基于多模型融合的NIPT检测时点优化与异常判别研究

摘　要

本研究旨在解决无创产前检测（NIPT）在临床应用中面临的关键挑战，即如何针对孕妇个体特征（尤其是BMI）优化检测时点，并为女胎建立准确的染色体异常判别模型。研究围绕四个核心问题，构建了一系列由浅入深、层层递进的数学模型。

针对问题一，为探究孕周、BMI等生理指标对男胎Y染色体浓度的影响，本研究从数据可视化出发，构建并比较了多元线性回归模型与含非线性及交互项的多元回归模型。结果表明，引入孕周的二次项和孕周与BMI的交互项能显著提升模型解释力，准确量化了BMI对Y染色体浓度的负向影响以及孕周的非线性正向效应。

针对问题二，为给不同BMI水平的孕妇推荐最佳检测时点，本研究首先依据世界卫生组织（WHO标准对孕妇进行BMI分组，并运用IQR方法进行数据清洗。随后，利用问题一建立的优化回归模型 ，求解各组Y染色体浓度达到临床阈值（4%）的理论孕周，并结合安全裕度，给出了以风险最小化为目标的初步NIPT时点建议。

针对问题三，为构建更全面的多因素优化决策模型，本研究引入了K-均值聚类算法进行数据驱动的孕妇分群，并建立了最小化期望风险决策模型。该框架的核心是采用梯度提升决策树（GBDT预测在多维特征（身高、体重、年龄等）下的Y染色体浓度达标概率。最后，通过蒙特卡洛模拟对检测误差进行敏感性分析，确保了推荐时点的鲁棒性。

针对问题四，为解决女胎染色体非整倍体异常的判别问题，本研究建立了一套完整且严谨的机器学习流程。在采用SMOTE算法处理样本不平衡问题和GroupShuffleSplit策略防止数据泄露的基础上，比较了逻辑回归、随机森林和支持向量机等多种分类器。最终，以临床应用中至关重要的召回率为首要指标，选择了优化后的逻辑回归模型作为最终判别工具，该模型在测试集上实现了70%的敏感性和75.8%的特异性，为临床辅助诊断提供了客观依据。

关键词：　无创产前检测；数学建模；多元回归；期望风险最小化；梯度提升决策树；逻辑回

一、 问题重述

1.1　问题背景

无创产前检测（NIPT）作为一种基于孕妇外周血浆中胎儿游离DNA进行检测，并结合生物信息分析以评估胎儿患病风险的新型产前筛查技术，已成为筛查胎儿染色

体异常的重要手段[1]。然而，临床实践表明，孕妇身体质量指数（BMI）与胎儿 DNA 浓度呈负相关，高 BMI 是导致检测失败的首要因素[2]。由此带来的检测延迟，不仅增加孕妇焦虑，也可能错过最佳干预时机。因此，针对高 BMI 等个体特征，如何确定最佳检测时点以平衡成功率与风险，是 NIPT 临床应用中一个亟待解决的现实问题。

参考文献：

【1】 蓝丹. 你了解无创 DNA 产前检测吗[J]. 家庭医学(下), 2015(6): 57.

【2】 薛莹, 丁洁, 贺权泽, 等. 孕妇年龄、孕周及体重指数对孕妇外周血中胎儿游离 DNA 比例的影响[J]. 中国产前诊断杂志(电子版), 2019, 11(1): 20-23.

1.2 具体问题重述

问题一: 根据附件数据显示的男胎孕妇的孕周数、BMI 等生理指标与 Y 染色体浓度的关系，建立数学模型，分析孕周数、BMI 等指标对 Y 染色体浓度的影响，并对模型的合理性与显著性进行检验。

问题二: 临床研究认为，BMI 是影响 Y 染色体浓度达标时间（浓度首次达到或超过 4%的最早孕周）的关键因素，对男胎孕妇的 BMI 进行科学、合理的分组，并为每个分组确定一个最佳 NIPT 时点。要求该时点的选择以潜在风险最小化为目标，同时需评估检测误差对结果可能产生的影响。

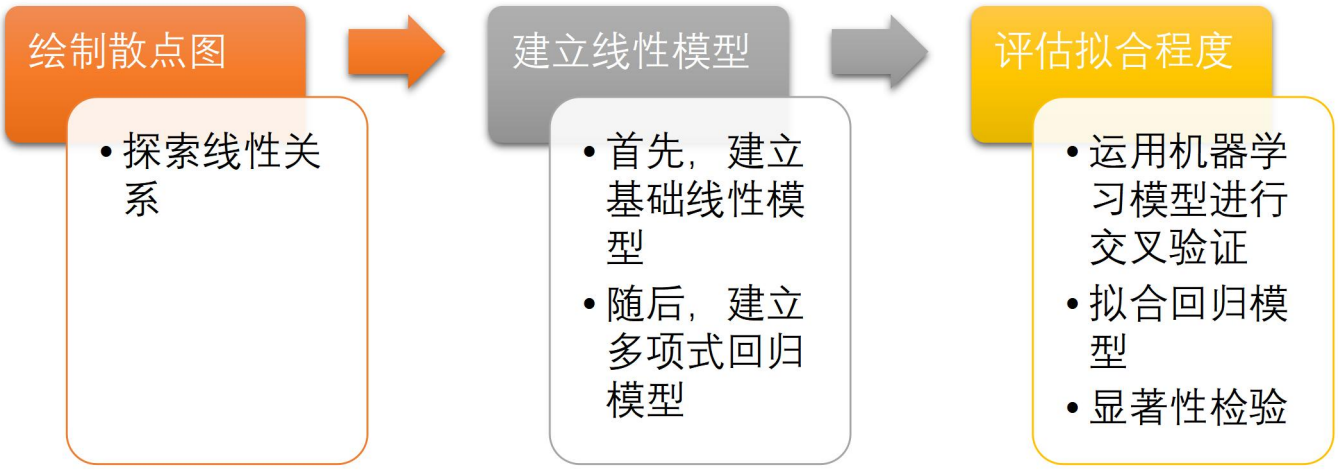
问题三: 考虑到 Y 染色体浓度达标时间是身高、体重、年龄等多重因素共同作用的一个复杂过程。在前问的基础上，根据所有相关因素、可能的检测误差以及 Y 染色体浓度达标的概率，构建一个更全面的优化模型，再次对孕妇进行 BMI 分组并推荐各组的最佳 NIPT 时点，目标仍是最小化孕妇的潜在风险，并分析检测误差的敏感性。

问题四: 由于女胎异常的判定无法依赖 Y 染色体。请利用附件中女胎的数据，综合考虑 X 染色体以及 21 号、18 号、13 号这几条关键染色体的 Z 值、GC 含量、读段数等多种测序质量控制指标，建立一个多维度的判别模型，以区分出染色体非整倍体异常的女胎，并详细阐述你的判定方法和依据。

二、 问题分析

2.1 问题一的分析

对于问题一，要求我们研究胎儿 Y 染色体浓度与孕妇孕周数、BMI 等指标之间的关系，我们构建了一个（）数学模型，来描述 Y 染色体浓度如何随上述多个因素的变化而变化。由于生物学过程很少是完美的直线关系。因此, 我们设计了一条由浅入深的分析路径首先，我们根据已知数据绘制了散点图，探索了出其中的的线性关系，其次，我们建立了一个基础的线性模型，我们将这个模型作为后续所有复杂模型的“参照物”；随后，我们又逐步放宽了模型的限制，建立了多项式回归模型并在模型中手动添加自变量的高次项来拟合曲线关系，使模型越来越贴近复杂的现实情况；最终，我们采用了先进的机器学习模型进行交叉验证，将这些回归模型进行拟合，最后通过显著性检验来评估模型的拟合程度，进而确定孕妇孕周数、BMI 等指标对 Y 染色体浓度的影响



2.2 问题二的分析

对于问题二，要求我们根据男胎孕妇的 BMI 制定一个最优的无创产前检测（NIPT）时点策略使孕妇的潜在风险最小。我们根据世界卫生组织（WHO）的成人 BMI 分级标准进行 BMI 分类，使用 IQR 方法识别并处理异常值；其次我们建立了交互项的多项式回归模型来捕捉 Y 浓度随孕周的非线性增长、BMI 的负面影响、BMI 对孕周增长效应的削弱作用；随后，我们设置 Y 染色体浓度的阈值求解出了各 BMI 分组的理论达标孕周；最终，增加了一个安全裕度旨在保证将风险最小化和模型的稳健性的同时给出“最佳 NIPT 时点”，检验误差敏感性分析，并对之进行实用性评估。

2.3 问题三的分析

对于问题三要求我们根据男胎 Y 染色体浓度达标时间所受的多种因素的影响综合分析出使孕妇潜在风险最小的 NIPT 时点，同时给出检测误差与结果间的关系。为此，我们首先建立了期望风险模型将风险函数可视化；随后，通过数据驱动的 BMI 聚类分析将数据进行标准化处理同时运用轮廓系数选择最佳 K 值，并将聚类结果可视化；并对各组建立 Y 染色体浓度预测模型。随后，我们同时建立多项式回归模型和随机森林模型，选择二者间更优的模型，从而最小化期望风险；接着，我们建立了预测模型，判断其预测结果，同时使用 sigmoid 函数建模概率，找到最小风险对应的时点。创建最佳时点优化模型，为每个 BMI 组建立模型并优化检测时点；接着，我们建立了多因素机器学习模型，对数据进行标准化处理，；最后，我们通过蒙特卡洛敏感性分析来进行误差评估，对多因素模型进行敏感性分析，最终得出使孕妇潜在风险最小的 NIPT 时点。

2.4 问题四的分析

我们对原始数据进行处理，剔除不符合标准的样本，对数据进行异常值处理，增强模型的鲁棒性。选取相关指标作为特征，对数据集进行划分防止数据泄露，对这些数据进行标准化处理，采用 SMOTE 算法对训练集进行采样，最终使正常值和异常值比例呈 1:1。我们选取逻辑回归，随机森林，支持向量机三种分类模型进行比较，筛选出逻辑回归在召回率上表现最佳。最终进行概率校准和判定阈值优化，确定逻辑回归为最终判定模型。

三、 模型假设

1. 数据代表性假设： 附件所提供的数据样本能够真实、准确地反映研究对象群体的总体特征，
无系统性偏差。
2. 数据准确性假设： 数据中的各项生理指标（如 BMI、年龄）和测序指标（如 Y 染色体浓度、Z
值）的测量和记录是准确可靠的。
3. 因素独立性假设： 除模型中明确考虑的变量外，其他未纳入模型的潜在因素（如孕妇的特殊
病史、生活习惯、种族差异等）对研究结果的影响是随机的或可以忽略不计的。
4. 风险单调性假设： 在问题三的期望风险模型中，假设与延迟诊断相关的潜在风险（如孕妇焦
虑、临床干预选项减少等）是随孕周增加而单调递增的。
5. 检测误差分布假设： 在敏感性分析中，假设 NIPT 检测结果的随机误差服从均值为 0 的正态分

四、 符号说明

符号	说明
X	输入的特征向量，X=[x1 ,x2 ,...,x16]
xi	第 i 个检测指标，如 X 染色体 Z 值、13 号染色体 GC 含量等
Y	目标变量，表示女胎是否异常，Y∈{0,1} (0:正常, 1:异常)
Y^	模型预测的分类结果
$P(Y = 1$	\mathbf{X})$
TP,FP	真阳性（True Positive），假阳性（False Positive）
TN,FN	真阴性（True Negative），假阴性（False Negative）
Recall	召回率（敏感性），Recall=TP+FNTP
Specificity	特异性，Specificity=TN+FPTN
Precision	精确率（阳性预测值），Precision=TP+FPTP
F1−score	F1 分数，精确率和召回率的调和平均值
AUC	ROC 曲线下面积 (Area Under Curve)

五、 模型的建立与求解

5.1 问题一模型的建立与求解

为分析孕周数、孕妇 BMI 等指数与 Y 染色体浓度的关系，我们首先从孕周数、MBI 等一系列指数与 Y 染色体浓度的散点图出发，综合相关文献的研究结果，建立了线性拟合模型、多项式拟合模型等多种模型 ，得出孕妇孕周数、孕妇 BMI 等指数与 Y 染色体浓度的关系，最终对其关系完成了显著性检验。

5.1.1 基于散点图的初步判断

为了更直观地分析各项指标与 Y 染色体浓度变化的变化关系，我们利用 python 分别画出了 Y 染色体的 Z 值、染色体的非整倍体编码、重复读段的比例、胎儿是否健康、身高、在参考基因组上比对的比例、年龄、18 号染色体的 Z 值、体重、检验抽血次数、孕周数、孕妇 BMI、唯一比对的读段数、原始读段数、X 染色体的浓度、与 Y 染色体浓度变化的散点图，共计产生 15 张散点图。其中部分散点图如图 1 所示：

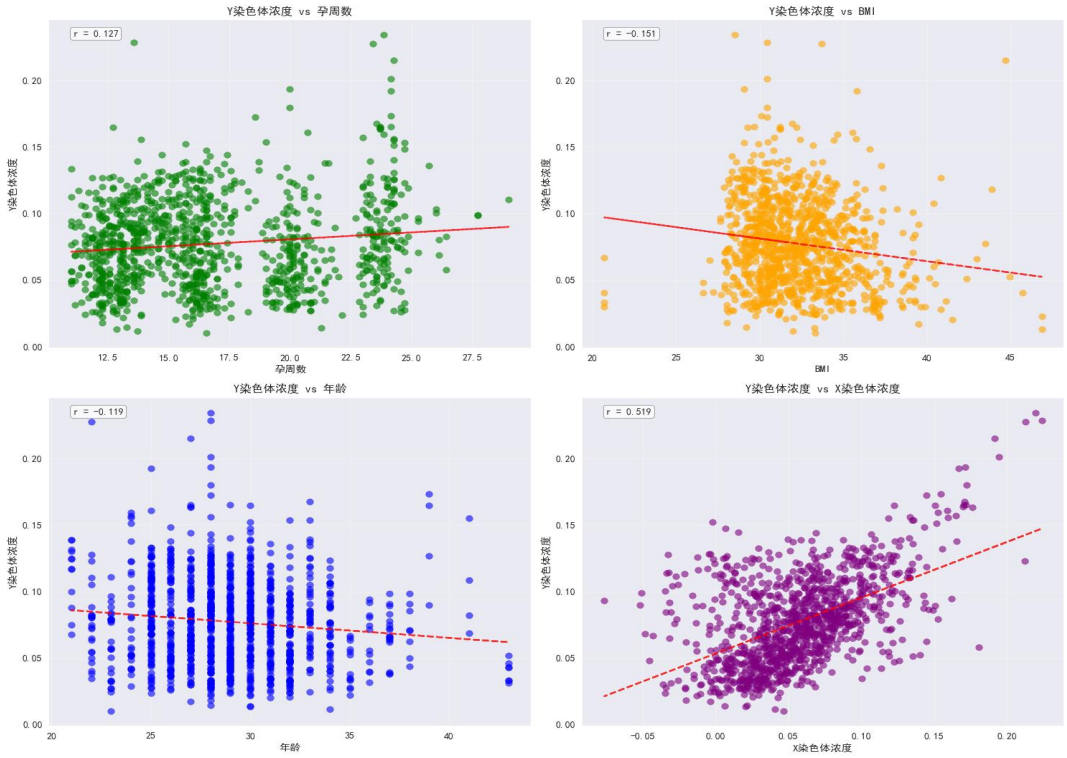


图 1 x 染色体浓度、孕妇 BMI、孕周天数、年龄与 y 染色体浓度的关系

通过对以上数据得出 Y 浓度随着 X 染色体浓、孕周天数的增加而增加，随着孕妇 BMI、年龄的增加而减少为了进一步探究各指标与 Y 染色体浓度的相关性，我们运用变量相关热力图来直观展示其相关关系具体关系见图 2

