基于回归算法的睡眠效率分析系统说明书

课程名称：Python 数据分析项目

组　　名：阿尔法小组

组　　员：陈介然、杨瀚森、赵翔、
解喆、牛雅曼、王佳幸

编写日期：2024.06.12

分工表						
	技术选型	需求分析	系统设计	后端代码	前端代码	文档
陈介然	√			√	√	√
杨瀚森				√		√
牛雅曼	√					√
赵　翔				√	√	
解　喆			√			
王佳幸		√				

目录

1、技术选型

1.1 开发环境

PyCharm 是由 JetBrains 公司打造的一款综合性 Python 开发工具，为 Python 程序员提供了一个全方位的集成开发环境。这个环境包括高级代码编辑功能，如智能提示和自动补全，以及代码格式化和重构工具，这些都有助于提升代码的整洁度和可维护性。此外，PyCharm 还提供了强大的调试工具，使得开发者能够轻松地追踪和修复代码中的错误。

在项目管理方面，PyCharm 支持 Git、SVN 等多种版本控制系统的集成，使得代码的版本管理变得更加简单。它还提供虚拟的环境管理，允许开发者在一个隔离的环境中测试和部署他们的应用程序。远程开发功能则让开发者可以在远程服务器上工作，而自动化测试工具则帮助确保代码的质量和稳定性。

PyCharm 对各种 Python 框架，如 Django、Flask、Pyramid 等提供了广泛的支持，并且集成了 Jupyter Notebook，这是一个广受欢迎的交互式计算环境，特别适合数据科学和机器学习项目。通过这些集成，PyCharm 为 Python 开发者提供了一个功能丰富且高效的开发平台。

1.1.1 sklearn 介绍

sklearn 是 Python 中的一个机器学习库，它包含了许多常见的机器学习算法和工具， 如分类、回归、聚类、降维、模型选择、数据预处理等，以及用于模型评估和调优的工具。sklearn 库提供了简单、一致的 API 接口，方便用户快速构建和调用机器学习模型，同时也具有可扩展性和灵活 性，可以与其他 Python 科学计算库，如 NumPy、Pandas、Matplotlib 等进行整合。

1.1.2 pytorch 介绍

PyTorch 是一个 Python 深度学习框架，它将数据封装成张量（Tensor）来进行运算。PyTorch 中的张量就是元素为同一种数据类型

的多维矩阵。在 PyTorch 中，张量以 "类" 的形式封装起来，对张量的一些运算、处理的方法被封装在类中。

1.2 机器学习相关模型理论

1.2.1 回归分析

回归分析是一种统计学上分析数据的方法，目的在于了解两个或多个变量间是否相关、相关方向与强度，并建立数学模型以便观察特定变量来预测研究者感兴趣的变量。更具体的来说，回归分析可以帮助人们了解在只有一个自变量变化时因变量的变化量。一般来说，通过回归分析我们可以由给出的自变量估计因变量的条件期望。

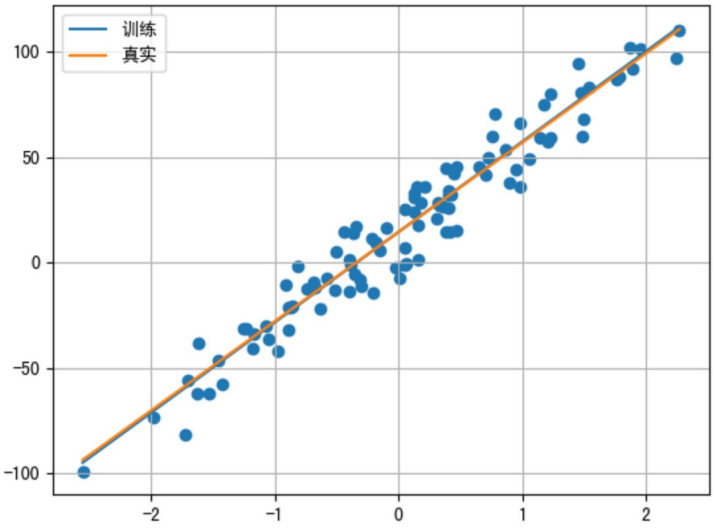

图 1.1 回归分析模型训练图

1.2.2 线性回归

回归分析中有线性回归、逻辑回归、岭回归、贝叶斯回归等。其中线性回归是线性回归方程的最小二乘函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。只有一个自变量的情况称为简单回归，大于一个自变量情况的叫做多元回归。

线性回归是回归分析中第一种经过严格研究并在实际应用中广泛使用的类型。这是因为线性依赖于其未知参数的模型比非线性依赖于其未知参数的模型更容易拟合，而且产生的估计的统计特性也更易确定。

线性回归模型经常用最小二乘逼近来拟合，在此过程中需要注意正则化（或惩罚），防止得到的模型“过拟合”。

1.2.3 随机森林

随机森林算法是一种可应用于分类和回归的算法，由 LeoBreiman 于 2001 年提出，是一种以决策树为“个体学习器”的集成学习方法，特别适用于处理非线性分类问题和高维数据。它通过构建多个决策树并将它们的预测结果进行综合，从而提高分类的准确性和稳定性。在预测空气质量等级的应用中，随机森林可以考虑各种环境因素的综合影响，并通过树结构的学习来捕捉这些因素之间的复杂关系。随机森林还可以通过袋装采样和特征采样的方式来提高模型的泛化能力和鲁棒性。

随机森林原理图如图 1.3 所示。

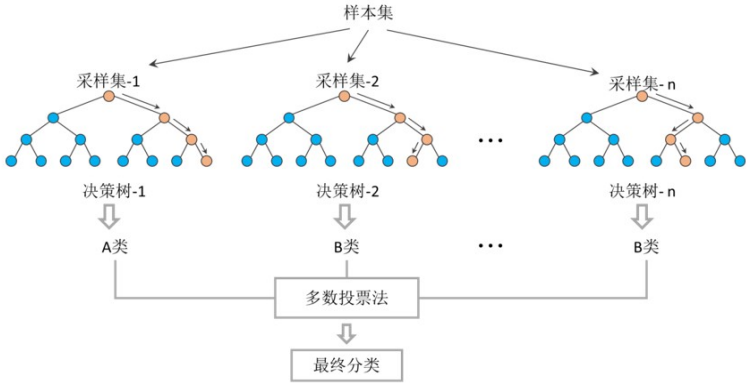

图 1.2 随机森林原理图

随机森林能够处理非线性关系和高维数据，通过集成多个决策树，提高了模型的稳定性和准确性，还能够处理缺失值和异常值。随机森林的计算成本较高，特别是在处理大量数据时。此外，它对特征选择的依赖性较强，如果特征选择不当，可能会影响模型的性能，对低维模型训练效果较差，在某些噪声比较大的样本集上，容易陷入过拟合。

1.2.4 XGBoost

XGBoost（分布式梯度增强库）是一种可扩展的基于决策树的集成学习算法，支持并行和分布式计算，并优化使用内存资源。通过构建多个决策树并逐步优化它们的权重，从而实现准确的分类。XGBoost 是

Boosting 类集成学习的典型代表，其基本思路为“基于残差的 BoostingTree”，使用梯度下降法，基于残差进行训练。在根据拥有的数据和目标变量训练出第一个决策树后，根据预测值和实际值之间的差值计算这棵树的残差。第二棵树使用残差作为目标进行训练，之后的决策树都在前一轮决策树残差的基础上进行训练，通过不断减少训练过程中产生的残差来达到数据回归的目的，最终的预测值是将第一棵树预测的目标值与所有其他树的学习率和残差的乘积相加。在预测空气质量等级的应用中，XGBoost 可以考虑各种环境因素的综合影响，并通过树结构的学习来捕捉这些因素之间的复杂关系。

相对于 AdaBoost 以权重作为每棵树的标签，XGBoost 使用梯度下降法，基于残差进行训练。XGBoost 目标函数的近似表示有两个关键技术：1）泰勒展开式，2）树结构的参数化。训练时的目标函数由两部分构成，第一部分为梯度提升算法损失，第二部分为正则化项。

$$L(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\Omega(f) = \gamma \mathrm{T} + \frac{1}{2}\lambda \|\omega\|^2$$

图 1.3 XGBoost 的目标函数

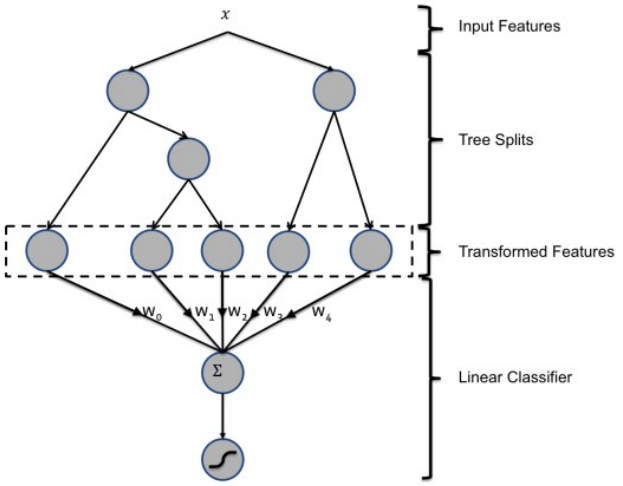

图 1.4 XGBoost 模型

1.2.5 人工神经网络

人工神经网络是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。神经网络由大量的人工神经元联结进行计算。大多数情况下人工神经网络能在外界信息的基础上改变内部结构，是一种自适应系统。

