 假设婴儿行为特征的类别数量为 3
num_categories = 3

# 根据类别数量平均划分 PCA 的值范围，你可以根据实际情况自行调整
pca_ranges = pd.cut(data['PCA'], num_categories, labels=False)
```

```
# 创建一个映射字典，将 PCA 的值对应到婴儿行为特征类别
pca_to_behavior_mapping = {
    0: '婴儿行为特征为 0',
    1: '婴儿行为特征为 1',
    2: '婴儿行为特征为 2'
}

# 根据 PCA 的值映射到对应的婴儿行为特征类别
data['婴儿行为特征分类'] = pca_ranges.map(pca_to_behavior_mapping)

# 打印结果
print(data[['PCA', '婴儿行为特征分类']])
import pandas as pd
from sklearn.cluster import KMeans

# 读取 data.csv 文件
data = pd.read_csv('data7.csv')

# 提取 PCA 数值作为聚类特征
X = data[['PCA']]

# 假设我们要将 PCA 数值划分为 3 个簇
k = 3
kmeans = KMeans(n_clusters=k, random_state=42)

# 使用 K-means 进行聚类
data['Cluster'] = kmeans.fit_predict(X)

# 打印每个簇的 PCA 数值范围
for i in range(k):
    cluster_min_pca = data[data['Cluster'] == i]['PCA'].min()
    cluster_max_pca = data[data['Cluster'] == i]['PCA'].max()
    print(f'簇 {i} 的 PCA 数值范围从 {cluster_min_pca} 到 {cluster_max_pca}')
import pandas as pd
from sklearn.cluster import KMeans

# 读取 data.csv 文件
data = pd.read_csv('data7.csv')

# 提取 PCA 数值作为聚类特征
X = data[['PCA']]

# 假设我们要将 PCA 数值划分为 3 个簇
k = 3
kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
```

```
# 使用 K-means 进行聚类
data['Cluster'] = kmeans.fit_predict(X)

# 打印每个簇的 PCA 数值范围
for i in range(k):
    cluster_min_pca = data[data['Cluster'] == i]['PCA'].min()
    cluster_max_pca = data[data['Cluster'] == i]['PCA'].max()
    print(f'簇 {i} 的 PCA 数值范围从 {cluster_min_pca} 到 {cluster_max_pca}')

# 如果您愿意，可以将结果保存到一个新的 CSV 文件中
data.to_csv('clustered_data.csv', index=False)
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('data5.csv')
onehot_encoder = OneHotEncoder(sparse=False)
infant_behavior = onehot_encoder.fit_transform(data['婴儿行为特征'].values.reshape(-1, 1))
data = pd.concat([data[['CBTS', 'EPDS', 'HADS']], pd.DataFrame(infant_behavior)], axis=1)

num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data['cluster'] = kmeans.fit_predict(data)
sns.scatterplot(x='CBTS', y='EPDS', hue='cluster', data=data, palette='Set1', s=100, edgecolor='black')
plt.xlabel('CBTS')
plt.ylabel('EPDS')
plt.title('K-means Clustering of CBTS and EPDS')
plt.show()

sns.scatterplot(x='CBTS', y='HADS', hue='cluster', data=data, palette='Set1', s=100, edgecolor='black')
plt.xlabel('CBTS')
plt.ylabel('HADS')
plt.title('K-means Clustering of CBTS and HADS')
plt.show()

sns.scatterplot(x='EPDS', y='HADS', hue='cluster', data=data, palette='Set1', s=100, edgecolor='black')
plt.xlabel('EPDS')
plt.ylabel('HADS')
plt.title('K-means Clustering of EPDS and HADS')
plt.show()

import pandas as pd
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 读取数据
data = pd.read_csv('data6.csv')

# 提取CBTS, EPDS 和HADS 列的数值
X = data[['CBTS', 'EPDS', 'HADS']]

# 假设要分为三个类
num_clusters = 3

# 进行K-means 聚类
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)

# 获取聚类结果
labels = kmeans.labels_

# 添加聚类结果到数据集中
data['Cluster'] = labels

# 绘制三维散点图
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# 每个点的颜色表示它所属的聚类类别
ax.scatter(X['CBTS'], X['EPDS'], X['HADS'], c=labels, cmap='rainbow')

ax.set_xlabel('CBTS')
ax.set_ylabel('EPDS')
ax.set_zlabel('HADS')
ax.set_title('K-means Clustering in 3D')