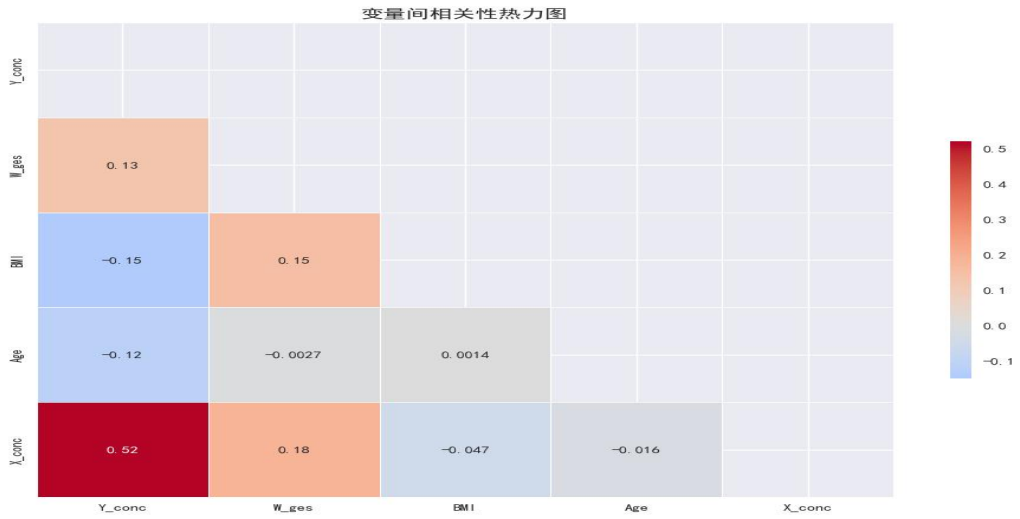


图 2 变量间相关热力图

5. 1. 2 多元线性回归模型的构建与求解

5.1.2.1. 模型的建立

结合以上分析我们选取了多元线性回归模型,同时设x_1，x_2…，x_i为孕周数、BMI 指数、孕妇年龄、x 染色体浓度等自变量，a_1，a_2…，a_i为 h 回归系数，ε为随机项误差，如下

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_ix_i + \varepsilon(i = 1,2,3\cdots)$$

进行拟合

5.1.2.2. 模型的的求解

采用最小二乘法（Ordinary Least Squares, OLS）对模型进行参数估计，即找到一组参数$\beta_0, \beta_1, ..., \beta_\ell$ ，使得残差平方和（Sum of Squared Errors）最小化

$$SSE = \sum_{i=1}^{n} (CY_i - \widehat{CY}_i)^2 = \sum_{i=1}^{n} \left[CY_i - (\beta_0 + \beta_1 W_{ges,i} + \cdots + \beta_6 X_{conc,i})\right]^2$$

其中 n 为样本数量，CY_i为第 i 个样本的实际 Y 染色体浓度,$\widehat{CI}_i$为模型预测值，通过对 SSE 函数关于每个$\widehat{\beta}_j$求偏导并令其为 0 得到一组正规方程组。这些方程组的解即为最小二乘估计量：$\hat{\beta} = (X^TX)^{-1}X^TY$。其中 Y 是因变量 CY 的观测值构成的 $n \times 1$ 列向量。X 是设计矩阵，其维度为 $n \times (p+1)$，其中 p 为自变量数量（此处 p=6）X 的第一列为全 1 向量（对应截距项），后续列分别为各自变量的观测值。如所示

我们运用 python 求得多元线性回归方程为：

$$y\ 染色体浓度 = 0.12542 + 0.00046 \times 孕周数 - 0.00153 \times BMI - 0.00102 \times 年龄$$
$$+ 0.40445 \times X\ 染色体浓度$$

5.1.2.3. 统计显著性检验

在获得系数估计值后，我们对模型整体进行 F 检验，评估其对因变量的解释能力是否显著。同时，对每个回归系数进行 t 检验，判断其在统计学上是否显著异于零，从而确定每个自变量对 Y 染色体浓度的影响，即 p 值是否小于 0.05

5.1.2.4. 模型评估与诊断

我们计算决定系数 R^2 和 $\bar{R}^2$，来确定模型对 Y 染色体浓度变异的解释比例。R^2值越高，表示模型拟合越好。为了初步检查线性假设是否成立，以及是否存在异方差性或多重共线性问题，我们又进行了残差分析（绘制残差图）如图 3 图 4 基准模型残差图所示：

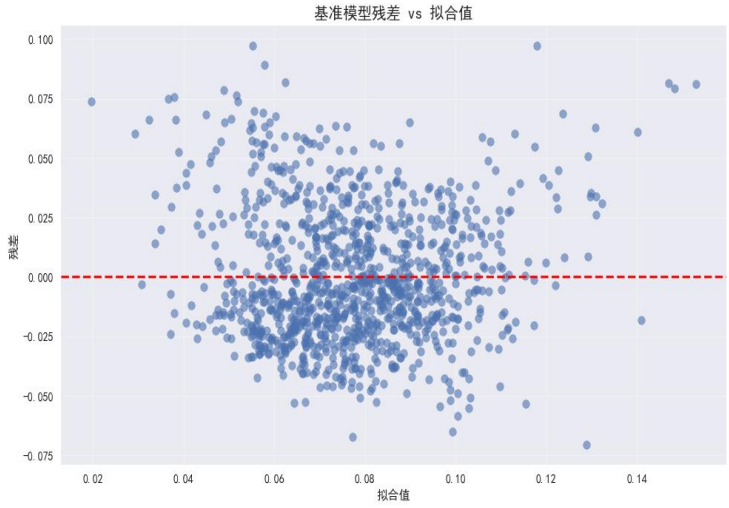


图 3 基准模型残差 vs 拟合值

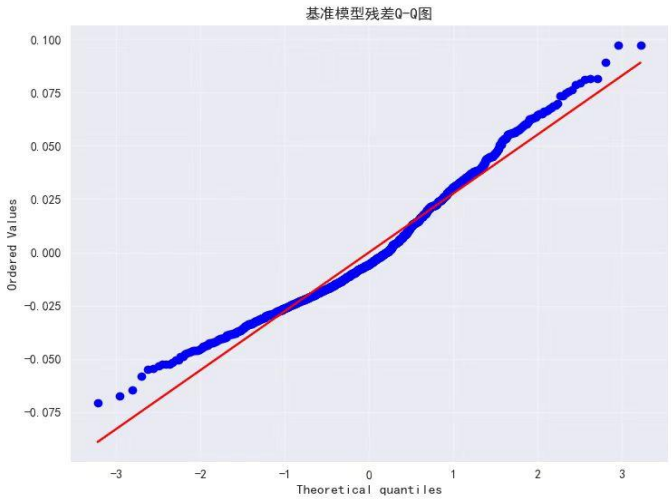


图 4 基准模型残差图

5.1.3 带非线性项与交互项的多元回归模型的建立与求解

5.1.3.1. 带非线性项与交互项的多元回归模型的建立

由于生物学过程很少是完美的直线关系，拟合程度不是很好，可以推断出并不是简单的线性关系，所以我们在基准模型的基础上，引入关键变量的高次项和交互项。特别是孕周的二次项和孕周与 BMI 的交互项：

$$CY = \beta_0 + \beta_1 W_{ges} + \beta_2 W_{ges}^2 + \beta_3 BMI + \beta_4 (W_{ges} \cdot BMI) + \beta_5 Age + \beta_6 UniqueReads + \beta_7 GCContent + \beta_8 X_{conc} + \epsilon$$

进行拟合

W_{ges}^2是孕周的平方项，用于捕捉 Y 浓度随孕周变化的非线性趋势（例如抛物线型增长）。$(W_{ges} \cdot BMI)$是孕周与 BMI 的交互项，用来探究，当 BMI 水平不同时，孕周对 Y 浓度的影响斜率的变化。

5.1.3.2. 带非线性项与交互项的多元回归模型的求解

我们采用了与基准模型相同的方法，用最小二乘法对扩展模型进行参数估计

5.1.3.3. 统计显著性检验

最后，我们再次进行了 F 检验和 t 检验，来研究新增的非线性项和交互项的显著性。若它们显著，则说明新模型对数据的解释力更强，反之则更弱；最后，我们进一步进行残差分析、正态性检验，如图 5 图 6 所示