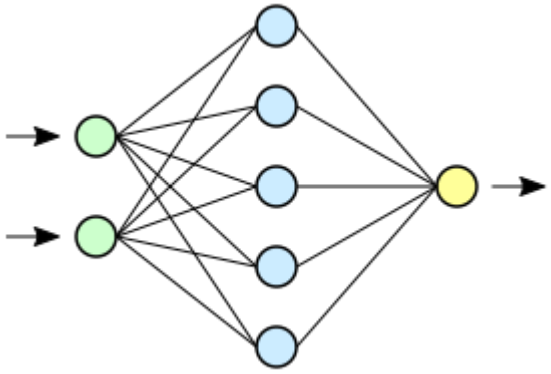

图 1.5 人工神经网络示意图

典型的人工神经网络具有以下三个部分：

结构（Architecture）结构指定了网络中的变量和它们的拓扑关系。例如，神经网络中的变量可以是神经元连接的权重（weights）和神经元的激励值（activities of the neurons）。

激励函数（Activation Rule）大部分神经网络模型具有一个短时间尺度的动力学规则，来定义神经元如何根据其他神经元的活动来改变自己的激励值。一般激励函数依赖于网络中的权重（即该网络的参数）。

学习规则（Learning Rule）学习规则指定了网络中的权重如何随着时间推进而调整。这一般被看做是一种长时间尺度的动力学规则。一般情况下，学习规则依赖于神经元的激励值。它也可能依赖于监督者提供的目标值和当前权重的值。例如，用于手写识别的一个神经网络，有一组输入神经元。输入神经元会被输入图像的数据所激发。在激励值被加权并通过一个函数（由网络的设计者确定）后，这些神经元的激励值被传递到其他神经元。这个过程不断重复，直到输出神经元被激发。最后，输出神经元的激励值决定了识别出来的是哪个字母。

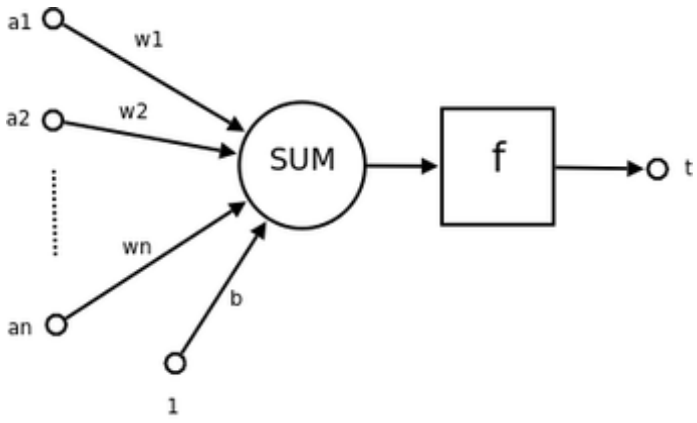

图 1.5 神经元示意图

1.2.6 回归分析评价标准

回归分析中以均方误差（MSE）和决定系数（R^2）作为最主要评价标准的之一。均方误差是“误差”的平方的期望。决定系数是相关系数的平方，用来衡量回归方程对 y 的解释程度，也是对估计的回归方程拟合优度的度量。

MSE 的计算公式：

$$MSE=\frac{\text{组内方差}}{\text{自由度}}=\frac{(y_i-\hat{y}_i)^2}{n\text{-}k}=\frac{SSE}{n\text{-}k}$$

图 1.6 MSE 计算公式

R^2 的计算公式：

$$r^2=\frac{\sum(\hat{y}_i-\bar{y})^2}{\sum(y_i-\bar{y})^2}=1-\frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\bar{y})^2}$$

图 1.7 R^2 计算公式

1.2.7 Web 框架 Flask

Flask 是一个轻量级的 Web 应用框架，它的核心是 WSGI（Web Server Gateway Interface）服务器和路由系统，由 Armin Ronacher 开发，使用 Python 语言编写。它被设计为易于扩展和定制，同时保持核心简

单。Flask 适用于构建从小型到中型的 Web 应用程序，并且可以轻松地与各种数据库和其他 Web 工具集成。Flask 被称为“微框架”，因为它使用简单的核心，用扩展增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。然而，Flask 保留了扩增的弹性，可以用 Flask-extension 加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

图 1.7 Flask 商标

1.3 机器学习一般过程

数据分析包括数据理解、数据准备、模型训练、模型评估、模型优化与重选、模型应用等六个步骤。每个步骤都是非常关键。

• 数据理解：了解数据的特征、分布情况、缺失值等，以便更好地理解数据。

• 数据准备：对数据进行清洗、转换、处理，使数据能被模型所使用。

• 模型训练：选择合适的模型，并使用训练数据对模型进行训练。

• 模型评估：使用验证算法选择最优的模型。

• 模型优化与重选：根据评估结果对模型进行调优，以提高模型性能。

• 模型应用：将优化后的模型应用于真实的数据，从中获得实际价值。

1.3.1 数据理解

数据理解是数据分析过程中最重要部分之一,对数据集的全面了解和分析，以便更好地进行后续的处理、分析和建模。

1.3.2 数据准备

数据准备是指在进行数据分析和建模之前,对原始数据进行清洗、

转换、集成和规约等操作的过程。数据预处理的目的是提高数据的质量,使数据更加适合进行分析和建模。特征工程是机器学习过程中非常重要的一环，它是指通过对原始数据进行加工和处理，将其转化为可供机器学习算法使用的特征集合。

数据清洗：比如在数据采集过程中，可能会包含噪声、异常值、缺失值或重复数据。数据是清洗识别和纠正这些问题的过程，以确保数据的质量。

数据转换：比如有时收集到的数据需要转换或格式化，以便于机器学习算法能够处理。

数据分析：比如说使用均值、中位数、标准差等统计量来描述数据的基本特性。

数据集划分：比如将数据集划分为训练集和测试集，其中训练集用于训练模型，测试集用于评估模型的性能。

1.3.3 模型训练

在完成数据准备和特征工程之后,就可以开始选择和训练机器学习模型了。在这个阶段，可以根据不同的问题类型、数据结构和业务需求等因素来选择合适的模型，并利用训练数据对模型进行训练和调优。

1.3.4 模型评估

完成模型的训练后，使用测试集来评估模型的性能，以了解模型在未知数据上的表现。

1.3.5 模型优化与重选

在完成模型评估之后，如果模型的性能没符合要求，需要重选模型或进行不断地优化和调整，以保证其能够持续地提供最优的预测结果。

1.3.6 模型应用

如果模型的性能符合要求，就可以开始将其部署到实际场景中。但在这之后还需要不断地更新数据以维护模型的准确性。

2、需求分析

2.1 可行性分析

本系统所需要的数据集均可获取，数据来源于医疗机构、睡眠研究中心或第三方数据服务提供商。通过数据收集工具获得历史睡眠效率数据、健康相关数据（如年龄、性别、饮食习惯）、生活习惯数据（如运动频率、咖啡因摄入）等与睡眠效率相关的多维度信息。获取的数据有足够的时间跨度和细粒度（如每日数据），保证了模型训练的准确性和可靠性。逻辑回归、随机森林和 XGBoost 等算法在许多领域已有成功应用，因此本系统在实现方面不存在问题。其次，本课题主要是软件方面的实现，不需要额外的硬件费用。此外，所有数据的采集和使用符合当地法律和政策，特别是涉及个人隐私和数据保护的规定，因此，不存在任何利益纠纷问题。

2.2 系统功能需求分析

为确保系统满足用户需求并提供实用价值，以下是详细需求分析。

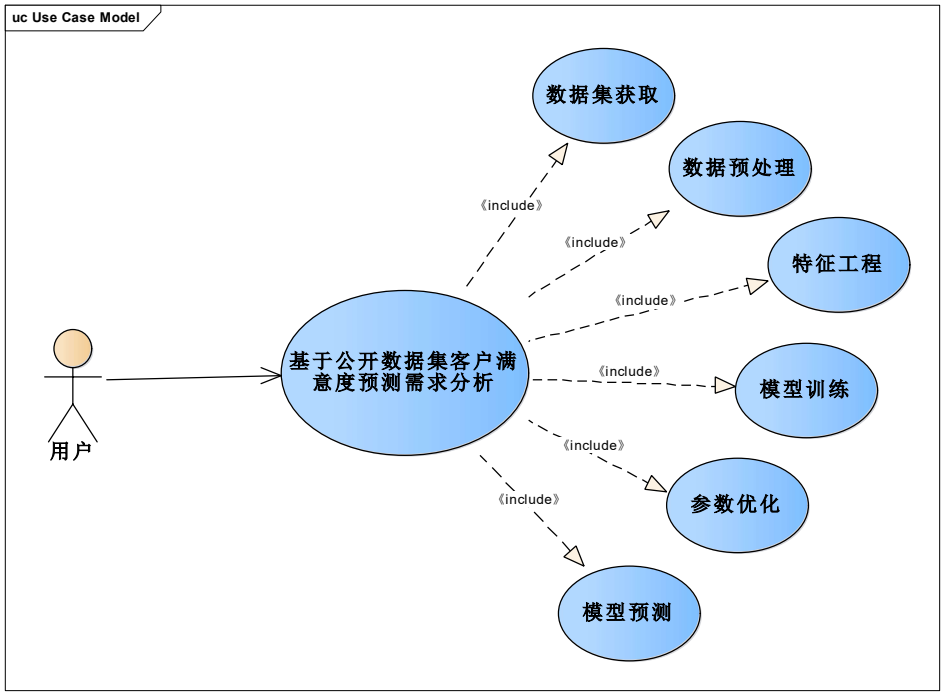

图 2.1 需求分析用例图

在数据预处理方面，我们需要处理缺失值、异常值和数据格式标准化。在特征工程方面，我们根据领域知识提取相关特征，可能包括年龄、性别、

睡眠时长、饮食习惯（如咖啡因和酒精摄入量）、生活习惯（如运动频率和吸烟状态）等。在模型构建与评估方面，我们需要根据数据特性和初步分析结果，选择合适的机器学习模型，如多元线性回归、随机森林或 XGBoost。使用交叉验证等技术进行超参数调优，以达到最优模型性能。使用 R^2、MES 等指标评估模型性能，确保模型具有高准确度和可靠性。