plt.show()
import pandas as pd
from sklearn.cluster import KMeans

# 读取数据
data = pd.read_csv('data5.csv')

# 提取CBTS, EPDS 和HADS 列的数值
X = data[['CBTS', 'EPDS', 'HADS']]
```

```
# 假设要分为三个类
num_clusters = 3

# 进行K-means 聚类
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)

# 获取聚类结果
labels = kmeans.labels_

# 添加聚类结果到数据集中
data['Cluster'] = labels

# 计算每个聚类簇中婴儿行为特征的频率
cluster_behavior_mapping = {}
for cluster in range(num_clusters):
    cluster_data = data[data['Cluster'] == cluster]
    behavior_counts = cluster_data['婴儿行为特征'].value_counts()
    most_common_behavior = behavior_counts.idxmax()
    cluster_behavior_mapping[cluster] = most_common_behavior

# 输出聚类簇和对应的婴儿行为特征值
print("Cluster - 婴儿行为特征")
for cluster, behavior in cluster_behavior_mapping.items():
    print(f"{cluster} - {behavior}")
```

第四题代码:

```
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 读取数据
data = pd.read_csv('data3.csv')

# 绘制相关性矩阵图
correlation_matrix = data.corr()
```

```
plt.figure(figsize=(4, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 读取 CSV 文件
data = pd.read_csv('data5.csv')

# 计算 Spearman 相关系数
spearman_corr = data.corr(method='spearman')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='整晚睡眠时间（时：分：秒）', y='睡醒次数', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('CBTS')
plt.ylabel('EPDS')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='睡醒次数', y='入睡方式', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('CBTS')
plt.ylabel('HADS')

# 绘制线性回归图
plt.figure(figsize=(4, 3))
sns.regplot(x='整晚睡眠时间（时：分：秒）', y='入睡方式', data=data)
plt.title('Spearman Correlation - Regression Plot')
plt.xlabel('EPDS')
plt.ylabel('HADS')

plt.show()
print('两个正相关，一个负相关')
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# 读取数据
data = pd.read_csv('data3.csv')
```

```
# 准备数据
X = data['整晚睡眠时间（时：分：秒）']
Y = data['睡醒次数']
Z = data['入睡方式']

# 绘制三维曲面图
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

ax.plot_trisurf(X, Y, Z, cmap='viridis')

ax.set_xlabel('整晚睡眠时间（时：分：秒）')
ax.set_ylabel('睡醒次数')
ax.set_zlabel('入睡方式')

plt.title('3D Surface Plot')
plt.show()
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 读取 CSV 数据
data = pd.read_csv('data3.csv')

# 提取三个变量的数据
X = data[['整晚睡眠时间（时：分：秒）','睡醒次数', '入睡方式']]

# 对数据进行标准化处理（PCA 对数据的尺度敏感，需要进行标准化）
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 创建 PCA 模型，指定主成分数量为 1
pca = PCA(n_components=1)

# 对数据进行 PCA 转换
X_pca = pca.fit_transform(X_scaled)

print(X_pca)
# 将 PCA 转换后的数据添加为新的一列并保留小数点后 3 位
data['PCA'] = X_pca.round(3)

# 将带有新列的数据保存到原始数据文件中
data.to_csv('data4.csv', index=False)
import pandas as pd
from factor_analyzer import FactorAnalyzer
```

```
# 读取数据
data = pd.read_csv('data3.csv')

# 只选择'整晚睡眠时间'和'睡醒次数'两列数据用于因子分析
# 你可以根据实际需求选择不同的列进行分析
selected_columns = ['整晚睡眠时间（时：分：秒）', '睡醒次数','入睡方式']
selected_data = data[selected_columns]

# 使用因子分析法，指定因子数为 1
factor_model = FactorAnalyzer(n_factors=1, rotation=None)

# 拟合数据
factor_model.fit(selected_data)

# 获取因子载荷矩阵（成分矩阵）
factor_loadings = factor_model.loadings_

# 打印因子载荷矩阵
print("因子载荷矩阵（成分矩阵）表：")
print(pd.DataFrame(factor_loadings, index=selected_columns, columns=['Factor 1']))
data = pd.read_csv('data2.csv')
# Calculate the scores
data['评分标准'] = 0.54 * data['整晚睡眠时间（时：分：秒）'] - 0.54 * data['睡醒次数'] +0.49 * data['
入睡方式']
# Round the '评分标准' values to two decimal places
data['评分标准'] = data['评分标准'].round(2)
# Output the data with the new '评分标准' column
print(data)
# Save the updated data to a new CSV file
data.to_csv('data7.csv', index=False)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# 1. 读取数据文件 data.csv 并加载为 DataFrame
df = pd.read_csv('data7.csv')