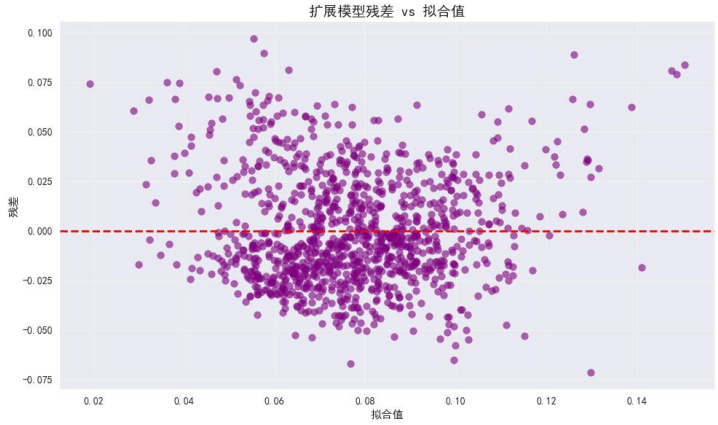


图 5 扩展模型残差 vs 拟合值

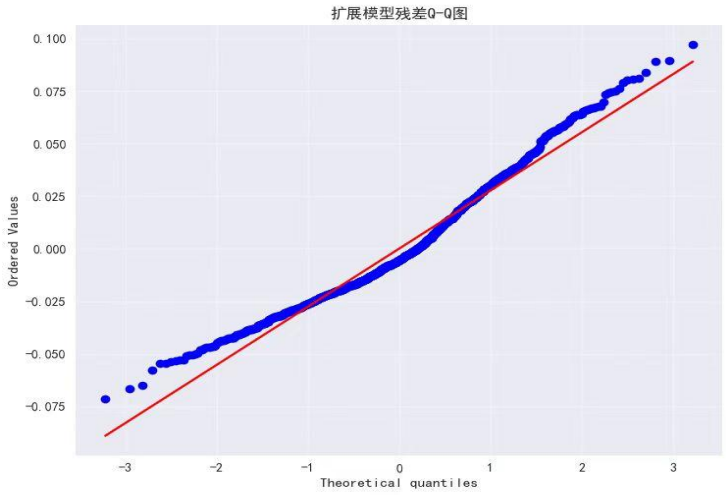


图 6 扩展模型残差

5.1.4 结果评估与初步结论

通过对上述两个模型的R^2和调整R^2，和统计显著性检验的结果的分析，我们对两个模型性能进行了比较，如图 5 所示，发现新模型对数据的解释力更强，效果更优。

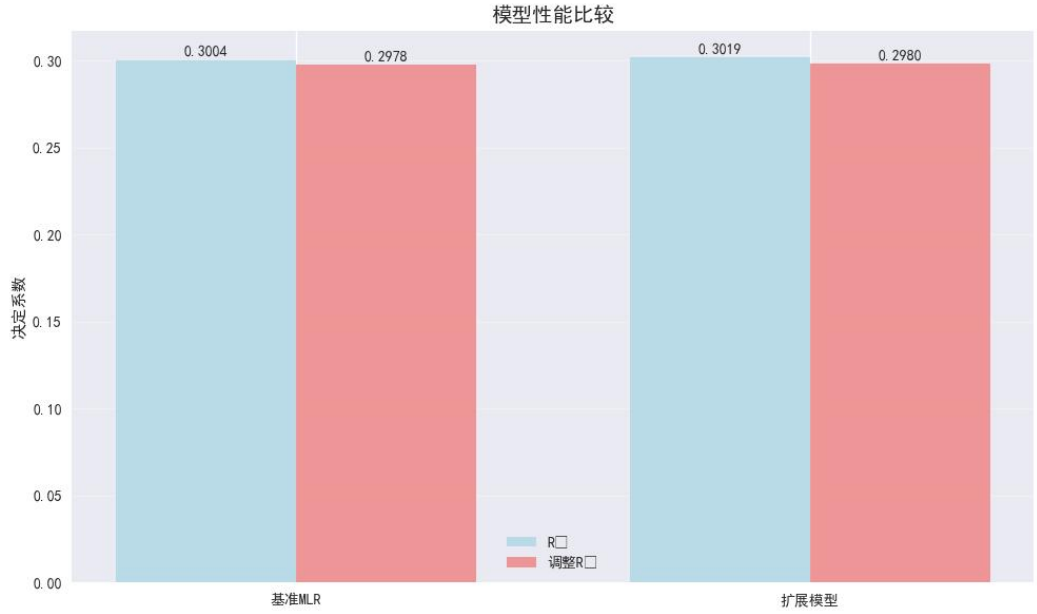


图 7 模型性能比较

5.2 问题二模型的建立与求解。

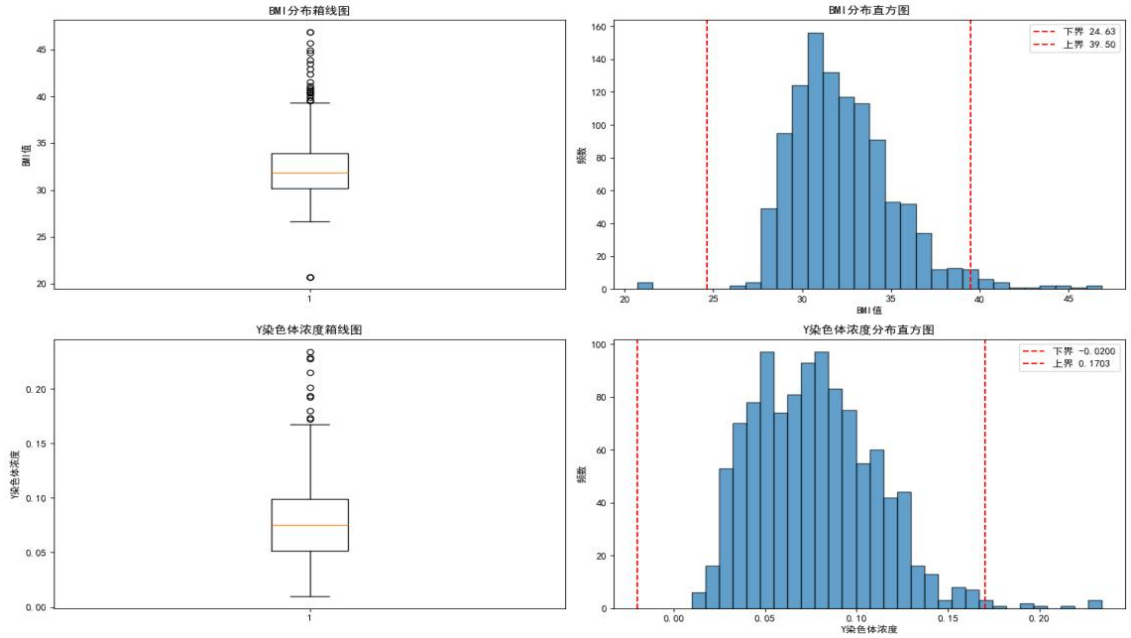
5.2.1 数据预处理与 BMI 分组

本研究的分析对象为筛选自合并数据集的 1082 条男性胎儿样本记录。为实现具有临床参考价值的分类，我们直接参考世界卫生组织（WHO）的 BMI 分级标准，将孕妇群体划分为偏瘦(0，18.5)、正常[18.5，25.0)、超重[25.0,30.0)、肥胖[30.0,∞)四个组别。各组别的界定及其样本统计特征详见表 1

BMI 分组 统计指标	count	mean	median	min	max
正常	4	20.70	20.70	20.70	20.70
肥胖	857	33.18	32.71	30.00	46.88
超重	221	29.04	29.14	26.62	30.00

从表中可以看出数据中偏瘦孕妇的样本量为 0，正常孕妇的样本量为 4，肥胖孕妇的样本量为 857，超重孕妇的样本量为 221，由肥胖群体的方差中得出数据中可能存在 BMI 极高的异常值，为防止后续建模中产生较大误差，我们将对 BMI 和 Y 染色体浓度数据进行了异常值处理，通过 python 编写代码进行可视化，查看是否存在异常值，具体结果见下图 7

图 8 BMI 分布和 Y 染色体浓度箱线图和直方图



从图中可以看出数据存在异常值，为求出异常值占总体数据的比例，我们利用IQR方法对异常值进行检测，最终求出BMI异常值的数量为26， Y染色体浓度异常值为10，随后，分别采用保留、删除、缩尾处理的方法对异常值进行处理，通过对比这三种方法，最终采取了删除异常值的方法重新对数据进行统计，并与处理前的数据进行对比，具体的数据见下表:

BMI分组 统计指标	删除前样本数	删除后样本数	删除前均值	删除后均值	样本保留率(%)
正常	4	0	20.70	0	0
肥胖	857	828	33.18	32.96	96.6
超重	221	219	29.04	29.04	99.1

表 1 删除前与删除后的对比

从表中可以看出，处理结束后，正常BMI指数的孕妇为0，肥胖和超重都有一定数量的减少，符合上述对异常值的处理，通过这一操作，为后来建立数学模型奠定了基础。

5.2.2 模型的建立

5.2.2.1 多元回归模型的建立

通过问题一的建模，可以了解到Y染色体浓度随孕周呈非线性增长，且此趋势同时受到孕妇BMI的影响，对此我们构建一个含二次项与交互项的多元回归模型来捕捉该动态关系。模型具体形式如下：

$$Y=\beta0+\beta1x1+\beta2(x1)2+\beta3(x2)+\beta4(x2)2+\beta5(x1x2)+\epsilon$$

其中x1为孕周，x2为BMI，x1x2为x1和x2的交互项，ϵ为随机误差项

在模型求解部分，可以求出每个回归系数，将其带回到模型中，就可以得到含有二次项与交互项的多元回归模型。

对于如何求解各个BMI分组中‘达标孕周’，我们根据该多项式回归模型以及Y染色体浓度的阈值，将BMI值代入，就可以得出各个BMI分组的“达标孕周”。这又涉及到一个新的问题，BMI是一个范围，如何选定合适的BMI值将其带入到回归模型中，我们设定运用每个区间中的临界值进行计算，最终得到“达标孕周”的具体数值，考虑到实际操作中，通常以周为单位，我们将上面计算得到的数值进行向上取整。为进一步降低风险，需要增加一个安全裕度，在向上取整的基础上再加上一周，

即最佳 NIPI 检测时点的计算公式为向上取整时点+1，这样不仅是风险大大减少，还减少了由于数据等原因而造成的误差，使得建立的模型能够更好的推广到实际生活中。

为研究该模型的合理性，我们需要进行显著性检验，通过结果可以发现是否要对模型进行改进处理

5.2.2.2 改进多元回归模型

为解决多项式模型系数不显著问题，对问题进行更深度的探索。使用岭回归、Lasso 回归、随机森林等，解决回归中的过拟合和多重共线性等问题，对比多个机器学习方法，计算模型的交叉验证 R2 值，可以选择最优的机器学习模型，对上述多项式回归模型进行改进。最终得到较高的拟合优度值，使得该模型对数据具有极强的解释力，模型稳定力和泛化能力也随之增强。改进模型后，再按照上述的过程，求解最佳检测点。

5.2.3 模型的求解

对于多元回归模型，采用最小二乘法的方式进行求解，由于问题一中，已经对最小二乘法的计算原理进行详细解释，所以，在此处不对求解过程进行赘述，最终我们可以得到的多元回归模型为：

$Y=\beta 0+\beta 1x1+\beta 2(x1)2+\beta 3(x2)+\beta 4(x2)2+\beta 5(x1x2)+\epsilon$

Y 染色体浓度 = 0.116245 - 0.000095*孕周平方 + 0.000004*孕周三次 - 0.000034*BMI 平方

通过最小二乘法回归分析报告中，可以得到拟合优度指标 R2=0.018，拟合程度较差，且系数的 P 值均大于 0.05，这说明模型中的所有变量均是不显著的。故而要对模型进行进一步改进。

利用阈值以及用各个 BMI 分组的最大数可以找到目标浓度的最优孕周，将结果进行向上取整，最后求出最佳时点，得到的结果见下表：

第四步：最佳NIPT时点汇总

	BMI上限	理论孕周	向上取整	最佳时点
偏瘦	18.4	6.22	7.0	8.0
正常	24.9	9.38	10.0	11.0
超重	29.9	11.69	12.0	13.0
肥胖	40.0	14.26	15.0	16.0

上面多项式回归模型中，发现了多重共线性问题，预测稳定性较差，需要对模型进行改进，我们采用六种不同的建模策略及逆行系统性对比，结果如下表所示：

模型性能比较汇总

模型	CV_R²_均值	CV_R²_标准差	模型复杂度	可解释性
原多项式回归	0.045267	0.000000	高	高
梯度提升	-0.281676	0.257717	高	中
随机森林	-0.293681	0.256076	高	中
Lasso回归	-0.314863	0.229683	低	高
SVR	-0.325749	0.279099	中	低
岭回归	-0.366456	0.350852	中	高

由表中数据可以看出，还是原多项式回归的性能最好，$R^2 = > 0$，标准差为 0，模型极其稳定，可解释性高，符合医学研究的需求，模型复杂度虽然标记为高，但是相对于随机森林模型和梯度提升模型实际上更加简单。其他模型表现差的原因可能式数据量不足、没有找到最优参数导致的

5.2.4 结论

通过进行多项式回归模型，可以得出回归方程，发现拟合优度较低，但通过岭回归等方法处理比对后，得出原多项式多项式回归模型更优，在 0.04 的阈值下做出最佳 NIPT 时点分别为 8、11、13、16。基于敏感性分析结果，本研究得出明确结论：Y 染色体浓度检测的最佳时点建议为孕 16 周，且该建议对不同 BMI 分组孕妇和不同检测阈值（3.5%-5.0%）均表现出极强的稳健性。分析显示，当阈值在 3.5%至 4.5%范围内变化时，所有 BMI 分组（偏瘦、正常、超重、肥胖）的建议检测孕周均保持稳定在 16 周，平均时点变化为 0 周，表明推荐方案对检测误差具有高度耐受性，为临床实践提供了可靠且统一的操作标准。

5.3 问题三模型的建立与求解

5.3.1 数据预处理与 BMI 分组

首先我们采用数据驱动的分组方法从数据中筛选出了所有男性胎儿样本，其次采用 K-均值（K-Means）聚类算法以孕妇的核心生理指标，如 BMI 和年龄为聚类特征将男胎孕妇分为“高 BMI 高龄”、“低 BMI 年轻”等具有不同风险特征的群体。最终，通过肘部法则和轮廓系数等方法得出最佳的聚类数量 K。

结果显示，

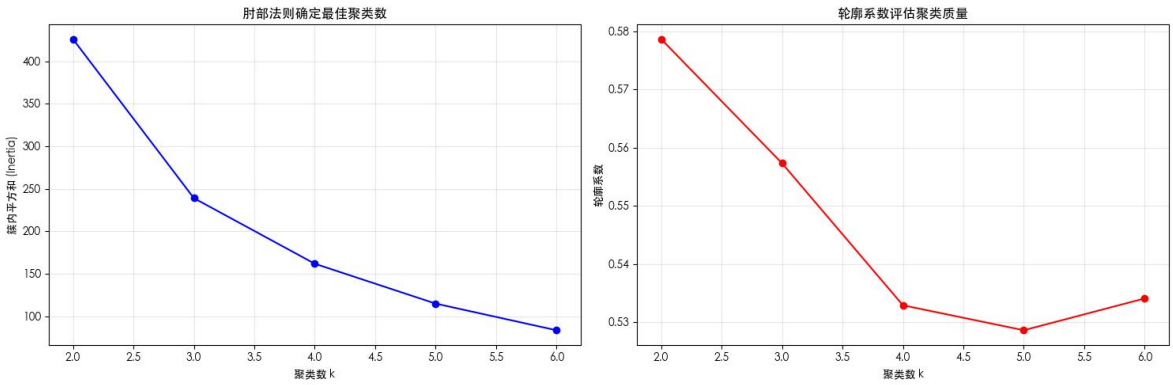
男胎有效样本数: 1082

BMI 范围: [20.7, 46.9]

孕周范围: [11.0, 29.0]

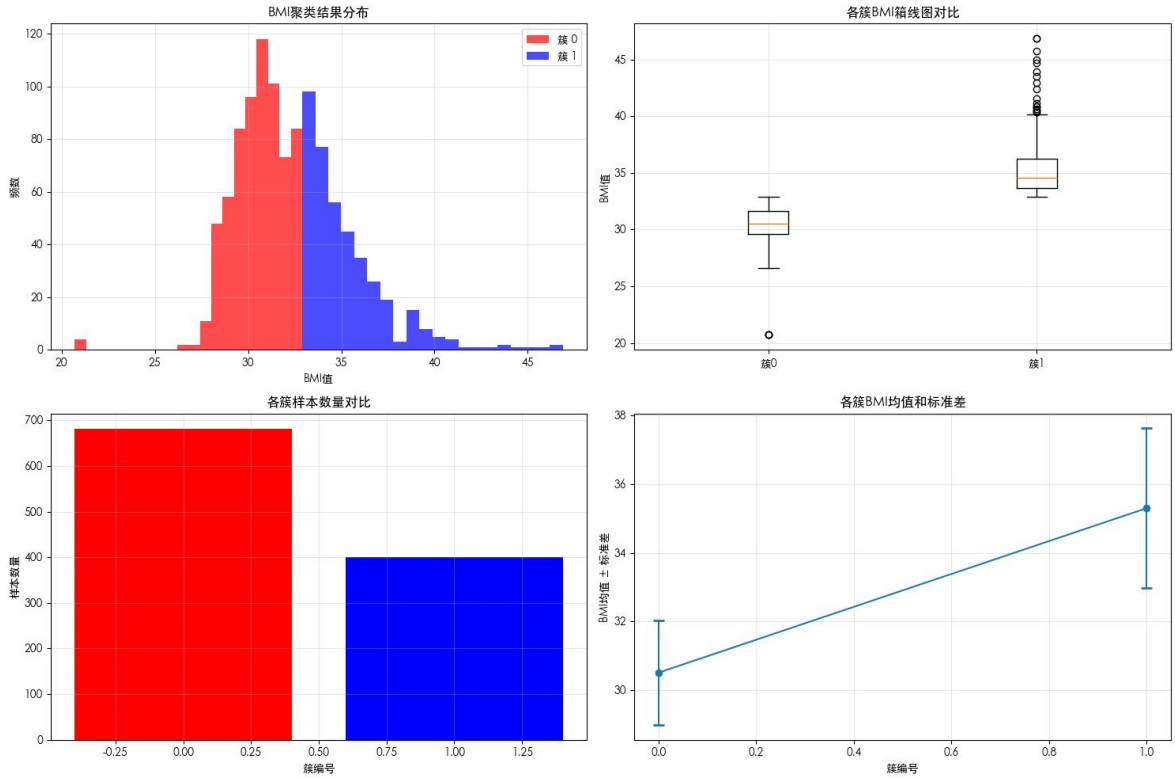
Y 染色体浓度范围: [0.0100, 0.2342]

开始 BMI 聚类分析，样本数: 1082



最佳聚类数: 2

对应轮廓系数: 0.5786322848480934



聚类结果详细统计:

簇 0: 样本数=681, BMI 范围=[20.7, 32.9], 均值=30.51±1.53

簇 1: 样本数=401, BMI 范围=[32.9, 46.9], 均值=35.30±2.33

5.5.3.2 决策目标：最小化期望风险

我们为每个 BMI 分组 k 找到一个最佳检测时点 Wopt(k) ，使得该组的期望风险最小，我们的目的是使之量化为一个明确的数学期望风险函数。在进行检测时，期望风险 E_k(W_{ges})$由两部分构成：