在输入界面方面，用户可以通过简单友好的界面输入个人基本信息（如年龄、性别）、生活习惯（如运动频率、咖啡因摄入量等）。在结果显示方面，系统能够清晰展示预测结果，包括睡眠效率评分和相关的健康建议。在实时查询方面，用户可以随时查询特定日期的睡眠效率预测。在数据可视化方面，提供直观的图表和报告，帮助用户更好地理解预测结果和睡眠效率趋势。最后，我们设计反馈机制，收集用户对预测结果的满意度和准确性反馈，并根据用户反馈调整和优化模型，持续改进系统性能和用户体验。

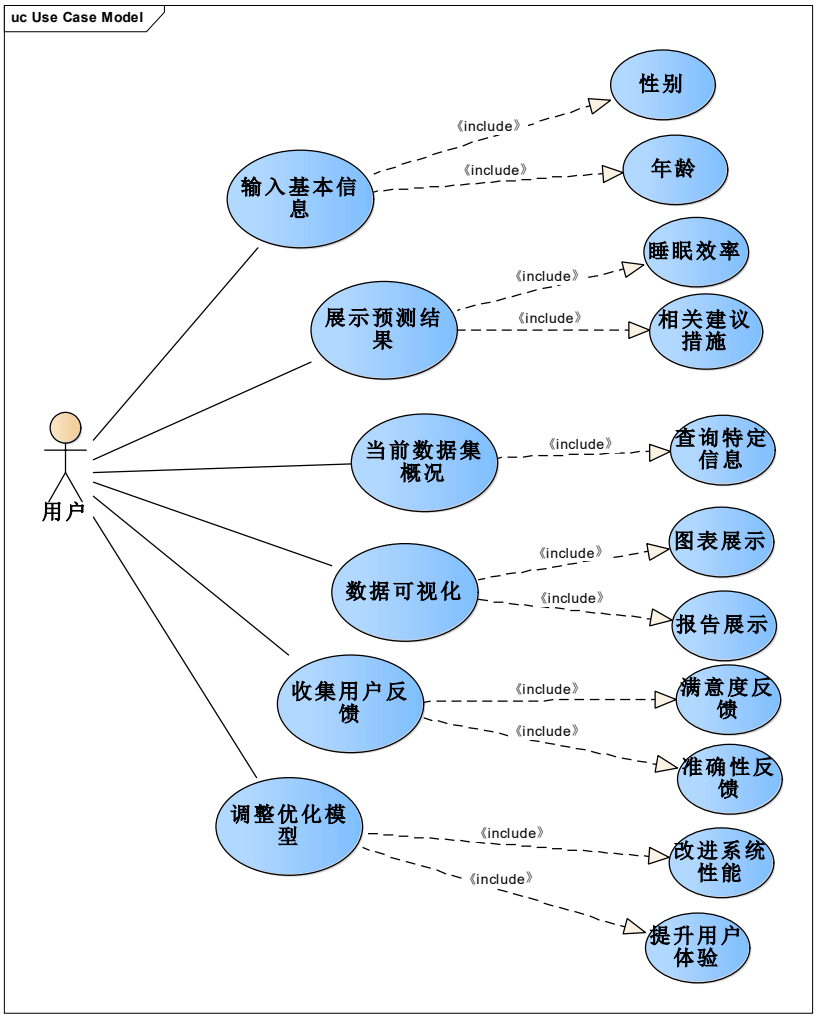

图 2.2 操作模块需求分析用例图

3、系统设计

3.1 总体设计

3.1.1 系统技术架构设计

图 3.1 展示了算法模块与 Web-Flask 技术架构之间的关联，其中算法模块负责处理数据和执行预测逻辑，而 Flask 则负责接收 HTTP 请求，调用算法模块，并将结果返回给前端用户界面。这种分层的设计不仅使得系统结构清晰，易于维护，也便于未来的扩展和优化。

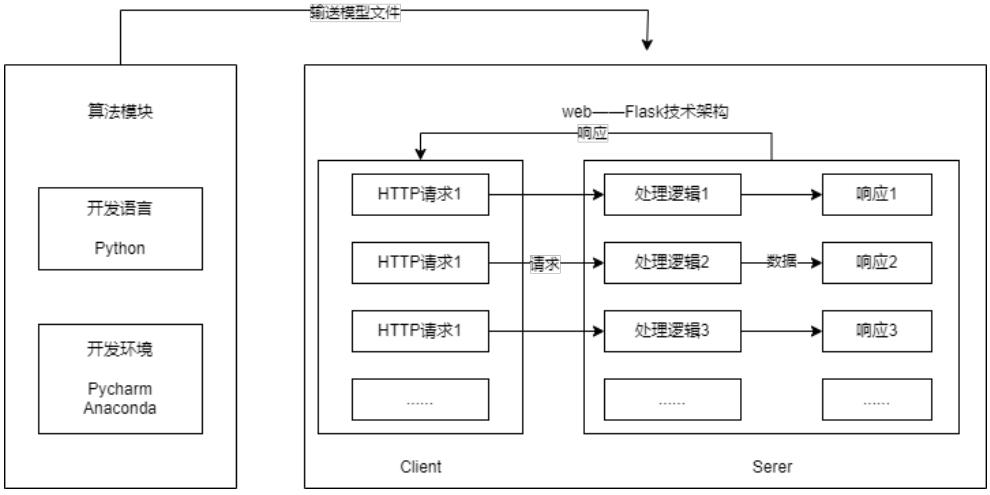

图 3.1 系统技术架构图

在本课题中，采用 Python 语言结合 PyCharm 和 Anaconda 开发环境，致力于算法模块的开发，重点在于睡眠效率预测模型的训练、评估及参数调优。我们首先在本地开发环境中，利用 Python 的强大数据处理和机器学习库，对收集到的历史睡眠效率数据进行预处理和特征工程，之后训练多种预测模型，如线性回归、随机森林、神经网络等，通过交叉验证和性能指标来比较不同模型的表现，最终选定表现最佳的模型并保存其参数。

睡眠效率预测 Web 系统的核心功能在于，它能够调用上述保存的模型文件，实现对不同年龄睡眠效率的预测。当用户在前端界面输入年龄或性别并触发预测请求时，系统会通过 HTTP 协议向后端发送请求。后端接收到请求后，立即调用最优模型进行预测计算，处理完毕后将预测结果及睡眠效率指数以响应的形式返回给前端。前端页面随即更新显

示预测的睡眠效率状况，以及整体睡眠效率等级，帮助用户了解并应对可能的健康风险。

3.1.2 功能模块结构设计

本系统分为二个模块，第一个模块为基于公开数据集的睡眠效率预测模块，第二模块为用户操作模块。第一个模块为算法模块设计，采用了系统化的数据处理流程，从数据读取开始，针对不同数据集的特点实施精细化的预处理和特征工程，确保数据的质量与适用性。通过智能算法的选择与模型训练，结合参数调优策略，不断迭代优化，以期达到最优的预测性能。这一过程严格遵循科学的评估体系，采用指标进行全面评估，并借助可视化工具深入分析模型效能，最终锁定表现最为出色的模型版本，作为最优模型。

用户操作模块：在算法模块完成后，构建了一个直观且友好的 Web 界面，用户只需输入目标，即可一键发起预测请求。系统后端即时响应，调用前述算法模块中的最优模型，对指定区域的睡眠效率进行精准预测，迅速返回预测结果。为用户提供全面的概况，助力其做出健康决策。系统功能模块结构，如图 3.2 所示，清晰地展示了数据流与控制流的交互逻辑，确保了从数据处理到用户反馈的每一个环节都高效、准确，充分体现了本系统在技术实现与用户体验设计上的双重追求。