# 2. 提取评分标准这一列作为聚类的数据
data_for_clustering = df['评分标准'].values.reshape(-1, 1)

# 3. 进行 KMeans 聚类分析，设置聚类个数为 4
num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
clusters = kmeans.fit_predict(data_for_clustering)
```

```
# 4. 将聚类结果加入 DataFrame
df['Cluster'] = clusters

# 5. 可视化聚类结果
plt.scatter(df.index, df['评分标准'], c=df['Cluster'], cmap='viridis')
plt.xlabel('Index')
plt.ylabel('评分标准')
plt.title('评分标准聚类结果')
plt.colorbar(label='Cluster')
plt.show()

# 6. 保存结果到表中
df.to_csv('clustered_data.csv', index=False)

# 7. 聚类汇总
cluster_summary = df.groupby('Cluster')['评分标准'].agg(['count', 'mean', 'std'])
print(cluster_summary)
# 聚类汇总
cluster_summary = df.groupby('Cluster')['评分标准'].agg(['count', 'mean', 'std'])

# 按照聚类标签的平均值对聚类进行排序
sorted_clusters = cluster_summary.sort_values(by='mean').index

# 给聚类重新赋值，使得 0 对应的平均值最小，3 对应的最大
cluster_mapping = {cluster: new_label for new_label, cluster in enumerate(sorted_clusters)}
df['Cluster'] = df['Cluster'].map(cluster_mapping)

# 重新计算聚类汇总，按照重新赋值后的聚类标签进行分组
cluster_summary_sorted = df.groupby('Cluster')['评分标准'].agg(['count', 'mean', 'std'])

# 输出重新赋值后的聚类汇总
print(cluster_summary_sorted)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# 1. 读取数据文件 data.csv 并加载为 DataFrame
df = pd.read_csv('data7.csv')

# 2. 提取评分标准这一列作为聚类的数据
data_for_clustering = df['评分标准'].values.reshape(-1, 1)

# 3. 进行 KMeans 聚类分析，设置聚类个数为 4
num_clusters = 4
```

```
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
clusters = kmeans.fit_predict(data_for_clustering)

# 4. 将聚类结果加入 DataFrame
df['Cluster'] = clusters

# 5. 聚类汇总
cluster_summary = df.groupby('Cluster')['评分标准'].agg(['count', 'mean', 'std'])

# 6. 按照聚类标签的平均值对聚类进行排序，并重新赋值
sorted_clusters = cluster_summary.sort_values(by='mean').index
cluster_mapping = {cluster: new_label for new_label, cluster in enumerate(sorted_clusters)}
df['New_Cluster'] = df['Cluster'].map(cluster_mapping)

# 7. 输出重新赋值后的聚类汇总
cluster_summary_sorted = df.groupby('New_Cluster')['评分标准'].agg(['count', 'mean', 'std'])
print(cluster_summary_sorted)
# 5. 可视化聚类结果
plt.scatter(df.index, df['评分标准'], c=df['Cluster'], cmap='viridis')
plt.xlabel('Index')
plt.ylabel('评分标准')
plt.title('评分标准聚类结果')
plt.colorbar(label='Cluster')
plt.show()


# 8. 保存新的结果到源文件中
df.to_csv('data.csv', index=False)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report

# 读取数据
data = pd.read_csv('data.csv')

# 自变量
X = data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS', 'HADS']]

# 因变量
y = data['睡眠质量']
```

```
# 对婚姻状况、教育程度、分娩方式进行热编码
X_encoded = pd.get_dummies(X, columns=['婚姻状况', '教育程度', '分娩方式'])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.8, random_state=42)

# 建立随机森林模型
model = RandomForestClassifier()

# 拟合模型
model.fit(X_train, y_train)

# 使用模型进行预测
y_pred = model.predict(X_test)

# 绘制目标变量类别分布（条形图）
plt.figure(figsize=(6, 4))
y.value_counts().plot(kind='bar')
plt.xlabel("类别")
plt.ylabel("数量")
plt.title("目标变量类别分布")
plt.show()

# 计算混淆矩阵
conf_matrix = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵（热图）
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel("预测类别")
plt.ylabel("实际类别")
plt.title("混淆矩阵")
plt.show()

# 输出分类报告
print("分类报告：")
print(classification_report(y_test, y_pred))

# 绘制特征重要性条形图
plt.figure(figsize=(8, 6))
feat_importance = model.feature_importances_
feat_names = X_encoded.columns
plt.barh(feat_names, feat_importance)
plt.xlabel('特征重要性')
```