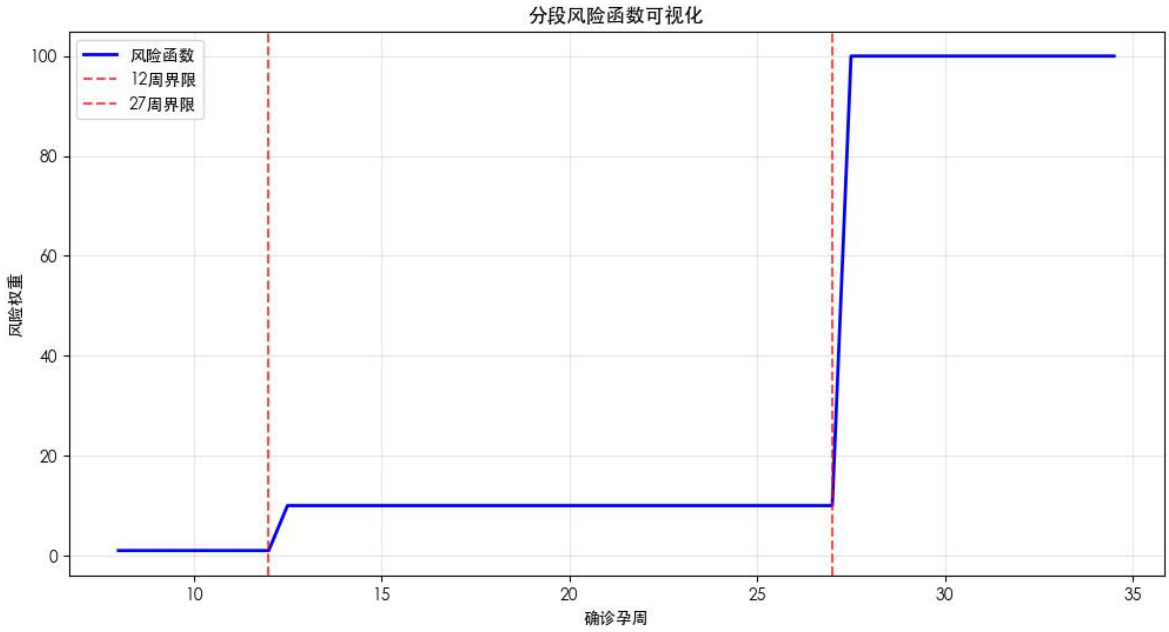
检测成功（达标）：其概率为 Pk (Wges)，此时承担的风险为 R(Wges)。

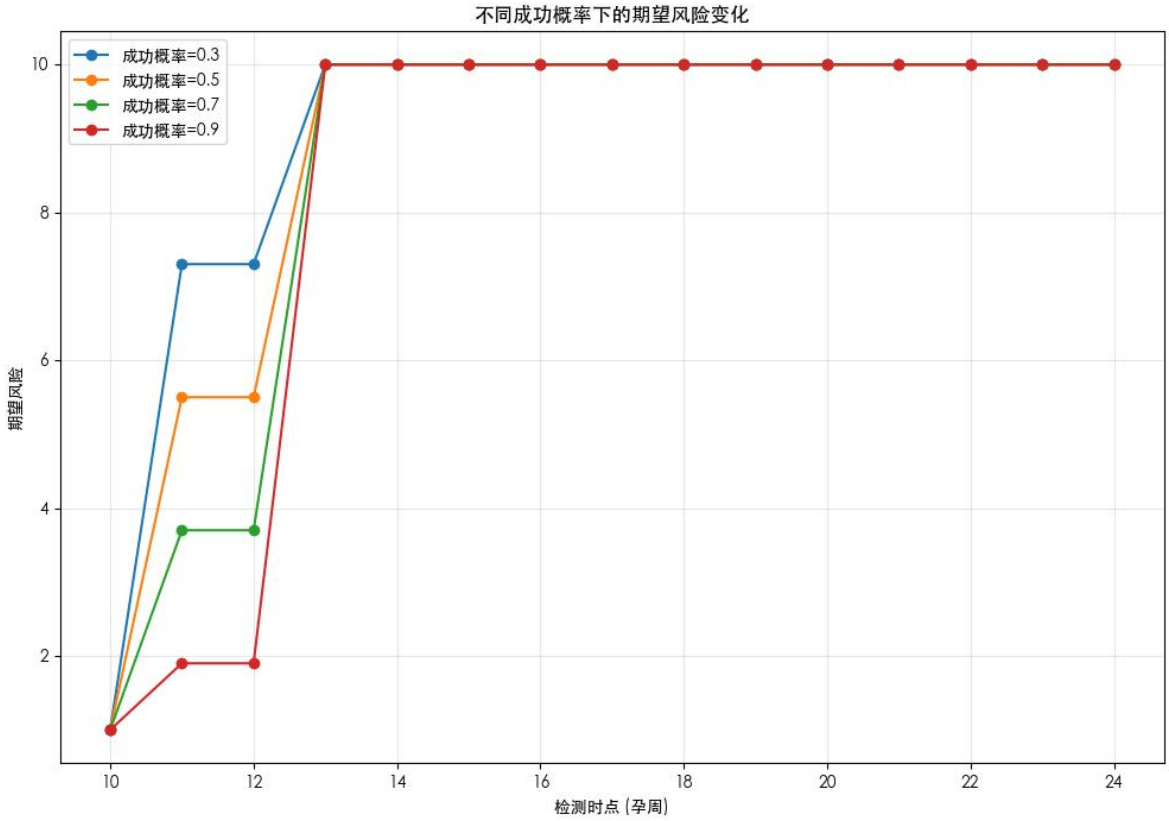
检测失败（未达标）：其概率为 1−Pk (Wges)。失败则意味着需要延迟检测（假设延迟 d 周），从而承担更高的风险 R(Wges +d)

因此，我们的目标函数被定义为：

Ek (Wges)=Pk (Wges)·R(Wges)+(1−Pk (Wges))·R(Wges +d)

求解：为实现最小化期望风险，我们用梯度提升决策树进行求解





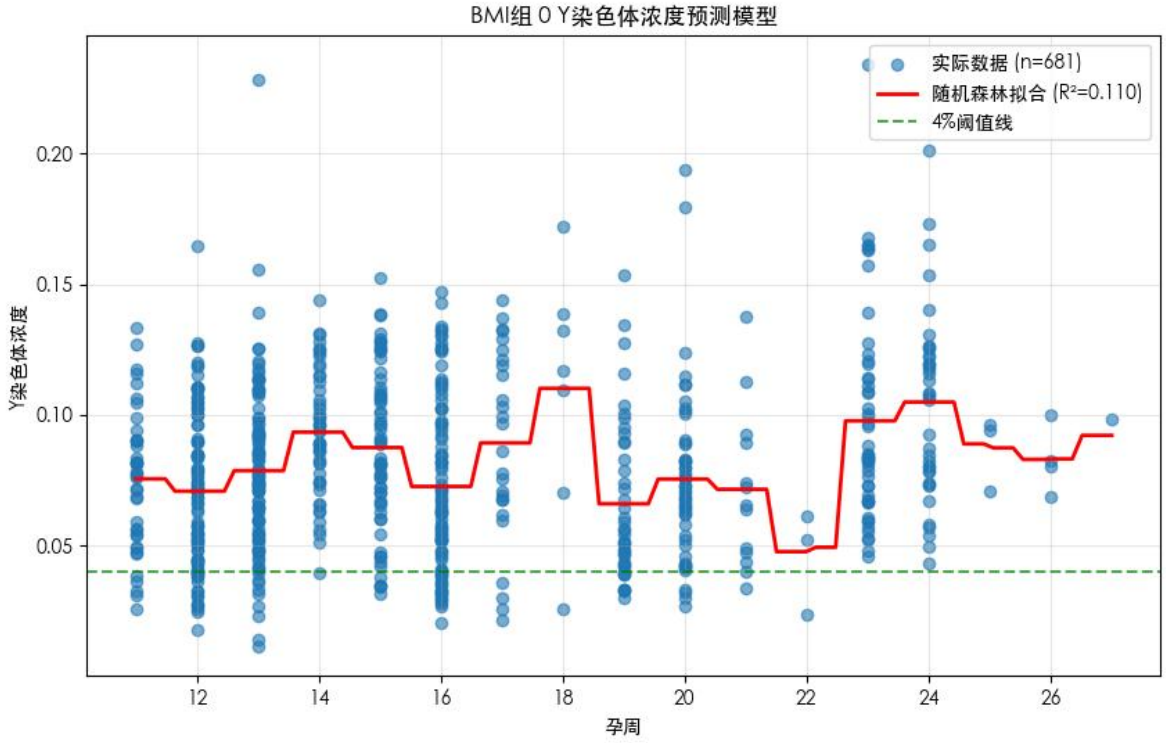
5.3.2 建立决策模型与多因素机器学习预测

5.3.2.1 预测模型

由于影响 Y 染色体浓度的因

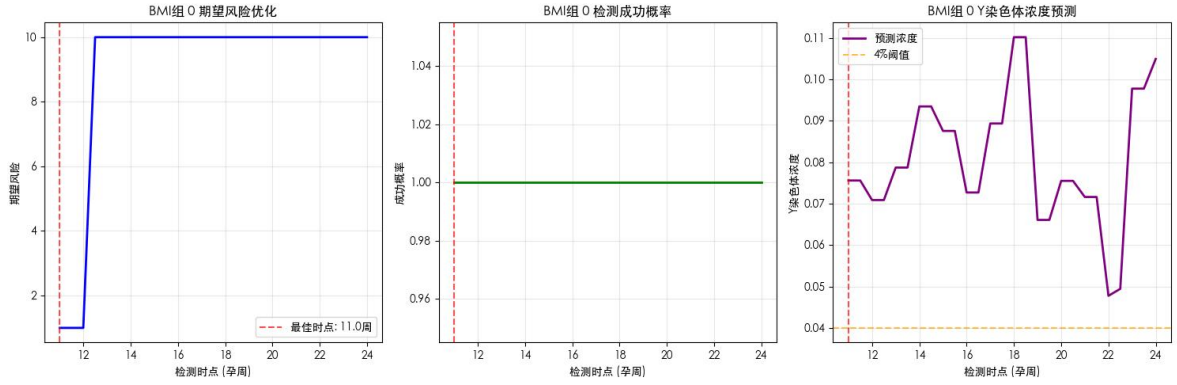
素增多（身高、体重、年龄等）变量间可能存在复杂的非线性关系和交互效应，传统的线性回归模型难以胜任，因此，我们构建了一个机器学习模型，直接预测在给定孕妇特征和孕周下 Y 染色体浓度达标的概率，以有效处理多变量见的非线性与交互

关系。
为各 BMI 组建立预测模型并优化检测时点，建立预测模型
该组样本数: 681
选择的模型: 随机森林, R^2 = 0.1097

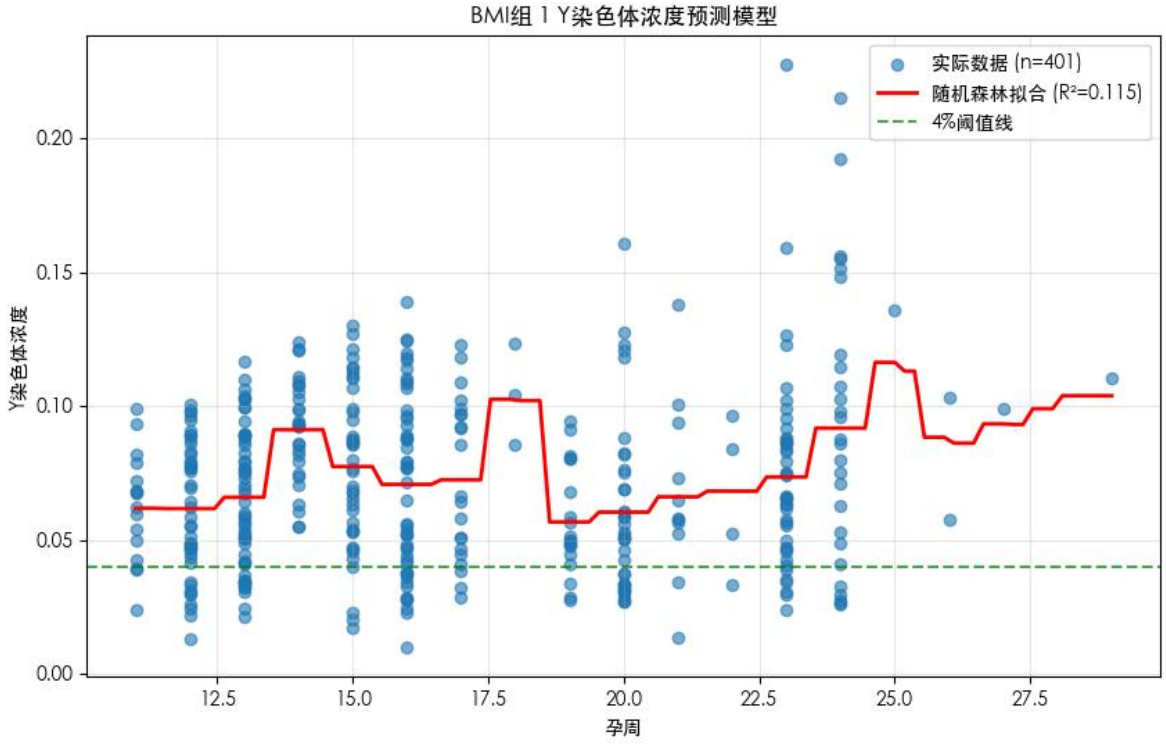


BMI 组 0

最佳检测时点	11.0 周
预期成功概率	1.000
最小期望风险	1.000
最佳检测时点	11.0 周
预期成功概率	1.000
最小期望风险	1.000

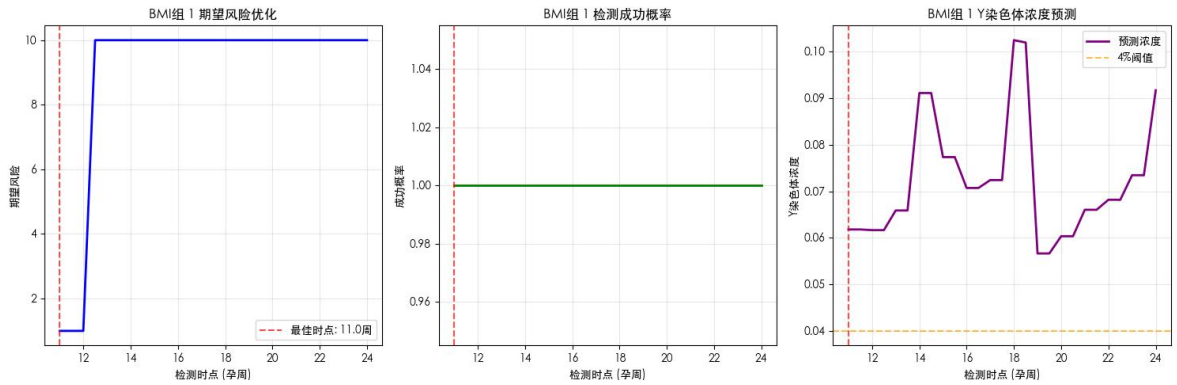


接下来为 BMI 组 1 建立预测模型
该组样本数: 401；选择的模型: 随机森林, R^2 = 0.1147



BMI 组	
最佳检测时点	11.0 周
预期成功概率	1.000
最小期望风险	1.000
最佳检测时点	11.0 周
预期成功概率	1.000
最小期望风险	1.000