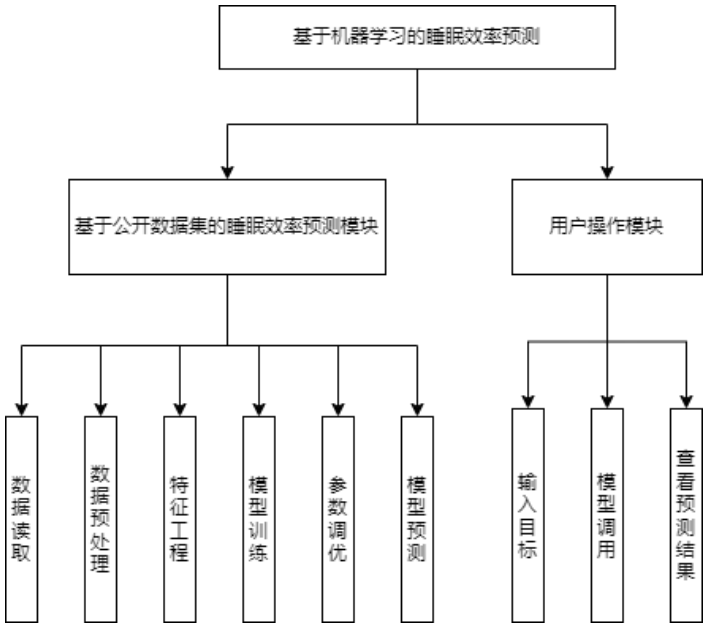

图 3.2 系统功能模块结构图

3.1.3 系统工作流程设计

在“基于公开数据集的睡眠效率预测模块”，对公开数据集进行“数据读取-数据预处理-特征工程-模型训练-参数调优-模型评估-模型预测”等工作，数据获取，从公开数据集中获取原始数据，这些数据可能包含有关睡眠效率和相关影响因素的各种信息。数据预处理，对获取的数据进行预处理，包括处理重复值、缺失值，进行特征编码和离群值处理。特征工程，在此阶段，提取关键特征，并对这些特征进行标准化和归一化处理，以确保所有变量具有相似的尺度。模型训练，使用多种机器学习算法对数据进行训练，以构建预测模型。参数调优，对各个模型进行参数调优，以寻找最佳的超参数配置，提高模型的预测准确性。模型评估，评估每个模型的性能，比较它们的预测效果，选择表现最好的模型。模型预测，最终，使用选定的最佳模型进行预测，得出睡眠效率的未来趋势。保存模型参数，将得到的最优模型及其参数保存在文件中，便于后续使用和更新。开发一个睡眠效率预测 Web 系统，允许用户输入他们的性别和年龄。生成目标和预测，根据用户的输入，系统自动生成目标，并使用最佳模型进行预测，显示预测结果。

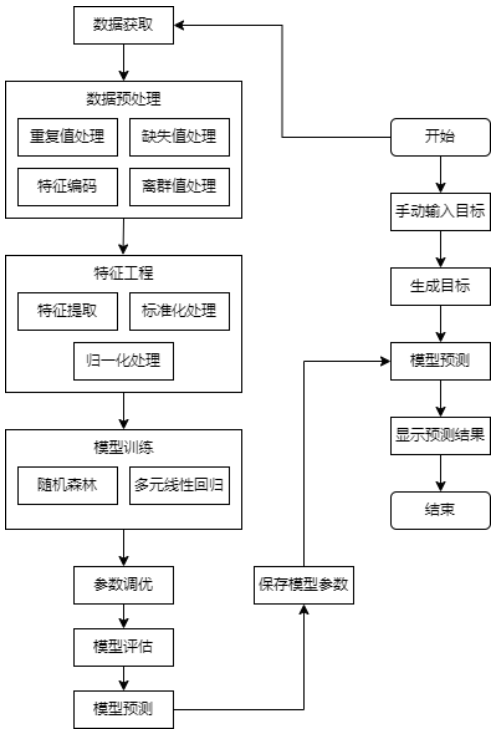

图 3.3 系统流程图

3.2 详细设计

3.2.1 基于公开数据集的满意度预测设计

图 3.4 展示了睡眠效率预测模块的基本工作流程，开始，从公开数据集读取数据集，通常使用 pandas 库中的 read_csv()函数。数据预处理，删除重复数据：使用 pandas 库的 drop_duplicates()函数。缺失值处理，使用 dropna()和 fillna()函数。特征编码，使用 sklearn.preprocessing 库的 LabelEncoder 类。离群值处理，调用 remove_outlier()函数。标准化处理，使用 sklearn.preprocessing 库的 StandardScaler 类。特征提取，提取出有效特征信息，准备用于模型训练。模型训练，使用 sklearn 库中的机器学习模型（多元线性回归和随机森林）进行二分类和五分类预测。参数调优，对每个模型进行参数调优，以提高模型的预测性能。模型评估，计算并可视化四个模型的评估指标，包括准确率（accuracy）、召回率（recall）、精确率（precision）和 F1 得分（f1-score）。模型选择，根据评估结果，选择表现最优的模型。模型保存，将最优模型保存为文件，供后续使用。模型预测,利用选定的最优模型进行睡眠效率预测。完成整个预测流程。

这个模块的设计目的是根据公开数据集构建一个可靠的睡眠效率预测工具，通过一系列的数据处理和模型训练，最终输出一个准确的预测模型。用户可以根据实际需求，输入特定的目标，获取对应的预测结果。

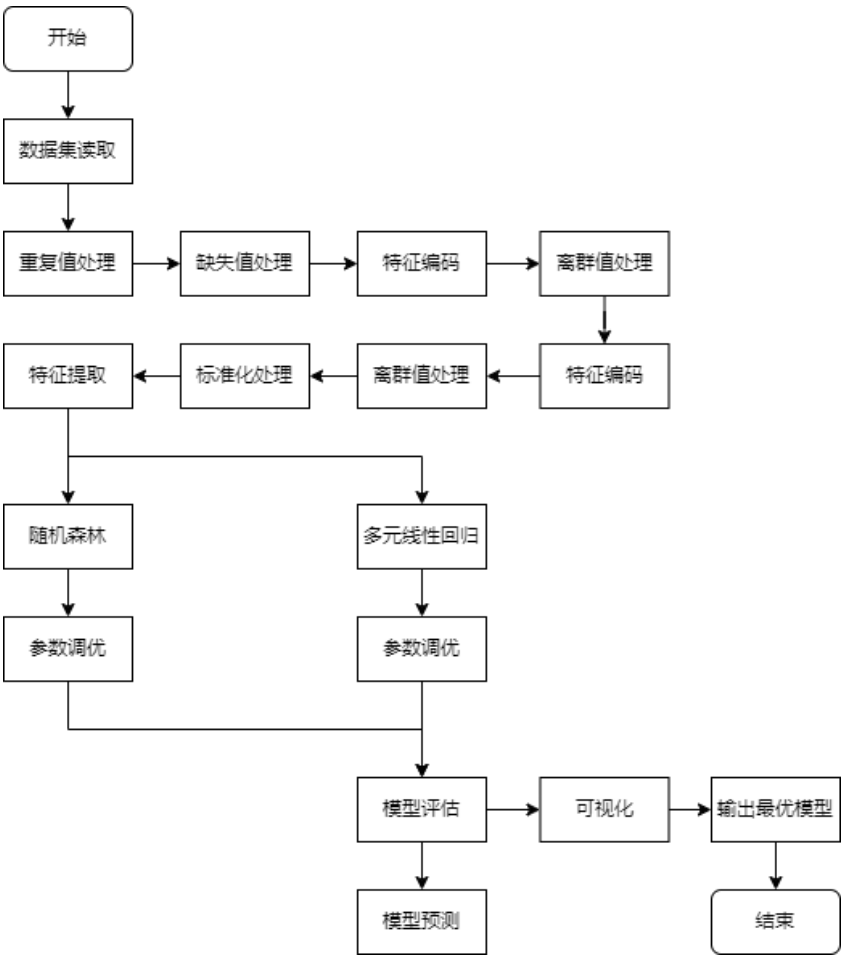

图 3.4 基于公开数据集模块流程图

3.2.2 用户操作模块

图 3.5 描述了一个睡眠效率预测 Web 系统的操作流程，开始，用户启动睡眠效率预测 Web 系统。手动输入目标，性别和年龄。生成目标，系统根据用户输入的关键影响因素生成相应的预测目标。模型预测，系统调用预先训练好的最优模型（基于公开数据集上的数据分析和建模方法）进行预测。显示预测结果，系统根据模型预测的结果，显示当前年龄的睡眠效率状况的预测等级。结束预测过程。

这个流程旨在帮助用户自己当前的睡眠效率状况，以便采取适当的健康观测或者规划活动。通过这样的方式，用户可以获得个性化的睡眠效率分析 1。

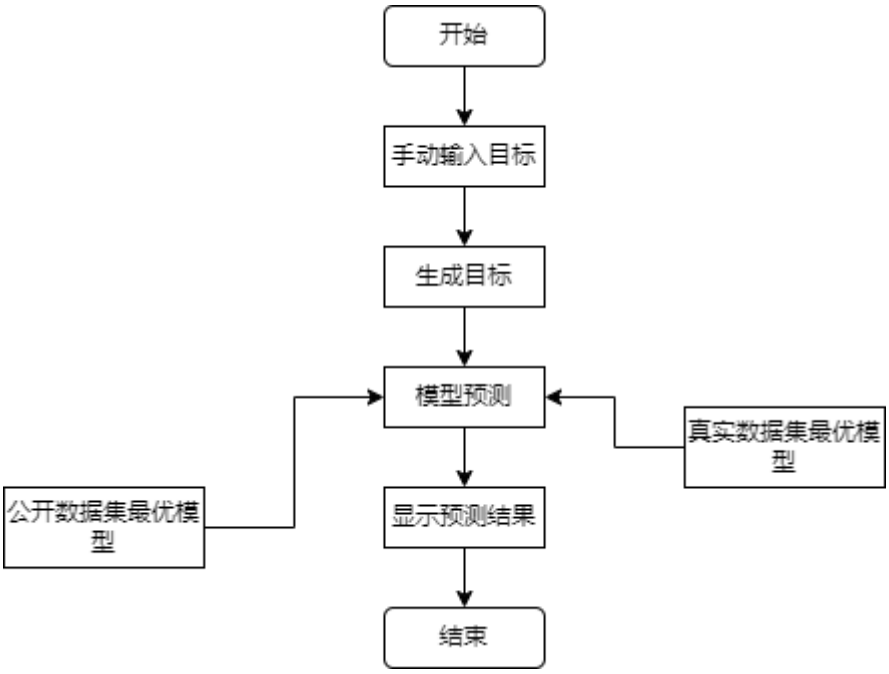

图 3.5 用户操作模块流程图

4、系统的实现与测试

4.1 数据理解

（1）数据集来源：

睡眠效率数据集是一份专注于探究个体睡眠质量与生活习惯关联的资源，由一组专注于人工智能研究的学者在摩洛哥通过 ENSIAS 工程学院精心收集而成。

数据采集方法论涉及一个综合性的研究设计，旨在深入剖析诸如咖啡因摄入、酒精消费及运动习惯等生活方式因素如何影响个人的睡眠模式与睡眠质量。研究团队从当地社区招募志愿者，并在数月期间持续积累数据。数据收集过程中，结合了自我报告问卷、活动记录仪监测及多导睡眠监测等手段，后者是一种精确的睡眠监测技术。这一多元化的数据收集方法确保了研究结果的全面性和准确性。该睡眠效率数据集囊括了关键指标。