```
plt.title('特征重要性条形图')
plt.show()
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1)
grid_search.fit(X_train, y_train)
print("Best parameters found: ", grid_search.best_params_)
print("Best accuracy found: {:.2f}".format(grid_search.best_score_))
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
import pandas as pd

new_data = pd.read_csv('pred.csv')
# 假设新数据的特征与训练数据相同，直接选取相同的特征列
X_new = new_data[['母亲年龄', '婚姻状况', '教育程度', '妊娠时间（周数）', '分娩方式', 'CBTS', 'EPDS',
 'HADS']]

# 对婚姻状况、教育程度、分娩方式进行热编码
X_new_encoded = pd.get_dummies(X_new, columns=['婚姻状况', '教育程度', '分娩方式'])
# 使用之前调优过的最佳模型
best_model = grid_search.best_estimator_

# # 对新数据进行预测
#y_pred_new = best_model.predict(X_new_encoded)
y_pred_new = model.predict(X_new_encoded)
new_data_with_predictions = new_data.copy()
new_data_with_predictions['Predicted_婴儿行为特征'] = y_pred_new

# Print the new DataFrame
# Print rows from index 390 to the end
print(new_data_with_predictions.tail(20))
```

第五题代码:

```
## 图像显示中文的问题
import matplotlib
matplotlib.rcParams['axes.unicode_minus']=False
import seaborn as sns
sns.set(font= "Kaiti",style="ticks",font_scale=1.4)
## 输出高清图像
%config InlineBackend.figure_format = 'retina'
```

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 读取数据
data = pd.read_csv('data2.csv')

# 进行Pearson 相关性分析
correlation_matrix = data.corr(method='pearson')

# 绘制热力图
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Pearson Correlation Heatmap")
plt.show()
import pandas as pd
import matplotlib.pyplot as plt

def caseplot(name):
    # 读取数据
    data = pd.read_csv('data.csv')
    plt.figure(figsize=(5, 5))
    # 绘制箱线图
    plt.boxplot([data[data['睡眠质量'] == 0][name],
                 data[data['睡眠质量'] == 1][name],
                 data[data['睡眠质量'] == 2][name],
                 data[data['睡眠质量'] == 2][name]],
                 labels=['0', '1', '2','3'])
    plt.xlabel('睡眠质量')
    plt.ylabel(name)
    plt.title(f'{name}和睡眠质量之间的关联')
    plt.show()
caseplot('CBTS')
caseplot('EPDS')
caseplot('HADS')
import pandas as pd
import numpy as np
from sklearn.cross_decomposition import PLSRegression

# 从 CSV 文件读取数据
data = pd.read_csv('data.csv')
```

```
# 分离自变量和因变量
X = data[['CBTS', 'EPDS', 'HADS']].values
y = data['评分标准'].values

# 计算自变量的均值和标准差
mean_X = np.mean(X, axis=0)
std_X = np.std(X, axis=0)

# 对自变量进行标准化处理
X_normalized = (X - mean_X) / std_X

# 使用偏最小二乘-判别分析建立模型
pls_model = PLSRegression(n_components=2)  # 这里选择 2 个主成分，也可以根据需要调整
pls_model.fit(X_normalized, y)

# 查看模型的系数
print("回归系数:")
print(pls_model.coef_)

# 查看模型的截距
print("截距:")
print(pls_model.intercept_)
import matplotlib.pyplot as plt

# Get the predicted values from the PLSRegression model
predicted_values = pls_model.predict(X_normalized)

# Create a scatter plot of actual vs predicted values
plt.figure(figsize=(8, 6))
plt.scatter(y, predicted_values, c='b', label='Predicted')
plt.scatter(y, y, c='r', label='Actual')  # Add actual values with a different color
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.show()
import matplotlib.pyplot as plt

# Get the predicted values from the PLSRegression model
predicted_values = pls_model.predict(X_normalized)

# Create an array of indices (assuming 'data.csv' contains an 'ID' column)
indices = data['编号'].values
```

```
# Create a scatter plot of actual and predicted values with different colors
plt.figure(figsize=(10, 6))
plt.scatter(indices, predicted_values, c='b', label='Predicted', marker='o')
plt.scatter(indices, y, c='r', label='Actual', marker='x')  # Add actual values with a different color
plt.xlabel('Index')
plt.ylabel('Values')
plt.title('精确值和实际值对比')
plt.legend()
plt.show()
import pandas as pd