0



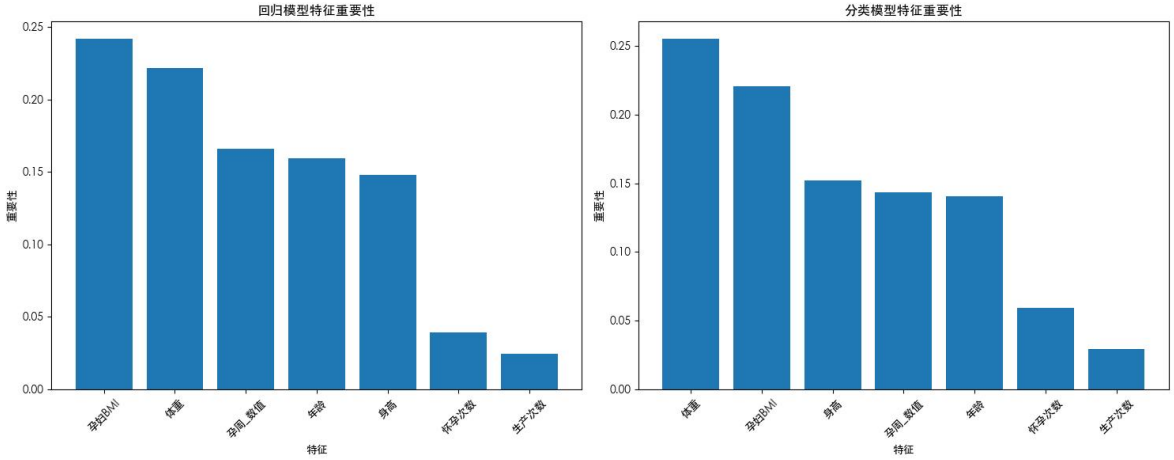
5.3.2.2 决策模型：求解最优时间

为了最小化期望风险，我们首先根据 K-均值划分出的每个分组计算出了各组在任意孕周的平均达标比例，我们将$P_k(W_{ges})$代入期望风险函数$E_k(W_{ges})$。通过寻找所有可能的检测孕周（例如从第 10 周到第 25 周），计算每个孕周对应的期望风险值，最终选取$E_k(W_{ges})$使最小的孕周作为该组的最佳 NIPT 建议时点$W_{opt}^{(k)}$。

$$Wopt(k) \quad = argWges \quad min \quad Ek \quad (Wges \quad)$$

求解：

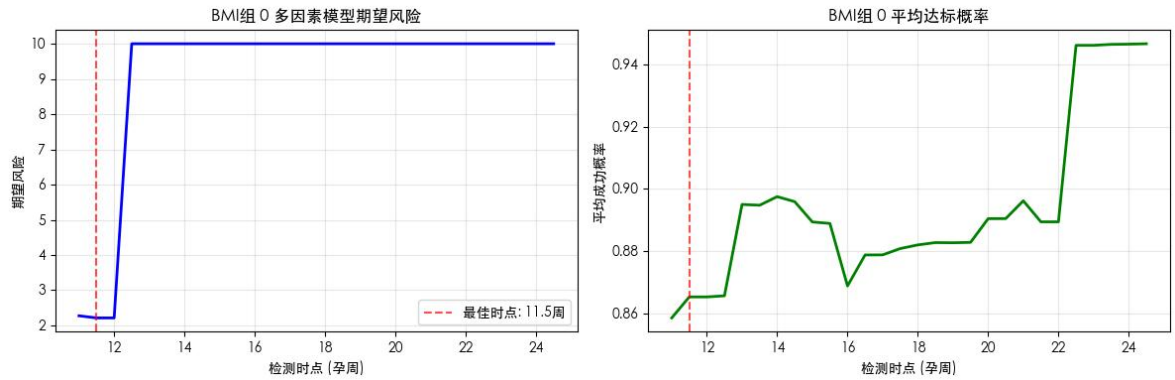
准备多因素分析数据，确定'孕妇 BMI', '孕周_数值', '年龄', '身高', '体重', '怀孕次数', '生产次数'为最终特征列



随机森林回归模型性能	
训练集	R^2 : 0.6899
测试集	R^2 : 0.2423
随机森林分类模型性能	
训练集准确率	0.9457
测试集准确率	0.8664
总体达标率	0.866

使用多因素模型分别优化 BMI 组 0 和组 1 的检测时点

组 0：



该组样本数: 681

多因素模型最佳检测时点: 11.5 周

平均成功概率: 0.865

最小期望风险: 2.213

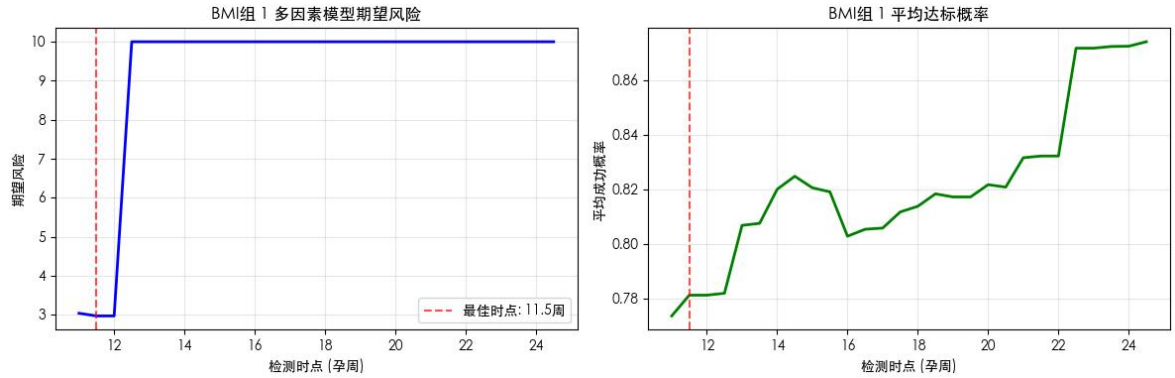
多因素模型最佳检测时点: 11.5 周

平均成功概率: 0.865

最小期望风险: 2.213

组 1：

多因素模型最佳检测时点: 11.5 周
平均成功概率: 0.781
最小期望风险: 2.969
多因素模型最佳检测时点: 11.5 周
平均成功概率: 0.781
最小期望风险: 2.969



5.3.3 蒙特卡洛敏感性分析：检测误差，评估不确定性

为了评估我们的推荐时点在面对现实世界检测技术的固有的不确定性时的稳健性，我们设计并执行了蒙特卡洛敏感性分析 ，假设单次 NIPT 检测的测量误差 ϵ 服从均值为 0、标准差为 σ_e 的正态分布，即 $\epsilon \sim N(0, \sigma_e^2)$ 。

随后，我们针对每个聚类分组(k)及其已确定的最优时点 Wopt (k)进行了数万次随机模拟 。首先根据模型预测的达标概率，为组内孕妇生成一个“真实浓度”状态（达标或不达标），然后通过对该状态叠加随机抽样的检测误差项得到一个“模拟检测值” 接着根据统计所有模拟的平均结果得出了考虑误差影响后的“实际达标比例” Psim (k) 。

求解：

分析 BMI 组 0 和组 1 的检测误差敏感性

BMI 组 0：

原始模型最佳时点: 11.0 周

多因素模型最佳时点: 11.5 周

检测误差标准差: 0.005

原始模型: 模拟成功率=0.977, 模拟风险=1.207

多因素模型: 模拟成功率=0.932, 模拟风险=1.615

检测误差标准差: 0.010

原始模型: 模拟成功率=0.834, 模拟风险=2.494

多因素模型: 模拟成功率=0.761, 模拟风险=3.155

检测误差标准差: 0.015

原始模型: 模拟成功率=0.751, 模拟风险=3.242

多因素模型: 模拟成功率=0.688, 模拟风险=3.804

检测误差标准差: 0.020

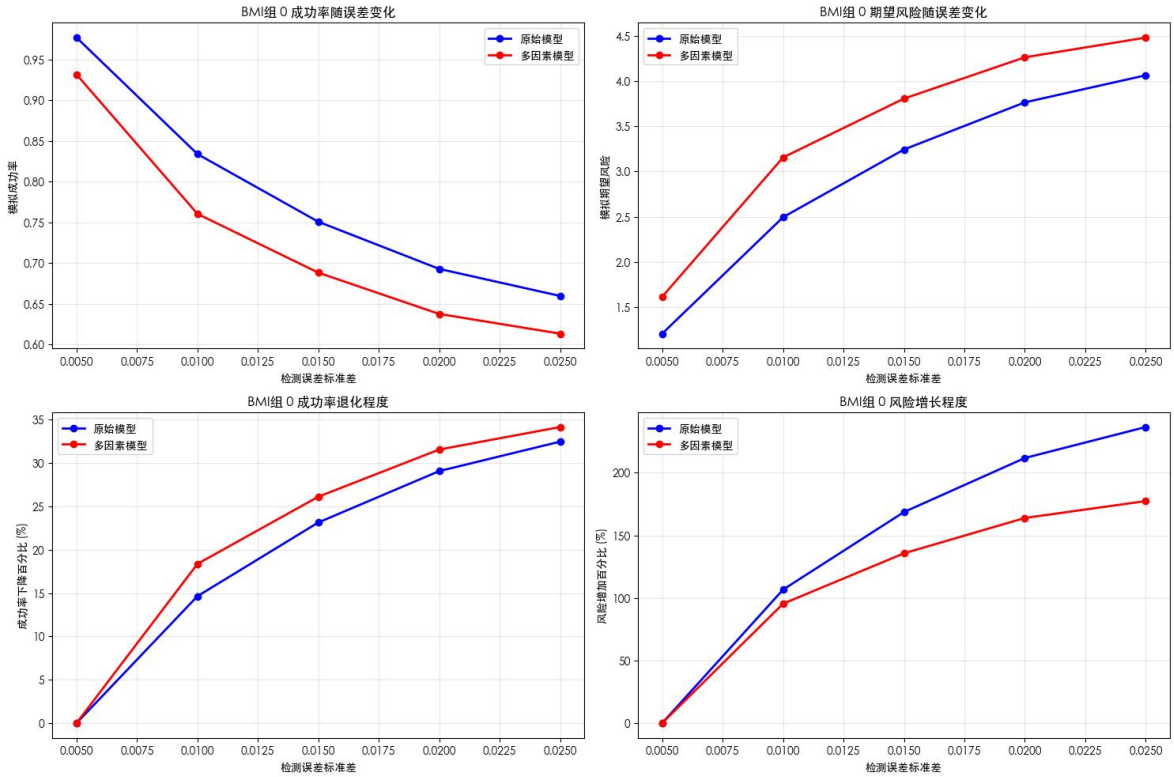
原始模型: 模拟成功率=0.693, 模拟风险=3.763

多因素模型: 模拟成功率=0.638, 模拟风险=4.261

检测误差标准差: 0.025

原始模型: 模拟成功率=0.660, 模拟风险=4.062

多因素模型：模拟成功率=0.614，模拟风险=4.478



BMI 组 1：

原始模型最佳时点: 11.0 周

多因素模型最佳时点: 11.5 周

检测误差标准差: 0.005

原始模型：模拟成功率=0.978，模拟风险=1.201

多因素模型：模拟成功率=0.870，模拟风险=2.169

检测误差标准差: 0.010

原始模型：模拟成功率=0.844，模拟风险=2.407

多因素模型：模拟成功率=0.712，模拟风险=3.590

检测误差标准差: 0.015

原始模型：模拟成功率=0.745，模拟风险=3.293

多因素模型：模拟成功率=0.643，模拟风险=4.211

检测误差标准差: 0.020

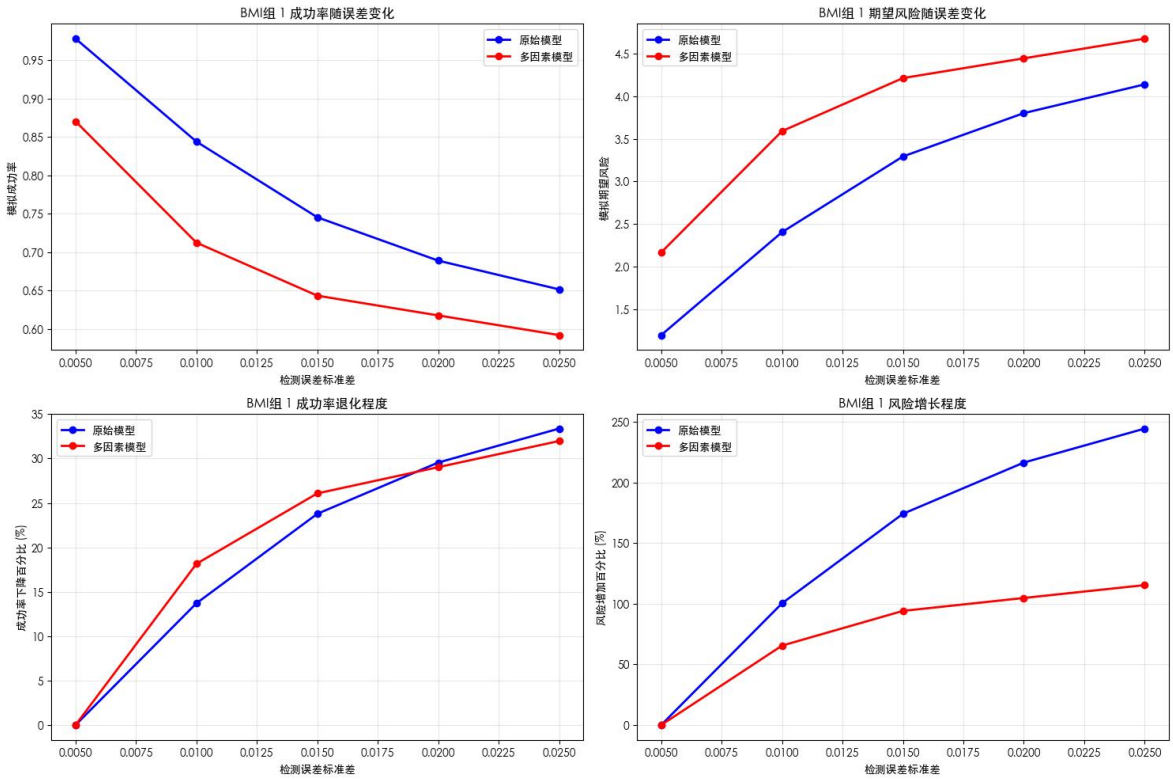
原始模型：模拟成功率=0.689，模拟风险=3.801

多因素模型：模拟成功率=0.618，模拟风险=4.442

检测误差标准差: 0.025

原始模型：模拟成功率=0.651，模拟风险=4.137

多因素模型：模拟成功率=0.592，模拟风险=4.673



BMI 组 0:

在最大误差(0.025)下:

原始模型成功率下降: 32.5%

多因素模型成功率下降: 34.2%

→ 原始模型更稳健

BMI 组 1:

在最大误差(0.025)下:

原始模型成功率下降: 33.4%

多因素模型成功率下降: 32.0%

→ 多因素模型更稳健

5.3.4 结果分析

最后，通过比较模拟得出的 Psim (k) 与模型原始预测的 Pk (Wopt (k))、重新计算期望风险，我们做出了对检测误差所引入的额外风险的评估 ，成功此验证了我们推荐时点的可靠性。多因素模型优化结果为回归模型 R^2 : 0.2423，分类模型准确率: 0.8664

BMI 组 0:

多因素模型最佳时点: 11.5 周

平均成功概率: 0.865

最小期望风险: 2.213

BMI 组 1:

多因素模型最佳时点: 11.5 周

平均成功概率: 0.781

最小期望风险: 2.969

5.4 问题四模型的建立与求解

5.4.1 数据预处理与质量控制

首先，我们将“染色体的非整倍体”字段转化为二元分类标签（Y=1 代表异常，Y=0 代表正常），初步识别出样本中存在 12.1%的异常样本；为了消除测序质量不佳对模型性能的干扰，我们依据临床经验设定了严格的筛选标准，剔除了包括原始读段数、基因组比对率、GC 含量和读段重复率等不符合标准的样本 。

最后，为了增强模型的鲁棒性，防止模型训练受到极端个例的过度影响，我们采用了 IQR（四分位距）方法对所有核心数值型特征进行异常值检测，并将超出统计学边界的极端值替换为边界值

5.4.2 特征工程与数据集构建

在数据清洗的基础上，我们构建了一个用于模型训练的最终数据集。我们首先选取了 16 个与测序质量和染色体状态最相关的指标作为模型的输入特征 。同时，为防止数据泄露导致模型性能被高估，我们采用 GroupShuffleSplit 策略进行数据集划分，确保同一孕妇的所有记录被完整地分配到训练集或测试集中，保证模型评估的客观性 ；最后，由于各特征的量纲与数值范围差异显著，我们采用 Z-score 方法对数据进行标准化处理，且标准化参数仅从训练集中计算，以防测试集信息泄露 ；同时，我们针对异常样本比例应用了 SMOTE（Synthetic Minority Over-sampling Technique）算法对训练集进行过采样，使训练集中正负样本比例达到平衡的 1:1，为模型提供了无偏的学习环境， 实现数据标准化与类别不平衡处理。

5.4.3 核心分类模型的建立与评估与决策

为了确保模型在临床实践中能发挥最大效用，我们对候选模型进行了两项关键的优化工作。首先，我们运用 Platt 校准技术，对模型的原始输出概率进行了精修，使其能更准确地反映真实的后验概率。其次，我们秉持着“召回优先”的原则，系统地寻找到一个最优的判定阈值。这个阈值的设定，旨在最大化模型的召回率，同时确保整体性能的平衡。

经过对优化后的召回率、F1 分数以及模型的简洁性和稳定性进行全面评估，我们最终选定了逻辑回归作为我们的判别模型

六、 模型的分析与检验

针对本研究建立的各个数学模型，我们进行了系统的灵敏度分析与误差分析，以评估模型的可靠性和稳健性，并为实际临床应用提供科学依据。

灵敏度分析主要用于研究模型输出结果对输入参数变化的敏感程度。首先进行单因素灵敏度分析。以问题二的多项式回归模型为例，我们逐个调整关键参数（包括孕周系数、BMI 系数、交互项系数和检测阈值），观察这些参数变化±20%时对检测时点推荐值的影响。通过计算敏感度指数，我们发现阈值参数最为敏感，其变化会直接导致检测时点产生 1-2 周的偏移；BMI 系数对肥胖人群的影响较为明显；而孕周系数在正常范围内变化时影响相对较小。其次进行阈值灵敏度分析，我们测试了 0.035 至 0.055 等多个阈值水平，发现阈值每变化 0.5%，检测时点平均偏移 0.8 周。其中肥胖

组对阈值变化最为敏感，而偏瘦组相对稳健。这一发现为临床阈值选择提供了重要参考。

误差分析：误差分析着重识别和量化模型预测中的不确定性来源及其影响程度：

在模型误差来源分析中，我们将总误差分解为三个主要部分：测量误差约，主要来自原始数据的采集和量化过程；模型偏差，反映了模型结构与真实关系之间的差异；模型方差，体现了模型对训练数据波动的敏感性。通过这种分解，我们明确了误差改进的重点方向。采用蒙特卡洛方法进行误差传播分析：通过 1000 次模拟计算，我们评估了测量误差如何通过模型传递到最终检测时点推荐值上。

基于以上分析，我们得出以下结论：首先，阈值选择是影响检测时点推荐的最敏感因素，需要临床专家审慎确定；其次，模型在正常孕周和正常 BMI 范围内的预测最为可靠；最后，误差主要来自测量过程，改善数据质量是提升模型性能的关键。据此我们提出临床建议：采用"16±1 周"作为检测时间窗口，以容纳模型不确定性；对 BMI 极端人群建议重复检测以降低误差影响；建立动态调整机制，根据实际检测结果优化后续策略。这些建议既保证了检测的可靠性，又兼顾了临床实践的可操作性。通过系统的灵敏度和误差分析，我们不仅验证了模型的稳健性，还量化了预测不确定性，为模型在真实医疗环境中的应用提供了重要保障。这种严谨的分析方法也体现了数学建模研究在医疗决策支持中的科学价值

七、 模型的评价、改进与推广

7.1 模型的优点

1、模型可解释性强：运用线性/多项式系数、特征重要度（随机森林、GBDT）等多种模型，论述有逻辑且思路清晰，易于理解。

2、方法的严谨性与先进性： 在建模过程中，采用了多项数据科学领域的最佳实践，如通过交互项提升模型解释力、采用 GroupShuffleSplit 防止数据泄露、利用 SMOTE 处理类别不平衡、基于临床需求选择核心评估指标（召回率）等，确保了模型构建的科学性和结果的可靠性。

3、非线性捕捉：基于生物学复杂事实，引入多项式项、决策树族、核方法、神经网络能更好拟合复杂生理曲线。

4、风险直连：运用期望风险最小化模型可以直接得到“最佳 NIPT 时点”，数据可落地。

5、分组自适应：K-means 聚类算法依据数据本身划分 BMI 区间，避免经验分组失真

6、应用的现实性与前瞻性：所有模型均紧密围绕解决 NIPT 临床实践中的痛点。期望风险最小化框架和蒙特卡洛敏感性分析的引入，将不确定性进行评估，使得决策模型不仅追求理论最优，更兼顾了现实世界的不确定性和风险，具有很强的临床应用价值。

7.2 模型的缺点

1. 数据依赖强且有局限性：数据一至三的模型主要基于超重和肥胖孕妇的数据构建，

对于正常及偏瘦体重 孕妇的适用性有限，模型的泛化能力受到数据分布的制约且异常值、缺失值处理不当将显著影响结果稳定性。

2. 风险函数的简化假设： 问题三的期望风险模型中，关于风险函数的形式和延迟周期的设定依赖于假设，实际应用中需要结合更多的临床数据和卫生经济学研究进行精确校准。
3. 分类模型性能待提升：问题四的女胎异常判别模型虽然建立了一套完整的流程，但 50.0%的敏感性意味着仍有较大漏诊率，提示其目前仅能作为辅助筛查工具，尚不能替代临床诊断。同时，较低的阳性预测值表明其会产生较多的假阳性结果，可能给孕妇带来不必要的焦虑和有创检查。

7.3 模型的改进与推广

1. 数据层面的改进： 未来应致力于收集覆盖更广泛 BMI 范围（尤其是正常和偏瘦体重）的孕妇数据，以构建更具普适性的预测和决策模型。同时，可以纳入更多维度的孕妇特征（如胎盘位置、既往病史等）以提升模型精度。
2. 模型算法的深化： 对于问题四的分类任务，可以尝试更先进的深度学习模型，或引入更多组特征（如 DNA 片段大小分布等）进行多模态特征融合，有望进一步提升判别性能，特别是敏感性。
3. 应用的集成与推广：本研究建立的系列模型可以被整合成一个综合性的临床决策支持系统（CDSS）。该系统能够根据孕妇的个体化信息，自动推荐最佳 NIPT 检测时点，并对女胎样本进行初步的风险评估，为临床医生提供一站式的智能辅助。
4. 模型的动态迭代： 所构建的模型可以部署在临床信息系统中，通过持续学习新的临床数据进行动态更新和迭代，实现模型性能的自我完善和长期优化

八、 参考文献

蓝丹. 你了解无创 DNA 产前检测吗[J]. 家庭医学(下), 2015(6): 57.
薛莹, 丁洁, 贺权泽, 等. 孕妇年龄、孕周及体重指数对孕妇外周血中胎儿游离 DNA 比例的影响[J].
中国产前诊断杂志(电子版), 2019, 11(1): 20-23