本项目旨在利用大规模睡眠效率数据集，通过分析个体年龄与其睡眠效率之间的关联性，建立预测模型，实现对特定年龄群体睡眠效率的预估，为个性化睡眠健康管理与干预提供数据支撑，提升公众睡眠质量与生活福祉。

（2）数据集介绍：

本数据集涵盖了广泛的睡眠模式调研信息，旨在通过分析生活习惯对睡眠质量的影响，特别是关注年龄与睡眠效率之间的关系，共收集了 452 位参与者的详细睡眠数据，每个样本包含以下 9 个核心属性，用于支持对特定年龄群体睡眠效率的预测与分析。

该数据集涵盖了一群测试对象及其睡眠模式信息。每个测试对象通过唯一的 ID 进行识别，并记录了他们的年龄和性别。“就寝时间”和“醒来时间”表明了每个受试者每天的睡觉和起床时间，“睡眠时长”记录了每个受试者的总睡眠时间，单位为小时。“睡眠效率”是衡量实际躺在床上的时间中有多少比例真正用于睡眠的一个指标。“REM 睡眠占比”、“深度睡眠占比”和“浅度睡眠占比”分别表示每个受试者在各个睡眠阶段所花费的时间比例。“觉醒次数”记录了每个受试者夜间醒来次数的信息。此外，数据集还包含了关于每个受试者在睡前 24 小时内咖啡因和酒精摄入情况、吸烟状况以及锻炼频率的信息。

表 1 睡眠效率数据集属性介绍

属性	含义	示例	描述
ID	参与者编号	250	1-452
Age	年龄	28	9-69：覆盖广泛年龄层
Gender	性别	Male/ Female	男性/女性
Bedtime	就寝时间	2021-12-05 23:00:00	格式为 YYYY-MM-DD HH:mm:ss，记录日常就寝时间
Wakeup time	起床时间	2021-12-06 07:30:00	格式为 YYYY-MM-DD HH:mm:ss，记录日常起床时间
Sleep duration	睡眠时长	8	小时为单位，记录每晚平均睡眠时长
Sleep efficiency	睡眠效率	0.85	小数形式，反映实际睡眠时间占卧床时间的比例，范围 0.5-0.99

属性	含义	示例	描述
REM sleep percentage	REM 睡眠占比	25	数字形式，表示 REM 睡眠在总睡眠时间中的比例，范围 15-30
Deep sleep percentage	深度睡眠占比	20	数字形式，表示深度睡眠在总睡眠时间中的比例，范围 18-75
Light sleep percentage	浅度睡眠占比	55	数字形式，表示浅度睡眠在总睡眠时间中的比例，范围 7-63
Awakenings	觉醒次数	2	夜间醒来次数
Caffeine consumption	咖啡因摄入	50	mg 为单位，表示睡前 24 小时摄入的咖啡因含量
Alcohol consumption	酒精摄入	3	oz 为单位，表示睡前 24 小时摄入的酒精含量
Smoking consumption	吸烟状况	No	Yes/No，抽烟/不抽烟
Exercise frequency	锻炼频率	3	每周锻炼频次

此数据集不仅提供了丰富的个体睡眠行为数据，还为探索年龄、性别等人口统计学因素如何影响睡眠效率、以及不同睡眠阶段的分布提供了详实的资料，为睡眠健康管理和干预策略的制定奠定了坚实的数据基础。

4.2 数据准备

数据理解

首先，我们通过 pandas 库读取名为 Sleep_Efficiency.csv 的睡眠效率数据集，并进行了初步的数据探索。

	ID	Age	Gender	Bedtime	Wakeup time	Sleep duration	Sleep efficiency	REM sleep percentage	Deep sleep percentage	Light sleep percentage	Awakenings	Caffeine consumption	Alcohol consumption	Smoking status	Exercise frequency
0	1	65	Female	2021-03-06 01:00:00	2021-03-06 07:00:00	6.0	0.88	18	70	12	0.0	0.0	0.0	Yes	3.0
1	2	69	Male	2021-12-05 02:00:00	2021-12-05 09:00:00	7.0	0.66	19	28	53	3.0	0.0	3.0	Yes	3.0
2	3	40	Female	2021-05-25 21:30:00	2021-05-25 05:30:00	8.0	0.89	20	70	10	1.0	0.0	0.0	No	3.0
3	4	40	Female	2021-11-03 02:30:00	2021-11-03 08:30:00	6.0	0.51	23	25	52	3.0	50.0	5.0	Yes	1.0
4	5	57	Male	2021-03-13 01:00:00	2021-03-13 09:00:00	8.0	0.76	27	55	18	3.0	0.0	3.0	No	3.0

图 4.1 data.head()

通过.head()函数查看数据集的前几行，以直观了解数据结构；

	ID	Age	Sleep duration	Sleep efficiency	REM sleep percentage	Deep sleep percentage	Light sleep percentage	Awakenings	Caffeine consumption	Alcohol consumption	Exercise frequency
count	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	452.000000	432.000000	427.000000	438.000000	446.000000
mean	226.500000	40.285398	7.465708	0.788916	22.615044	52.823009	24.561947	1.641204	23.653396	1.173516	1.791480
std	130.625419	13.172250	0.866625	0.135237	3.525963	15.654235	15.313665	1.356762	30.202785	1.621377	1.428134
min	1.000000	9.000000	5.000000	0.500000	15.000000	18.000000	7.000000	0.000000	0.000000	0.000000	0.000000
25%	113.750000	29.000000	7.000000	0.697500	20.000000	48.250000	15.000000	1.000000	0.000000	0.000000	0.000000
50%	226.500000	40.000000	7.500000	0.820000	22.000000	58.000000	18.000000	1.000000	25.000000	0.000000	2.000000
75%	339.250000	52.000000	8.000000	0.900000	25.000000	63.000000	32.500000	3.000000	50.000000	2.000000	3.000000
max	452.000000	69.000000	10.000000	0.990000	30.000000	75.000000	63.000000	4.000000	200.000000	5.000000	5.000000

图 4.2 data.describe()

使用.describe()方法概括性地展示了数据的统计摘要，如各数值型列的计数、均值、标准差、最小值、四分位数及最大值；

```
Index(['ID', 'Age', 'Gender', 'Bedtime', 'Wakeup time', 'Sleep duration',
       'Sleep efficiency', 'REM sleep percentage', 'Deep sleep percentage',
       'Light sleep percentage', 'Awakenings', 'Caffeine consumption',
       'Alcohol consumption', 'Smoking status', 'Exercise frequency'],
      dtype='object')
```

图 4.3 data.columns()

data.columns 显示了所有列名，帮助我们了解数据集包含的特征。

```
data.shape
```

```
(452, 15)
```

图 4.4 data.shape()

data.shape 属性提供了数据集的行数和列数，即样本量和特征数量

预处理

其次，调用 drop_duplicates()函数删除重复数据。在了解数据的基本情况后，针对数据中存在的缺失值问题，采取了直接删除含有空值行的策略，使用 dropna(inplace=True)方法，确保后续分析基于完整无缺的数据进行。这一步骤保证了模型训练时不会因缺失数据导致的错误或偏差。

```
#支持中文
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
matplotlib.rcParams['axes.unicode_minus'] = False

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

#数据清洗
data.drop_duplicates()
data.dropna(inplace=True)
```

图 4.5 数据预处理代码

特征工程

为了使数据更适用于模型输入，进行了以下关键的特征处理步骤：

```
# 特征工程
features = ['Age', 'Gender', 'Sleep duration', 'REM sleep percentage', 'Deep sleep percentage',
            'Light sleep percentage', 'Awakenings', 'Caffeine consumption',
            'Alcohol consumption', 'Smoking status', 'Exercise frequency']

# 将性别转换为数值
data['Gender'] = data['Gender'].map({'Male': 0, 'Female': 1})

# 将吸烟状态转换为数值
data['Smoking status'] = data['Smoking status'].map({'No': 0, 'Yes': 1})


# 提取特征和目标变量
X = data[features].values
y = data['Sleep efficiency'].values

# 标准化数据 帮助模型更快地收敛，提高模型的性能
scaler = StandardScaler()
X = scaler.fit_transform(X)

# 转换为 PyTorch 张量
X = torch.tensor(X, dtype=torch.float32)

# 转换为一个二维张量，使其形状变为 [样本数量, 1]
y = torch.tensor(y, dtype=torch.float32).view(-1, 1)
```

图 4.6 特征工程代码

特征编码：将类别型特征（如性别和吸烟状态）转换为数值型数据，以适应模型的输入要求。例如，将“Gender”映射为男性为 0，女性为 1，类似的处理也应用到了“Smoking status”。