import numpy as np
from sklearn.cross_decomposition import PLSRegression

# 从 CSV 文件读取数据
data = pd.read_csv('data.csv')

# 分离自变量和因变量
X = data[['CBTS', 'EPDS', 'HADS']].values
y = data['评分标准'].values

# 计算自变量的均值和标准差
mean_X = np.mean(X, axis=0)
std_X = np.std(X, axis=0)

# 对自变量进行标准化处理
X_normalized = (X - mean_X) / std_X

# 使用偏最小二乘-判别分析建立模型
pls_model = PLSRegression(n_components=2)  # 这里选择 2 个主成分，也可以根据需要调整
pls_model.fit(X_normalized, y)

# 查看模型的系数
print("回归系数:")
print(pls_model.coef_)

# 查看模型的截距
print("截距:")
print(pls_model.intercept_)
mport pulp

# 创建线性规划问题
lp_problem = pulp.LpProblem("Minimize_w", pulp.LpMinimize)

# 定义决策变量
x1 = pulp.LpVariable("x1", 0, 30, cat='Integer')
```

```
x2 = pulp.LpVariable("x2", 0, 30, cat='Integer')
x3 = pulp.LpVariable("x3", 0, 30, cat='Integer')
xC = pulp.LpVariable("xC", 0, 15, cat='Integer')
xE = pulp.LpVariable("xE", 0, 22, cat='Integer')
xH = pulp.LpVariable("xH", 0, 18, cat='Integer')

y1 = 870.67 * x1 + 200
y2 = 690 * x2 + 500
y3 = 2440 * x3 + 300
y =  0.33810148 * xC   -0.54102068 * xE   -0.01006329 * xH + 6.19028947
# c,e,h 和睡眠评分标准的映射

# 定义目标函数
lp_problem += y1 + y2 + y3, "Minimize_w"

# 定义约束条件
lp_problem += y >= 7.885748
lp_problem += x1 == 15 - xC
lp_problem += x2 == 22 - xE
lp_problem += x3 == 18 - xH

# 求解问题
lp_problem.solve()

# 输出结果
print("最小值 w 的取值为:", pulp.value(lp_problem.objective))
print("x1 的取值为:", pulp.value(x1))
print("x2 的取值为:", pulp.value(x2))
print("x3 的取值为:", pulp.value(x3))
```

```
import cvxpy as cp

# Define variables
x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()


objective_function = cp.Minimize(200 * cp.exp(0.88 * 15) - 200 * cp.exp(0.88 * x1) +
                                 500 * cp.exp(0.66 * 22) - 500 * cp.exp(0.66 * x2) +
                                 300 * cp.exp(0.75 * 18) - 300 * cp.exp(0.75 * x3))

objective_function = cp.Minimize(200 * cp.exp(0.88 * (15 - x1)) +
                                 500 * cp.exp(0.66 * (22 - x2)) +
                                 300 * cp.exp(0.75 * (18 - x3)))
```

```
# Define the constraints
constraints = [0.34 * x1 - 0.54 * x2 - 0.01 * x3 + 6.2 >= 7.8,
               x1 >= 0, x1 <= 15,
               x2 >= 0, x2 <= 22,
               x3 >= 0, x3 <= 18]

# Create the problem and solve
problem = cp.Problem(objective_function, constraints)
result = problem.solve()

# Output results
print("Optimal value of x1:", x1.value)
print("Optimal value of x2:", x2.value)
print("Optimal value of x3:", x3.value)
print("Optimal value of the objective function:", result)
```

```
import cvxpy as cp

# Define variables
x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()

# Define the objective function
objective_function = cp.Minimize(200 * cp.exp(0.88 * (15 - x1)) +
                                 500 * cp.exp(0.66 * (22 - x2)) +
                                 300 * cp.exp(0.75 * (18 - x3)))

# Define the updated constraints
constraints = [x1 >= 0, x1 <= 4,
               x2 >= 0, x2 <= 6.5,
               x3 >= 0, x3 <= 6]

problem = cp.Problem(objective_function, constraints)
result = problem.solve(solver=cp.SCS, verbose=True)

# Output results
print("Optimal value of x1:", x1.value)
print("Optimal value of x2:", x2.value)
print("Optimal value of x3:", x3.value)
print("Optimal value of the objective function:", result)

# # Create the problem and solve
# problem = cp.Problem(objective_function, constraints)
```

```
# result = problem.solve()

# # Output results
# print("Optimal value of x1:", x1.value)
# print("Optimal value of x2:", x2.value)
# print("Optimal value of x3:", x3.value)
# print("Optimal value of the objective function:", result)
```