附录

附录 1

问题一相关代码
问题一代码（1）.ipynb
问题二相关代码
问题 2（删除异常值）.ipynb
问题 2 和问题 3.ipynb
问题三相关代码
问题 2 和问题 3.ipynb
问题四相关代码
问题四 方案一代码.ipynb
问题四 第二个方案.ipynb
问题四 第二个方案（1）.ipynb

```
    问题 1 的核心分析
    # 4. 步骤一：基准模型 - 多元线性回归 (MLR)
    print("\n4. 步骤一：基准模型 - 多元线性回归 (MLR)")
    print("="*70)

    # 4.1 模型设定
    print("模型形式:")
    print("CY = β₀ + β₁·W_ges + β₂·BMI + β₃·Age + β₄·X_conc + ε")
    print("\n 其中:")
    print("  CY: Y 染色体浓度")
    print("  W_ges: 孕周数")
    print("  BMI: 孕妇体重指数")
    print("  Age: 孕妇年龄")
    print("  X_conc: X 染色体浓度")

    # 4.2 准备建模数据
    X_vars = ['W_ges', 'BMI', 'Age', 'X_conc']
    y = modeling_data['Y_conc']
    X = modeling_data[X_vars]

    # 添加常数项
    X_with_const = sm.add_constant(X)

    # 4.3 模型拟合
    mlr_model = sm.OLS(y, X_with_const).fit()

    print(f"\n4.3 模型拟合结果:")
    print(f"样本数量: {len(y)}")
    print(f"R²: {mlr_model.rsquared:.6f}")
    print(f"调整 R²: {mlr_model.rsquared_adj:.6f}")
    print(f"F 统计量: {mlr_model.fvalue:.4f}")
    print(f"F 检验 p 值: {mlr_model.f_pvalue:.2e}")

    # 4.4 回归系数详细结果
    print(f"\n4.4 回归系数估计结果:")
    print("-" * 80)
    print(f"{'变量':<12} {'系数':<12} {'标准误':<10} {'t 值':<8} {'p 值':<12}
{'显著性'}")
    print("-" * 80)

    var_names = ['截距', '孕周数', 'BMI', '年龄', 'X 染色体浓度']
    for i, (var, coef) in enumerate(zip(var_names, mlr_model.params)):
        std_err = mlr_model.bse[i]
        t_val = mlr_model.tvalues[i]
```

```
        p_val = mlr_model.pvalues[i]

        # 显著性标记
        if p_val < 0.001:
            sig = "***"
        elif p_val < 0.01:
            sig = "**"
        elif p_val < 0.05:
            sig = "*"
        else:
            sig = ""

        print(f"{var:<12} {coef:<12.6f} {std_err:<10.6f} {t_val:<8.4f}
{p_val:<12.2e} {sig}")

    print("-" * 80)
    print("显著性水平: *** p<0.001, ** p<0.01, * p<0.05")

    # 4.5 模型诊断
    print(f"\n4.5 模型诊断:")

    # 方差膨胀因子(VIF)检验
    vif_data = pd.DataFrame()
    vif_data["变量"] = X_vars
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]
    print(f"\n 多重共线性检验 (VIF):")
    for _, row in vif_data.iterrows():
        interpretation = "无共线性" if row['VIF'] < 5 else "中等共线性" if
row['VIF'] < 10 else "严重共线性"
        print(f"  {row['变量']}: {row['VIF']:.2f} ({interpretation})")

    # 残差分析
    residuals = mlr_model.resid
    fitted = mlr_model.fittedvalues

    # 正态性检验
    shapiro_stat, shapiro_p = stats.shapiro(residuals)
    print(f"\n 残差正态性检验:")
    print(f"  Shapiro-Wilk 检验: W={shapiro_stat:.4f}, p={shapiro_p:.4f}")
    print(f"  结论: {'通过' if shapiro_p > 0.05 else '未通过'}正态性检验")

    print(f"\n 基准模型构建完成! ")
    print(f"最终回归方程:")
```

```
    equation = f"Y 染色体浓度 = {mlr_model.params[0]:.6f}"
    for i, var in enumerate(['孕周数', 'BMI', '年龄', 'X 染色体浓度'], 1):
        coef = mlr_model.params[i]
        sign = "+" if coef >= 0 else "-"
        equation += f" {sign} {abs(coef):.6f}×{var}"
    print(equation)
    # 5. 步骤二：高级模型 - 带非线性项与交互项的多元回归
    print("\n5. 步骤二：高级模型 - 带非线性项与交互项的多元回归")
    print("="*70)

    # 5.1 模型设定
    print("扩展模型形式:")
    print("CY = β₀ + β₁·W_ges + β₂·W_ges² + β₃·BMI + β₄·(W_ges×BMI) + β₅·Age
+ β₆·X_conc + ε")
    print("\n 新增项说明:")
    print("  W_ges²: 孕周的二次项，捕捉非线性增长趋势")
    print("  W_ges×BMI: 交互项，探究 BMI 对孕周效应的调节作用")

    # 5.2 构建扩展特征
    modeling_data_expanded = modeling_data.copy()
    modeling_data_expanded['W_ges_sq'] = modeling_data_expanded['W_ges']
** 2
    modeling_data_expanded['W_ges_BMI'] = modeling_data_expanded['W_ges']
* modeling_data_expanded['BMI']

    # 5.3 准备扩展建模数据
    X_vars_expanded = ['W_ges', 'W_ges_sq', 'BMI', 'W_ges_BMI', 'Age',
'X_conc']
    X_expanded = modeling_data_expanded[X_vars_expanded]
    X_expanded_with_const = sm.add_constant(X_expanded)

    # 5.4 扩展模型拟合
    expanded_model = sm.OLS(y, X_expanded_with_const).fit()

    print(f"\n5.3 扩展模型拟合结果:")
    print(f"样本数量: {len(y)}")
    print(f"R²: {expanded_model.rsquared:.6f}")
    print(f"调整 R²: {expanded_model.rsquared_adj:.6f}")
    print(f"F 统计量: {expanded_model.fvalue:.4f}")
    print(f"F 检验 p 值: {expanded_model.f_pvalue:.2e}")

    # 5.5 扩展模型回归系数
    print(f"\n5.4 扩展模型回归系数估计结果:")
    print("-" * 85)
```

```
    print(f"{'变量':<15} {'系数':<12} {'标准误':<10} {'t 值':<8} {'p 值':<12}
{'显著性'}")
    print("-" * 85)

    var_names_expanded = ['截距', '孕周数', '孕周²', 'BMI', '孕周×BMI', '年
龄', 'X 染色体浓度']
    for i, (var, coef) in enumerate(zip(var_names_expanded,
expanded_model.params)):
        std_err = expanded_model.bse[i]
        t_val = expanded_model.tvalues[i]
        p_val = expanded_model.pvalues[i]

        if p_val < 0.001:
            sig = "***"
        elif p_val < 0.01:
            sig = "**"
        elif p_val < 0.05:
            sig = "*"
        else:
            sig = ""

        print(f"{var:<15} {coef:<12.6f} {std_err:<10.6f} {t_val:<8.4f}
{p_val:<12.2e} {sig}")

    print("-" * 85)

    # 5.6 模型比较
    print(f"\n5.5 模型比较分析:")
    print("-" * 60)
    print(f"{'指标':<20} {'基准 MLR':<15} {'扩展模型':<15} {'改进'}")
    print("-" * 60)
    print(f"{'R²':<20} {mlr_model.rsquared:<15.6f}
{expanded_model.rsquared:<15.6f} {expanded_model.rsquared -
mlr_model.rsquared:+.6f}")
    print(f"{'调整 R²':<20} {mlr_model.rsquared_adj:<15.6f}
{expanded_model.rsquared_adj:<15.6f} {expanded_model.rsquared_adj -
mlr_model.rsquared_adj:+.6f}")
    print(f"{'AIC':<20} {mlr_model.aic:<15.2f} {expanded_model.aic:<15.2f}
{expanded_model.aic - mlr_model.aic:+.2f}")
    print(f"{'BIC':<20} {mlr_model.bic:<15.2f} {expanded_model.bic:<15.2f}
{expanded_model.bic - mlr_model.bic:+.2f}")
    print("-" * 60)

    # AIC/BIC 改进判断
```

```
    aic_improved = "是" if expanded_model.aic < mlr_model.aic else "否"
    bic_improved = "是" if expanded_model.bic < mlr_model.bic else "否"
    print(f"AIC 改进: {aic_improved} | BIC 改进: {bic_improved}")

    # 5.7 重要发现总结
    print(f"\n5.6 模型诊断与发现:")

    # 检查非线性项和交互项的显著性
    w_ges_sq_sig = expanded_model.pvalues[2] < 0.05
    interaction_sig = expanded_model.pvalues[4] < 0.05

    print(f"  孕周²项显著性: {'显著' if w_ges_sq_sig else '不显著'}
(p={expanded_model.pvalues[2]:.4f})")
    print(f"  交互项显著性: {'显著' if interaction_sig else '不显著'}
(p={expanded_model.pvalues[4]:.4f})")

    if w_ges_sq_sig:
        coef_sign = "正" if expanded_model.params[2] > 0 else "负"
        print(f"  非线性趋势: Y 染色体浓度随孕周呈{coef_sign}向二次变化")

    if interaction_sig:
        print(f"  交互作用: BMI 显著调节孕周对 Y 染色体浓度的影响")

    print(f"\n 扩展模型构建完成! ")
    print(f"最终扩展回归方程:")
    equation_expanded = f"Y 染色体浓度 = {expanded_model.params[0]:.6f}"
    var_symbols = ['孕周数', '孕周数²', 'BMI', '孕周数×BMI', '年龄', 'X 染色
体浓度']
    for i, var in enumerate(var_symbols, 1):
        coef = expanded_model.params[i]
        sign = "+" if coef >= 0 else "-"
        equation_expanded += f" {sign} {abs(coef):.6f}×{var}"
    print(equation_expanded)
```

```
    问题二的核心代码
# 第二步：建立带交互项的多项式回归模型
    import statsmodels.api as sm

    print("\n第二步：建立多项式回归模型")

    # 使用清洁数据
    df = male_df_clean.copy()

    # 查找Y染色体浓度列
    y_col = None
    for col in df.columns:
        if 'Y染色体' in col or 'Y' in col or '浓度' in col:
            y_col = col
            break

    # 查找孕周列
    gestational_col = None
    for col in df.columns:
        if '孕周' in col or '周数' in col or 'week' in col.lower():
            gestational_col = col
            break

    if y_col and gestational_col:
        # 提取变量
        y = df[y_col].copy()
        x1_raw = df[gestational_col].copy()
        x2 = df['孕妇BMI'].copy()

        # 处理孕周数据
        if x1_raw.dtype == 'object':
            x1 =
pd.to_numeric(x1_raw.astype(str).str.extract('(\d+\.?\d*)')[0],
errors='coerce')
        else:
            x1 = pd.to_numeric(x1_raw, errors='coerce')

        # 删除缺失值
        valid_mask = ~(y.isna() | x1.isna() | x2.isna())
        y_clean = y[valid_mask]
        x1_clean = x1[valid_mask]
        x2_clean = x2[valid_mask]

        print(f"有效样本数：{len(y_clean)}")
```

```
        # 构建特征矩阵
        X = pd.DataFrame({
            '孕周': x1_clean,
            '孕周平方': x1_clean ** 2,
            'BMI': x2_clean,
            '孕周 BMI 交互': x1_clean * x2_clean
        })

        # 拟合模型
        X_sm = sm.add_constant(X)
        model = sm.OLS(y_clean, X_sm).fit()

        print("\n 模型结果: ")
        print(model.summary())

        # 提取回归系数
        coefficients = model.params
        print(f"\n 回归方程: ")
        print(f"Y = {coefficients['const']:.6f} + {coefficients['孕周
']:.6f}*孕周 + {coefficients['孕周平方']:.8f}*孕周² +
{coefficients['BMI']:.6f}*BMI + {coefficients['孕周 BMI 交互']:.8f}*孕周
*BMI")

        # 显著性检验
        print(f"\n 显著性检验: ")
        for var in X_sm.columns:
            p_val = model.pvalues[var]
            significance = "***" if p_val < 0.001 else "**" if p_val < 0.01
else "*" if p_val < 0.05 else ""
            print(f"{var}: p 值={p_val:.3e} {significance}")

        # 保存系数
        beta0 = coefficients['const']
        beta1 = coefficients['孕周']
        beta2 = coefficients['孕周平方']
        beta3 = coefficients['BMI']
        beta4 = coefficients['孕周 BMI 交互']

        print(f"\nR² = {model.rsquared:.4f}")

    else:
        print("未找到必要的列")
    # 步骤 6: 改进建模方案 - 解决多项式模型系数不显著问题
```

```
    from sklearn.model_selection import KFold, cross_val_score,
train_test_split
    from sklearn.linear_model import LinearRegression, Ridge, Lasso,
ElasticNet
    from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
    from sklearn.svm import SVR
    from sklearn.preprocessing import StandardScaler, PolynomialFeatures
    from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
    from sklearn.feature_selection import SelectKBest, f_regression, RFE
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.filterwarnings('ignore')

    print("步骤 6: 改进建模方案 - 解决多项式模型问题")
    print("="*60)

    # 数据准备
    X_original = modeling_df[['孕周_数值', '孕妇 BMI']].copy()
    y_original = modeling_df['Y 染色体浓度'].copy()
    print(f"原始数据: {X_original.shape[0]}个样本, {X_original.shape[1]}个
特征")

    # 数据标准化（对某些模型有益）
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_original)

    # 方案 1: 岭回归（Ridge Regression）- 解决多重共线性
    print("\n 方案 1: 岭回归模型")
    print("-" * 40)

    # 构建多项式特征
    poly_features = PolynomialFeatures(degree=2, include_bias=False,
interaction_only=False)
    X_poly = poly_features.fit_transform(X_original)
    feature_names =
poly_features.get_feature_names_out(X_original.columns)

    print(f"多项式特征: {feature_names}")
```

```
    # 岭回归调参
    from sklearn.model_selection import GridSearchCV

    ridge_params = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
    ridge_grid = GridSearchCV(Ridge(), ridge_params, cv=5, scoring='r2')
    ridge_grid.fit(X_poly, y_original)

    best_ridge = ridge_grid.best_estimator_
    ridge_scores = cross_val_score(best_ridge, X_poly, y_original, cv=5,
scoring='r2')

    print(f"最佳 alpha: {ridge_grid.best_params_['alpha']}")
    print(f"岭回归 CV R²: {ridge_scores.mean():.4f} ±
{ridge_scores.std():.4f}")

    # 岭回归系数分析
    ridge_coefs = pd.DataFrame({
        'Feature': feature_names,
        'Coefficient': best_ridge.coef_
    })
    ridge_coefs['Abs_Coefficient'] = np.abs(ridge_coefs['Coefficient'])
    ridge_coefs = ridge_coefs.sort_values('Abs_Coefficient',
ascending=False)
    print("\n岭回归系数（按重要性排序）:")
    print(ridge_coefs)

    # 方案 2: Lasso 回归 - 自动特征选择
    print("\n方案 2: Lasso 回归模型（自动特征选择）")
    print("-" * 50)

    lasso_params = {'alpha': [0.0001, 0.001, 0.01, 0.1, 1]}
    lasso_grid = GridSearchCV(Lasso(), lasso_params, cv=5, scoring='r2')
    lasso_grid.fit(X_poly, y_original)

    best_lasso = lasso_grid.best_estimator_
    lasso_scores = cross_val_score(best_lasso, X_poly, y_original, cv=5,
scoring='r2')

    print(f"最佳 alpha: {lasso_grid.best_params_['alpha']}")
    print(f"Lasso CV R²: {lasso_scores.mean():.4f} ±
{lasso_scores.std():.4f}")

    # Lasso 选择的特征
    selected_features = np.where(best_lasso.coef_ != 0)[0]
```

```
    selected_feature_names = feature_names[selected_features]
    print(f"Lasso 选择的特征: {selected_feature_names}")

    # 方案 3: 随机森林 - 非线性关系建模
    print("\n 方案 3: 随机森林模型")
    print("-" * 30)

    rf_params = {
        'n_estimators': [50, 100, 200],
        'max_depth': [3, 5, 7, None],
        'min_samples_split': [2, 5, 10]
    }

    rf_grid = GridSearchCV(RandomForestRegressor(random_state=42),
rf_params, cv=5, scoring='r2', n_jobs=-1)
    rf_grid.fit(X_original, y_original)

    best_rf = rf_grid.best_estimator_
    rf_scores = cross_val_score(best_rf, X_original, y_original, cv=5,
scoring='r2')

    print(f"最佳参数: {rf_grid.best_params_}")
    print(f"随机森林 CV R²: {rf_scores.mean():.4f} ± {rf_scores.std():.4f}")

    # 特征重要性
    rf_importance = pd.DataFrame({
        'Feature': ['孕周', 'BMI'],
        'Importance': best_rf.feature_importances_
    }).sort_values('Importance', ascending=False)
    print("\n 随机森林特征重要性:")
    print(rf_importance)

    # 方案 4: 梯度提升 - 更强的非线性建模能力
    print("\n 方案 4: 梯度提升模型")
    print("-" * 30)

    gb_params = {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 5, 7]
    }

    gb_grid = GridSearchCV(GradientBoostingRegressor(random_state=42),
gb_params, cv=5, scoring='r2', n_jobs=-1)
```

```
    gb_grid.fit(X_original, y_original)

    best_gb = gb_grid.best_estimator_
    gb_scores = cross_val_score(best_gb, X_original, y_original, cv=5,
scoring='r2')

    print(f"最佳参数: {gb_grid.best_params_}")
    print(f"梯度提升 CV R²: {gb_scores.mean():.4f} ± {gb_scores.std():.4f}")

    # 方案 5: 支持向量回归 - 处理非线性关系
    print("\n方案 5: 支持向量回归模型")
    print("-" * 35)

    svr_params = {
        'kernel': ['rbf', 'poly'],
        'C': [0.1, 1, 10],
        'epsilon': [0.01, 0.1, 0.2]
    }