特征选择：从原始数据集中选择了与睡眠效率预测高度相关的多个特征，包括年龄、性别、各类睡眠时长百分比、唤醒次数、咖啡因摄入量、酒精摄入量、吸烟状况和锻炼频率等，构建了模型的输入特征矩阵`X`。

标准化：使用“StandardScaler”对特征进行了标准化处理，缩小了不同特征间尺度的差异，使得每个特征具有相同的权重，避免了因量纲不同导致的模型学习偏斜。

特征转换：将处理后的特征和目标变量（睡眠效率）转换为 PyTorch 张量，便于利用深度学习框架进行模型训练。

4.3 模型训练

首先尝试以多元线性回归算法做预测。将数据集以二八比划分为训练集和测试集，并使用 DataLoader 创建训练和测试数据加载器，训练加载器打

乱数据以增强模型泛化能力，测试加载器保持数据顺序以确保评估的稳定性，每批次包含 32 个样本。

```
# 划分数据集
dataset = TensorDataset(X, y)

# 80%训练模型，20%评估模型
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size

# random_split数据集随机划分
train_dataset, test_dataset = random_split(dataset, [train_size, test_size])

# DataLoader创建训练加载器 batch_size每个批次包含32个样本 shuffle打乱数据以提高模型的泛化能力
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

# DataLoader创建评估加载器 batch_size每个批次包含32个样本 shuffle不打乱数据以确保评估的一致性
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

图 4.7 划分数据集代码

构建了一个基于 PyTorch 的线性回归模型，用于预测睡眠效率。模型设计包括线性变换层和 ReLU 激活函数，以引入非线性，提升模型表达能力。

```
# 定义模型
# nn.Module可以自动梯度计算、模型保存
class LinearRegressionModel(nn.Module):

    # input_dim 输入特征的维度
    def __init__(self, input_dim):
        super(LinearRegressionModel, self).__init__()

        # 定义线性层，输入维度为 input_dim，输出维度为 1
        self.linear = nn.Linear(input_dim, 1)

        # 定义ReLU激活函数缓解过拟合问题， 能够在x>0时保持梯度不衰减，从而缓解梯度消失问题
        self.relu = nn.ReLU()

    # 定义前向传播逻辑
    def forward(self, x):

        # 线性层对输入数据进行线性变换后传递给ReLU激活函数，得到非线性变换后的输出
        return self.relu(self.linear(x))
```

图 4.8 构建模型代码

采用随机梯度下降(SGD)作为优化器，并设置了学习率衰减策略，以期在训练过程中不断调整学习率，寻找更优解。

```
input_dim = X.shape[1]
model = LinearRegressionModel(input_dim)

# 定义损失函数均方误差，预测与实际之间的平均平方差
criterion = nn.MSELoss()

# 优化器 SGD=随机梯度下降 lr=学习率 momentum=动量加速，跳过局部最小值
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

# 学习率指数衰减 gamma=衰减率
scheduler = optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.95)
```

图 4.9 随机梯度下降代码

接着就是训练模型。

```
# 训练模型
num_epochs = 100  # 迭代100次
epoch_loss = []  # 存储每个 epoch 的损失
for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    for inputs, targets in train_loader:

        # 前向传播
        outputs = model(inputs)
        loss = criterion(outputs, targets)

        # 反向传播和优化
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(train_loader)
    epoch_loss.append(avg_loss)

    # 更新学习率
    scheduler.step()

    if (epoch + 1) % 10 == 0:
        print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {avg_loss:.4f}')
```

图 4.10 模型训练代码

4.4 模型评估

训练过程中，通过多轮迭代（epochs），不断优化模型参数以减小预测误差（均方误差 MSE）。评估阶段，利用测试集数据计算模型的 MSE 和决定系数 R^2，以量化模型的预测能力和拟合优度。

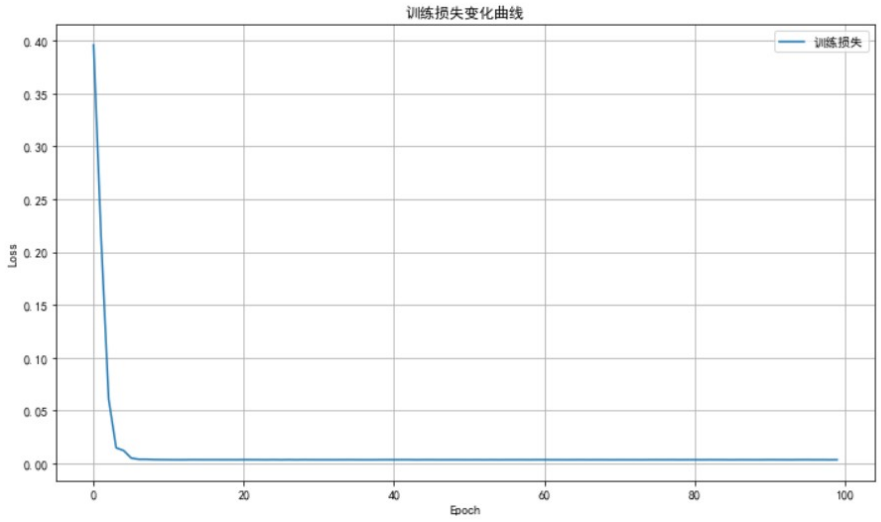

图 4.11 损失变化曲线

```
均方误差 (MSE): 0.0033806723076850176
决定系数 (R²): 0.7736834343703285
```

图 4.12 模型评估结果

4.5 模型应用

应用模型，输出折线图对比。

```
# 绘制预测结果的折线图
plt.figure(figsize=(10, 6))
plt.plot(y_true, label='真实值', marker='o', linestyle='-', color='r')
plt.plot(y_pred, label='预测值', marker='x', linestyle='--', color='b')
plt.xlabel('测试样本索引')
plt.ylabel('睡眠效率')
plt.title('真实值与预测值对比')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

图 4.13 可视化代码

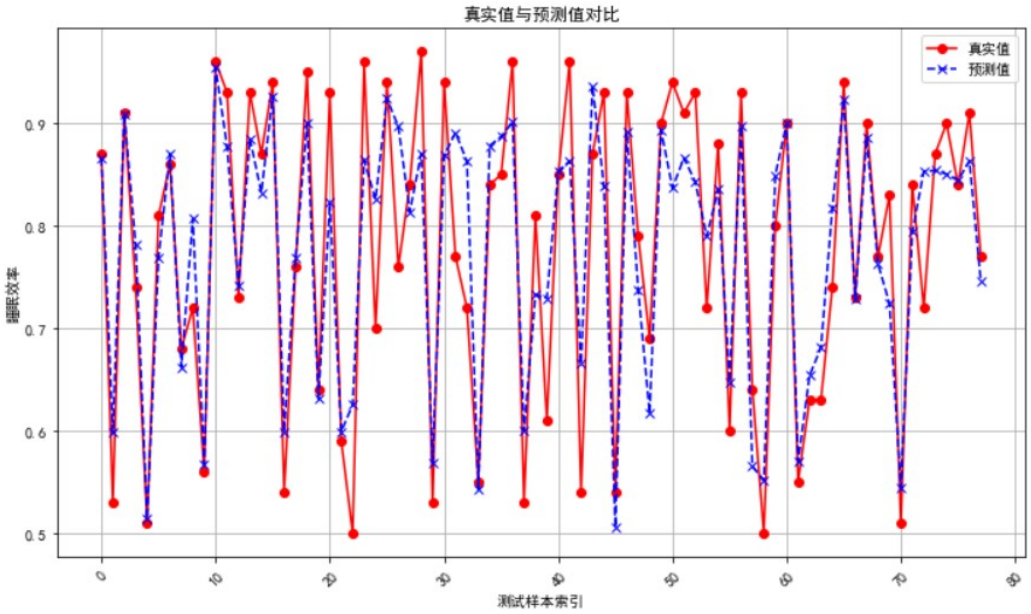

图 4.14 可视化结果

4.6 模型优化与重选

以相似的方法，我们分别使用了 pytorch 做了神经网络算法和使用 sklearn 做了随机森林和 XGBoost 的。它们的评价比较如下，随机森林算法最好，所以我们选用了随机森林算法。

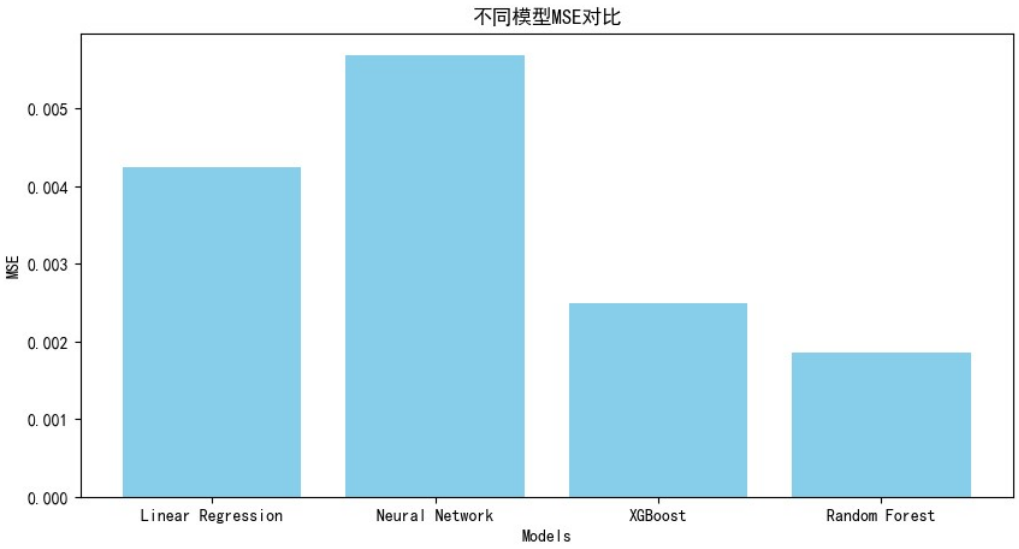

图 4.15 MSE 对比

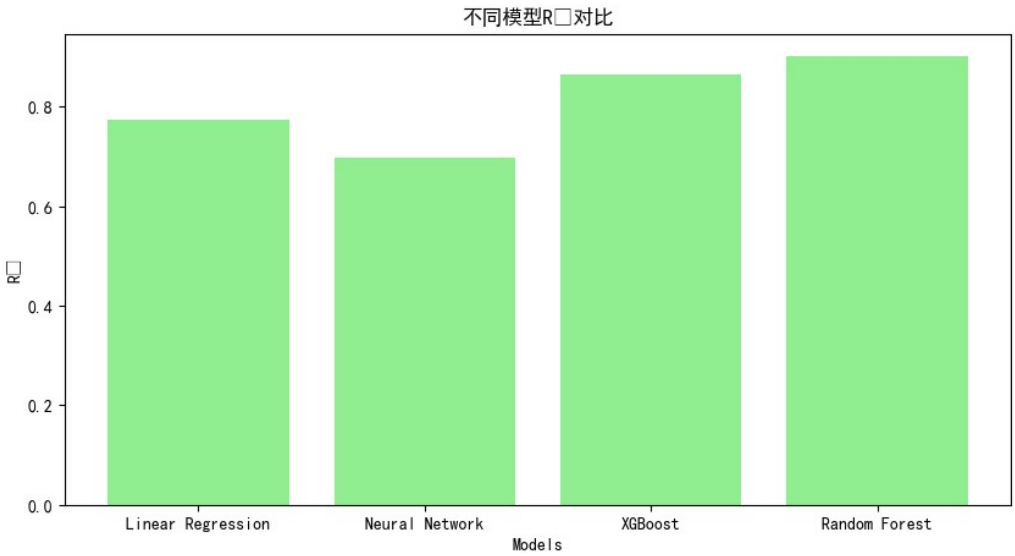

图 4.16 R^2 对比

4.7 前端框架

1. 关键影响因素与选择项

```python
def predict_sleep_efficiency(age, gender, model, scaler, default_values):
    """
    使用模型预测特定年龄和性别的睡眠效率。

    参数:
    - age: 年龄
    - gender: 性别 (0: 男性, 1: 女性)
    - model: 训练好的xgb模型
    - scaler: 标准化器
    - default_values: 特征默认值

    返回:
    - 预测的睡眠效率
    """
    new_data = {
        'Age': age,
        'Gender': gender,
        'Sleep duration': default_values['Sleep duration'],
        'REM sleep percentage': default_values['REM sleep percentage'],
        'Deep sleep percentage': default_values['Deep sleep percentage'],
        'Light sleep percentage': default_values['Light sleep percentage'],
        'Awakenings': default_values['Awakenings'],
        'Caffeine consumption': default_values['Caffeine consumption'],
        'Alcohol consumption': default_values['Alcohol consumption'],
        'Smoking status': default_values['Smoking status'],
        'Exercise frequency': default_values['Exercise frequency']
    }

    new_data_values = np.array(list(new_data.values())).reshape(1, -1)
    new_data_scaled = scaler.transform(new_data_values)
    predicted_efficiency = model.predict(new_data_scaled)
    return predicted_efficiency[0]
```

图 4.17 模型预测代码截图

这段代码定义了一个函数 predict_sleep_efficiency，用于预测特定年龄和性别的睡眠效率。函数接受四个参数：年龄 (age)、性别 (gender，0 表示男性，1 表示女性)、训练好的 XGBoost 模型 (model) 和标准化器 (scaler)，以及特征默认值 (default_values)。函数通过组合输入的年龄和性别，以及其他默认的特征值，创建一个新的数据点，随后对该数据点进行标准化处理，并使用训练好的模型进行预测，最后返回预测的睡眠效率