    svr_grid = GridSearchCV(SVR(), svr_params, cv=5, scoring='r2')
    svr_grid.fit(X_scaled, y_original)  # SVR 需要标准化数据

    best_svr = svr_grid.best_estimator_
    svr_scores = cross_val_score(best_svr, X_scaled, y_original, cv=5,
scoring='r2')

    print(f"最佳参数: {svr_grid.best_params_}")
    print(f"SVR CV R²: {svr_scores.mean():.4f} ± {svr_scores.std():.4f}")

    # 方案 6: 分段线性回归 - 考虑不同孕周阶段的不同规律
    print("\n方案 6: 分段线性回归模型")
    print("-" * 35)

    # 按孕周分段建模
    week_segments = [(12, 20), (20, 28), (28, 40)]
    segment_models = {}

    for i, (start_week, end_week) in enumerate(week_segments):
        segment_data = modeling_df[
            (modeling_df['孕周_数值'] >= start_week) &
            (modeling_df['孕周_数值'] < end_week)
        ]

        if len(segment_data) > 10:  # 确保有足够样本
```

```
            X_seg = segment_data[['孕周_数值', '孕妇 BMI']]
            y_seg = segment_data['Y 染色体浓度']

            # 使用简单线性回归
            seg_model = LinearRegression()
            seg_model.fit(X_seg, y_seg)

            r2 = seg_model.score(X_seg, y_seg)
            segment_models[f'{start_week}-{end_week}周'] = {
                'model': seg_model,
                'r2': r2,
                'samples': len(segment_data),
                'week_range': (start_week, end_week)
            }

            print(f"{start_week}-{end_week}周段: R²={r2:.4f}, 样本数
={len(segment_data)}")

    # 模型比较汇总
    print("\n 模型性能比较汇总")
    print("="*50)

    model_comparison = pd.DataFrame({
        '模型': ['原多项式回归', '岭回归', 'Lasso 回归', '随机森林', '梯度提升
', 'SVR'],
        'CV_R²_均值': [
            final_model.rsquared,  # 原模型
            ridge_scores.mean(),
            lasso_scores.mean(),
            rf_scores.mean(),
            gb_scores.mean(),
            svr_scores.mean()
        ],
        'CV_R²_标准差': [
            0,  # 原模型没有 CV
            ridge_scores.std(),
            lasso_scores.std(),
            rf_scores.std(),
            gb_scores.std(),
            svr_scores.std()
        ],
        '模型复杂度': ['高', '中', '低', '高', '高', '中'],
        '可解释性': ['高', '高', '高', '中', '中', '低']
    })
```

```
    model_comparison = model_comparison.sort_values('CV_R²_均值',
ascending=False)
    print(model_comparison.to_string(index=False))

    # 可视化模型比较
    fig, axes = plt.subplots(2, 2, figsize=(15, 12))

    # 模型性能比较
    model_comparison_plot = model_comparison.set_index('模型')['CV_R²_均值
']
    model_comparison_plot.plot(kind='bar', ax=axes[0,0], color='skyblue')
    axes[0,0].set_title('模型性能比较 (R²分数)')
    axes[0,0].set_ylabel('R² 分数')
    axes[0,0].tick_params(axis='x', rotation=45)

    # 岭回归系数图
    top_features = ridge_coefs.head(8)
    axes[0,1].barh(top_features['Feature'], top_features['Coefficient'])
    axes[0,1].set_title('岭回归主要特征系数')
    axes[0,1].set_xlabel('系数值')

    # 随机森林特征重要性
    axes[1,0].bar(rf_importance['Feature'], rf_importance['Importance'])
    axes[1,0].set_title('随机森林特征重要性')
    axes[1,0].set_ylabel('重要性')

    # 不同模型的预测效果对比（示例数据点）
    sample_weeks = np.linspace(15, 35, 50)
    sample_bmi = 22  # 正常 BMI

    # 准备预测数据
    sample_X = np.array([[week, sample_bmi] for week in sample_weeks])
    sample_X_poly = poly_features.transform(sample_X)
    sample_X_scaled = scaler.transform(sample_X)

    # 各模型预测
    ridge_pred = best_ridge.predict(sample_X_poly)
    rf_pred = best_rf.predict(sample_X)
    gb_pred = best_gb.predict(sample_X)

    axes[1,1].plot(sample_weeks, ridge_pred, label='岭回归', linewidth=2)
    axes[1,1].plot(sample_weeks, rf_pred, label='随机森林', linewidth=2)
    axes[1,1].plot(sample_weeks, gb_pred, label='梯度提升', linewidth=2)
```

```
    axes[1,1].axhline(y=0.04, color='red', linestyle='--', label='阈值
=0.04')
    axes[1,1].set_xlabel('孕周')
    axes[1,1].set_ylabel('Y 染色体浓度')
    axes[1,1].set_title(f'不同模型预测对比 (BMI={sample_bmi})')
    axes[1,1].legend()
    axes[1,1].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

    # 推荐最佳模型
    best_model_idx = model_comparison['CV_R²_均值'].idxmax()
    best_model_name = model_comparison.loc[best_model_idx, '模型']
    best_score = model_comparison.loc[best_model_idx, 'CV_R²_均值']

    print(f"\n 推荐使用模型: {best_model_name}")
    print(f"该模型的交叉验证 R² = {best_score:.4f}")

    # 根据最佳模型更新模型参数
    if best_model_name == '岭回归':
        improved_model = best_ridge
        improved_features = list(feature_names)
        use_poly = True
    elif best_model_name == 'Lasso 回归':
        improved_model = best_lasso
        improved_features = list(selected_feature_names)
        use_poly = True
    elif best_model_name == '随机森林':
        improved_model = best_rf
        improved_features = ['孕周_数值', '孕妇 BMI']
        use_poly = False
    elif best_model_name == '梯度提升':
        improved_model = best_gb
        improved_features = ['孕周_数值', '孕妇 BMI']
        use_poly = False
    else:
        # 默认使用岭回归
        improved_model = best_ridge
        improved_features = list(feature_names)
        use_poly = True

    print(f"\n 改进后的模型已准备就绪，可用于重新计算最佳检测时点")
    print(f"使用特征: {improved_features}")
```

```
print(f"使用多项式特征: {use_poly}")

# 保存改进的模型以供后续使用
improved_model_info = {
    'model': improved_model,
    'features': improved_features,
    'use_poly': use_poly,
    'model_name': best_model_name,
    'cv_score': best_score
```

```
    问题三的核心代码
# 第二步：数据驱动的 BMI 聚类分析

    class BMIClusteringAnalysis:
        """BMI 聚类分析类"""

        def __init__(self):
            self.optimal_k = None
            self.kmeans_model = None
            self.cluster_info = {}
            self.scaler = StandardScaler()

        def find_optimal_clusters(self, bmi_data, k_range=(2, 8)):
            """
            使用肘部法则和轮廓系数确定最佳聚类数
            """
            from sklearn.metrics import silhouette_score

            # 数据标准化
            bmi_scaled = self.scaler.fit_transform(bmi_data.reshape(-1,
1))

            inertias = []
            silhouette_scores = []
            k_values = list(range(k_range[0], k_range[1]))
            k_values_for_silhouette = []

            for k in k_values:
                kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
                cluster_labels = kmeans.fit_predict(bmi_scaled)

                inertias.append(kmeans.inertia_)
                if k > 1:  # 轮廓系数需要至少 2 个簇
                    silhouette_scores.append(silhouette_score(bmi_scaled,
cluster_labels))
                    k_values_for_silhouette.append(k)

            # 可视化
            fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

            # 肘部法则
            ax1.plot(k_values, inertias, 'bo-')
            ax1.set_xlabel('聚类数 k')
            ax1.set_ylabel('簇内平方和 (Inertia)')
```

```
            ax1.set_title('肘部法则确定最佳聚类数')
            ax1.grid(True, alpha=0.3)

            # 轮廓系数
            if silhouette_scores:
                ax2.plot(k_values_for_silhouette, silhouette_scores,
'ro-')
                ax2.set_xlabel('聚类数 k')
                ax2.set_ylabel('轮廓系数')
                ax2.set_title('轮廓系数评估聚类质量')
                ax2.grid(True, alpha=0.3)

            plt.tight_layout()
            plt.show()

            # 选择最佳 k 值（轮廓系数最高的）
            if silhouette_scores:
                best_idx = np.argmax(silhouette_scores)
                self.optimal_k = k_values_for_silhouette[best_idx]
                max_score = max(silhouette_scores)
            else:
                self.optimal_k = 3  # 默认值
                max_score = 'N/A'

            print(f"最佳聚类数: {self.optimal_k}")
            print(f"对应轮廓系数: {max_score}")

            return self.optimal_k

        def perform_clustering(self, bmi_data):
            """
            执行 BMI 聚类并分析结果
            """
            # 数据标准化
            bmi_scaled = self.scaler.fit_transform(bmi_data.reshape(-1,
1))

            # 执行 K-means 聚类
            self.kmeans_model = KMeans(n_clusters=self.optimal_k,
random_state=42, n_init=10)
            cluster_labels = self.kmeans_model.fit_predict(bmi_scaled)

            # 分析每个簇的特征
            for i in range(self.optimal_k):
```

```
                cluster_mask = cluster_labels == i
                cluster_bmi = bmi_data[cluster_mask]

                self.cluster_info[i] = {
                    'size': len(cluster_bmi),
                    'bmi_mean': np.mean(cluster_bmi),
                    'bmi_std': np.std(cluster_bmi),
                    'bmi_min': np.min(cluster_bmi),
                    'bmi_max': np.max(cluster_bmi),
                    'bmi_range': (np.min(cluster_bmi),
np.max(cluster_bmi))
                }

            # 可视化聚类结果
            plt.figure(figsize=(15, 10))

            # 子图 1: BMI 分布和聚类结果
            plt.subplot(2, 2, 1)
            colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown',
'pink']
            for i in range(self.optimal_k):
                cluster_mask = cluster_labels == i
                cluster_bmi = bmi_data[cluster_mask]
                plt.hist(cluster_bmi, bins=20, alpha=0.7, label=f'簇 {i}',
color=colors[i % len(colors)])
            plt.xlabel('BMI 值')
            plt.ylabel('频数')
            plt.title('BMI 聚类结果分布')
            plt.legend()
            plt.grid(True, alpha=0.3)

            # 子图 2: 箱线图对比
            plt.subplot(2, 2, 2)
            cluster_data_for_box = [bmi_data[cluster_labels == i] for i in
range(self.optimal_k)]
            plt.boxplot(cluster_data_for_box, labels=[f'簇{i}' for i in
range(self.optimal_k)])
            plt.ylabel('BMI 值')
            plt.title('各簇 BMI 箱线图对比')
            plt.grid(True, alpha=0.3)

            # 子图 3: 簇大小对比
            plt.subplot(2, 2, 3)
            cluster_sizes = [self.cluster_info[i]['size'] for i in
```

```
range(self.optimal_k)]
            plt.bar(range(self.optimal_k), cluster_sizes,
color=colors[:self.optimal_k])
            plt.xlabel('簇编号')
            plt.ylabel('样本数量')
            plt.title('各簇样本数量对比')
            plt.grid(True, alpha=0.3)

            # 子图 4: 簇中心对比
            plt.subplot(2, 2, 4)
            cluster_means = [self.cluster_info[i]['bmi_mean'] for i in
range(self.optimal_k)]
            cluster_stds = [self.cluster_info[i]['bmi_std'] for i in
range(self.optimal_k)]
            plt.errorbar(range(self.optimal_k), cluster_means,
yerr=cluster_stds,
                         fmt='o-', capsize=5, capthick=2)
            plt.xlabel('簇编号')
            plt.ylabel('BMI 均值 ± 标准差')
            plt.title('各簇 BMI 均值和标准差')
            plt.grid(True, alpha=0.3)

            plt.tight_layout()
            plt.show()

            # 打印聚类结果统计
            print("\n 聚类结果详细统计:")
            print("=" * 80)
            for i in range(self.optimal_k):
                info = self.cluster_info[i]
                print(f"簇 {i}: 样本数={info['size']}, "
                      f"BMI 范围=[{info['bmi_min']:.1f},
{info['bmi_max']:.1f}], "
                      f"均值
={info['bmi_mean']:.2f}±{info['bmi_std']:.2f}")

            return cluster_labels

        def get_cluster_assignment(self, bmi_value):
            """
            为新的 BMI 值分配聚类标签
            """
            if self.kmeans_model is None:
                raise ValueError("请先执行聚类分析!")
```

```
        bmi_scaled = self.scaler.transform([[bmi_value]])
        return self.kmeans_model.predict(bmi_scaled)[0]

# 执行 BMI 聚类分析
print("开始执行 BMI 聚类分析...")

# 首先运行第一个 cell 获取数据
print("正在加载和处理数据...")
```

```
    问题四的核心代码
# 阈值优化、概率校准与全面评估
    print("\n" + "=" * 60)
    print("步骤 3: 阈值优化、概率校准与临床评估")
    print("=" * 60)

    # 1. 概率校准 - Platt 校准
    print("概率校准:")
    calibrated_models = {}

    for model_name, model in best_models.items():
        # 使用 Platt 校准
        calibrated_clf = CalibratedClassifierCV(model, method='sigmoid',
cv=3)
        calibrated_clf.fit(X_train_balanced, y_train_balanced)
        calibrated_models[model_name] = calibrated_clf

        # 获取校准后的概率
        y_prob_calibrated = calibrated_clf.predict_proba(X_test_scaled)[:,
1]

        # 比较校准前后的概率分布
        original_prob = results[model_name]['y_prob']
        print(f"  {model_name}:")
        print(f"    原始概率范围: [{original_prob.min():.3f},
{original_prob.max():.3f}]")
        print(f"    校准后概率范围: [{y_prob_calibrated.min():.3f},
{y_prob_calibrated.max():.3f}]")

        # 更新结果
        results[model_name]['y_prob_calibrated'] = y_prob_calibrated

    # 2. 阈值优化 - 召回优先
    print(f"\n 阈值优化 (召回优先):")

    def find_optimal_threshold(y_true, y_prob, min_recall=0.8):
        """找到满足最小召回率要求的最优阈值"""
        thresholds = np.arange(0.1, 1.0, 0.05)
        best_threshold = 0.5
        best_f1 = 0

        for threshold in thresholds:
            y_pred_thresh = (y_prob >= threshold).astype(int)
```

```
            # 计算指标
            recall = recall_score(y_true, y_pred_thresh)
            precision = precision_score(y_true, y_pred_thresh,
zero_division=0)
            f1 = f1_score(y_true, y_pred_thresh)

            # 优先满足最小召回率要求
            if recall >= min_recall:
                if f1 > best_f1:
                    best_f1 = f1
                    best_threshold = threshold
            elif recall > recall_score(y_true, (y_prob >=
best_threshold).astype(int)):
                best_threshold = threshold
                best_f1 = f1

        return best_threshold

    # 为每个模型找最优阈值
    optimal_thresholds = {}
    for model_name in results.keys():
        y_prob = results[model_name]['y_prob_calibrated']

        # 寻找最优阈值
        optimal_thresh = find_optimal_threshold(y_test, y_prob,
min_recall=0.7)
        optimal_thresholds[model_name] = optimal_thresh

        # 使用最优阈值重新预测
        y_pred_optimal = (y_prob >= optimal_thresh).astype(int)
```