2. 系统界面

1、效果展示

- **预测睡眠效率**

图 4.18

- 多元线性回归

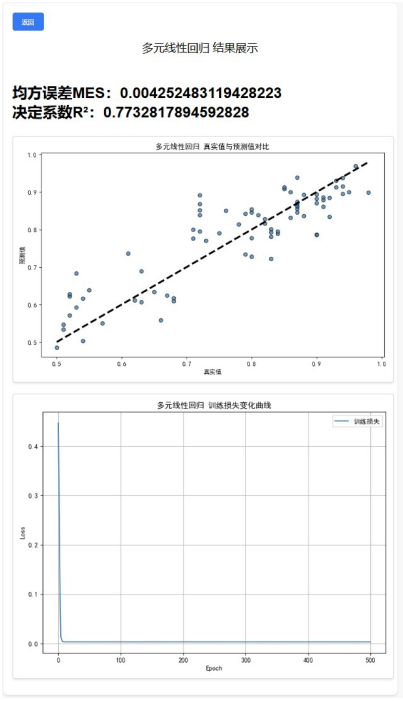

图 4.19

- 神经网络回归

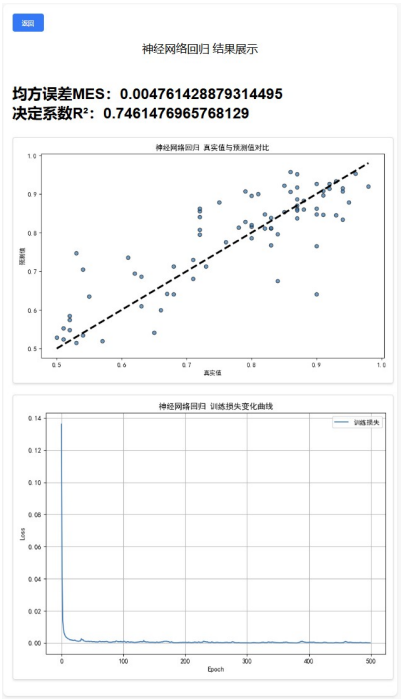

图 4.20

- XGBoost 回归

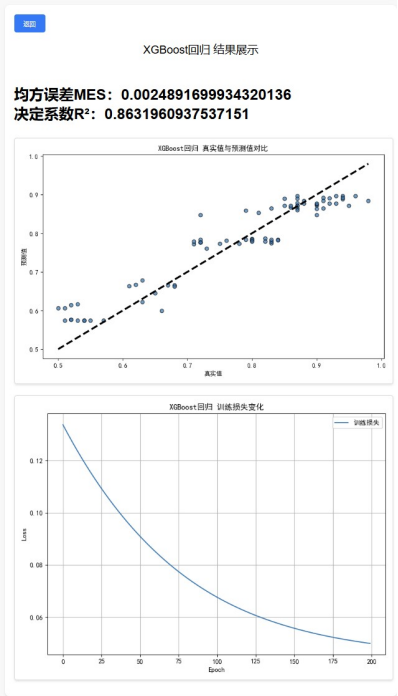

图 4.21

- **随机森林回归、**

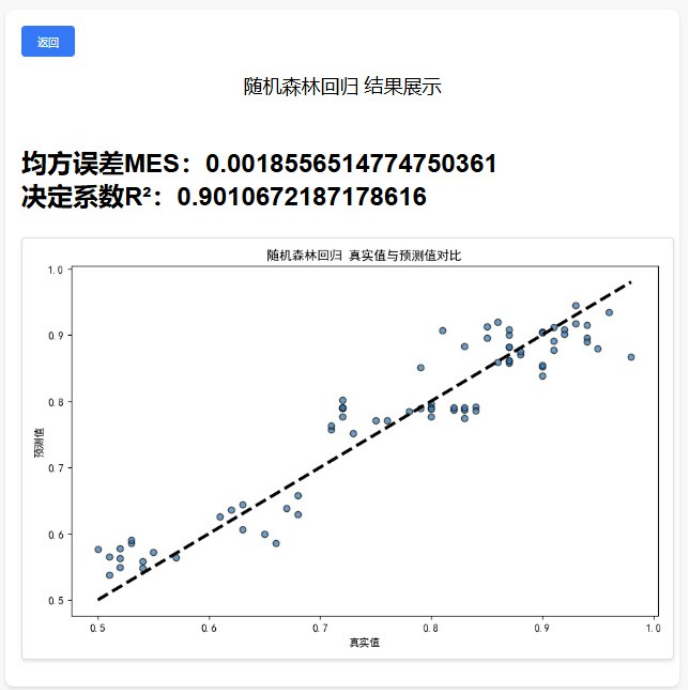

图 4.22

● 其他

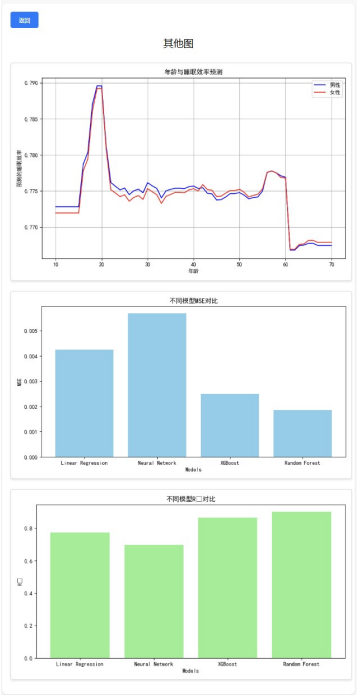

图 4.23

2、代码

```python
from flask import Flask, request, render_template, jsonify

from static.machine_learning.model_manage import predict_sleep_efficiency
from static.machine_learning.linear_torch import linear_regression
from static.machine_learning.main import load_data
from static.machine_learning.model_manage import model_save, model_load
from static.machine_learning.neural_network import neural_network_regression
from static.machine_learning.random_forest import random_forest_regression
from static.machine_learning.xgb_re import xgboost_regression

app = Flask(__name__)


file_path = 'static/machine_learning/Sleep_Efficiency.csv'
X_train, X_test, y_train, y_test, scaler, default_values = load_data(file_path)

# print("多元线性回归")

# print("神经网络回归")

# print("XGBoost回归")

# print("随机森林回归")

# 加载训练好的模型和标准化器
model, scaler, default_values = model_load('static/machine_learning/model/mf_model.pkl')


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def predict():
    try:
        age = int(request.form['age'])
        gender = int(request.form['gender'])

        # 预测
        prediction = predict_sleep_efficiency(age, gender, model, scaler, default_values)

        return jsonify({'prediction': prediction})
    except Exception as e:
        return jsonify({'error': str(e)})


@app.route('/linear-regression')
def linearre():
    linear_mse, linear_regression_r2 = linear_regression(X_train, X_test, y_train, y_test)
    return render_template('linear-regression.html', mse=linear_mse, r2=linear_regression_r2)


@app.route('/neural-network')
def neuraln():
    neural_mse, neural_r2 = neural_network_regression(X_train, X_test, y_train, y_test)
    return render_template('neural-network.html', mse=neural_mse, r2=neural_r2)


@app.route('/xgboost')
def xgbo():
    xgboost_mse, xgboost_r2 = model_xgb = xgboost_regression(X_train, X_test, y_train, y_test)
    return render_template('xgboost.html', mse=xgboost_mse, r2=xgboost_r2)


@app.route('/random-forest')
def rf():
    model_rf, rf_mse, rf_r2 = random_forest_regression(X_train, X_test, y_train, y_test)
    model_save(model_rf, scaler, default_values)
    return render_template('random-forest.html', mse=rf_mse, r2=rf_r2)


@app.route('/others')
def other():
    return render_template('others.html')


if __name__ == '__main__':
    app.run(debug=True)
```

图 4.24

这段代码定义了一个基于 Flask 的 Web 应用，用于处理和展示各种机器学习模型的预测结果。

1.导入模块

■ Flask、request、render_template 和 jsonify 用于创建和管理 Flask 应用。

■ 导入多个机器学习模型和工具函数。

2.初始化 Flask 应用

■ 创建一个 Flask 应用实例 app。

3.加载数据和模型

■ 从指定路径加载数据集，分割成训练集和测试集，并进行标准化处理。

■ 加载预训练的模型和标准化器。

4.定义路由和视图函数

■ **index**：渲染主页 index.html。

■ **predict**：处理 POST 请求，根据用户输入的年龄和性别使用预训练模型预测睡眠效率，并返回预测结果。

■ **linearre**：计算和显示线性回归模型的均方误差 (MSE) 和 R^2 分数。

■ **neuraln**:计算和显示神经网络模型的均方误差 (MSE) 和 R^2 分数。

■ **xgbo**：计算和显示 XGBoost 模型的均方误差 (MSE) 和 R^2 分数。

■ **rf**：计算和显示随机森林模型的均方误差 (MSE) 和 R^2 分数，并保存模型。

■ **other**：渲染 others.html 页面，展示预测图像和不同模型的对比。

4.8 测试输入

● 测试用例 1

年龄：空、性别：男性

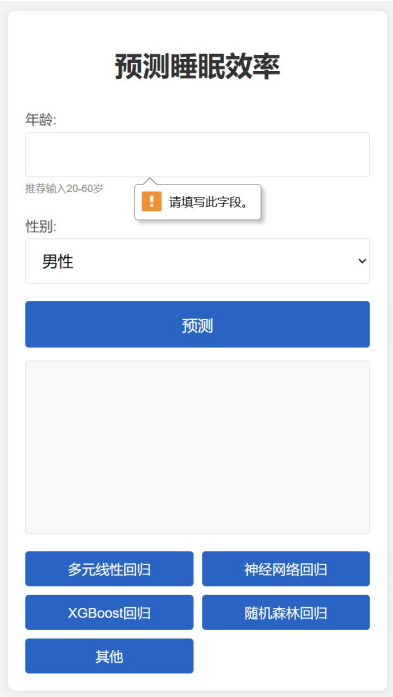

图 4.25

- 测试用例 2

年龄：10.5、性别：女性

图 4.26

- 测试用例 3
- 年龄：101、性别：男性

图 4.27

● 测试用例 4

年龄：35、性别：男性

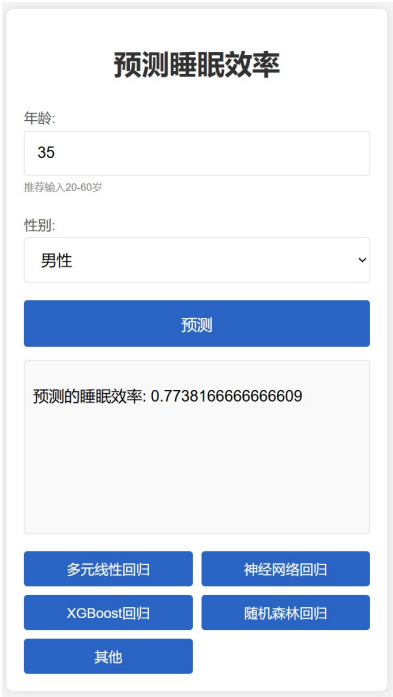

图 4.28

5、总结

本实验通过多元线性回归、神经网络回归、XGBoost 回归、随机森林回归等四种算法对睡眠效率进行了分析。

睡眠效率是总睡眠时间与卧床时间之比。睡眠效率与年龄密切相关,一般儿童和青少年睡眠效率较高。对比了四种模型评判结果，选择随机森林作为整个系统的预测模型。用户可以在前端输入年龄和性别，得到睡眠效率。经过这个值是在同龄人中的均值，但是对生活还是有很好的启发。在日常生活中注重自己的睡眠，也可以让身体更健康。本次实验的不足之处是分析还不够到位，需要更多的数据或是个人特征进行预测和判定。同时睡眠的记录还有可能与其他原因有关，比如天气或者其他事件。

对此，我们也尝试将不同性别 10~70 岁的年龄和睡眠效率预测做成一张图，猜测 20 岁前后睡觉效率达到最高，20-30 岁的人可能有工作压力或者熬夜玩耍，30-40 岁的人开始注意身体健康去运动，50 岁之后部分人群可能生病……

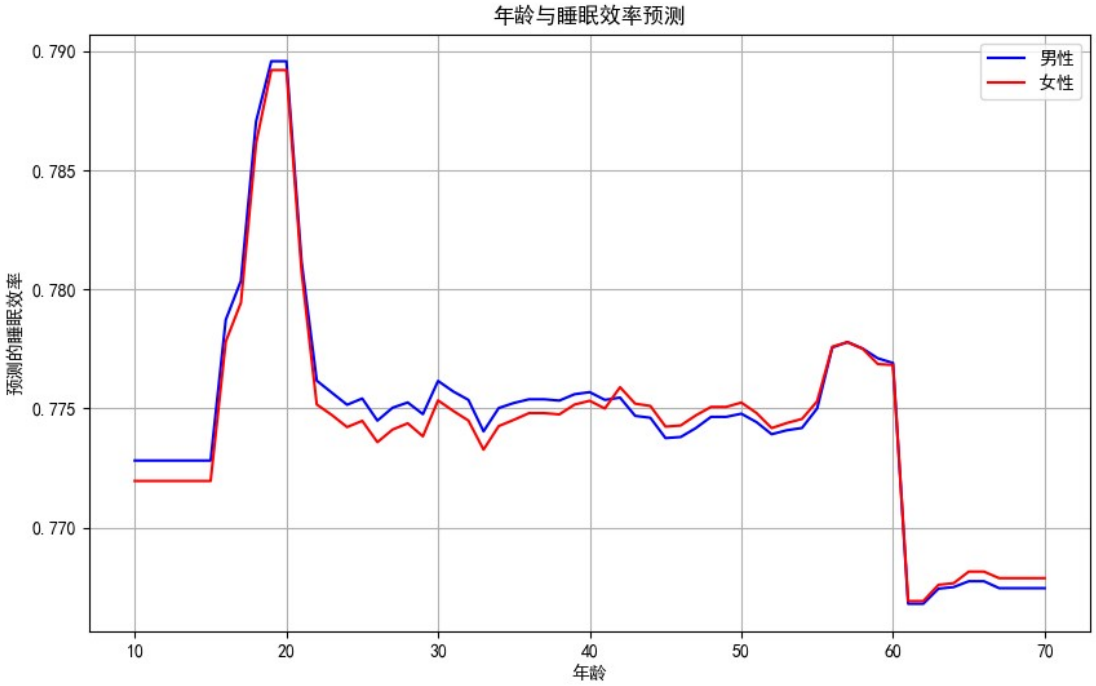

图 5.1

总的来说，这次的实验还是有很大的收获，小组成员也有从不同角度学习到了机器学习的方方面面。机器学习也是在当下比较热门，不仅可以为我们的职业规划做准备，也为后续的学习打下了基础。同时机器学习工程化仍然面临着很多挑战和问题，如模型的可解释性、数据隐私保护等。因此，未来的机器学习工程

化仍需要不断地探索和研究。