arXiv:2602.01078v1 [cs.AI] 1 Feb 2026

AutoHealth：一种用于自主健康数据建模的不确定性感知多智能体系统

Tong Xia [*†1] **Weibin Li** [*2] **Gang Liu** [2] **Yong Li** [†2]

摘要

基于大语言模型的智能体展现出自主机器学习的巨大潜力，但其在健康数据领域的适用性仍然有限。现有系统往往难以推广至不同类型的异构健康数据模态，严重依赖预定义的解决方案模板，无法充分适应特定任务目标，并且大多忽视了不确定性估计，而这对于医疗保健中的可靠决策至关重要。为应对这些挑战，我们提出了*AutoHealth*，这是一种新颖的不确定性感知多智能体系统，能够自主地对健康数据建模并评估模型可靠性。*AutoHealth* 通过五个专用智能体之间的闭环协作，执行数据探索、任务条件化模型构建、训练和优化，同时兼顾预测性能与不确定性量化。除了生成可直接使用的模型外，该系统还会生成详尽的报告，以支持可信的解读和风险感知决策。为严格评估其有效性，我们构建了一个具有挑战性的真实世界基准，包含涵盖多种数据模态和学习设置的 17 项任务。*AutoHealth* 完成了所有任务，预测性能较最先进的基线方法高出 29.2%，不确定性估计性能高出 50.2%。

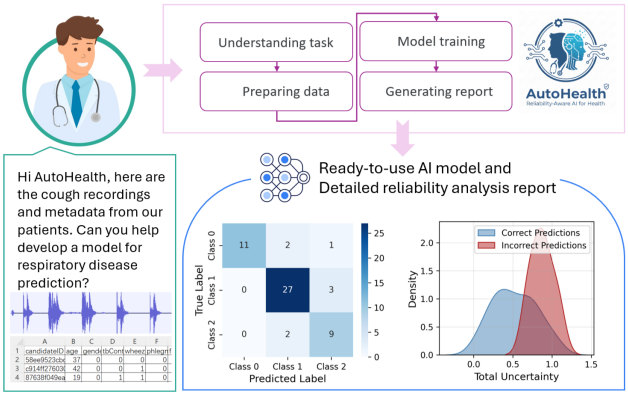


图*1. AutoHealth* 概览。一个能够从原始健康数据中自主构建预测模型并生成可靠性感知报告的多智能体系统。

1. 引言

人工智能驱动的健康预测在疾病风险评估、个性化医疗保健和公共卫生管理中发挥着关键作用（Purba等人，2019；Rajpurkar等人，2022）。然而，构建可靠的预测模型通常需要健康领域专家（例如临床医生）与人工智能专家（例如数据科学家）密切合作，以分析异构数据、设计特定任务模型并解释结果。在实践中，此类合作往往因数据共享限制以及协调跨学科专业知识的困难而受阻，从而大幅限制了研究效率与可扩展性。

大语言模型（LLM）和基于智能体的人工智能系统近期取得了长足进展，尤其是自动化数据科学代理（Chen 等人，2025；Hong 等人，2025；Jing 等人，2024），它们通过整合推理、编程和实验，开始重塑科学工作流程。这些系统有望降低技术门槛、减少模型开发中的人工投入，使其成为自主健康数据建模的一个有前景的方向。

尽管具有这种潜力，现有智能体系统应用于健康数据时仍显不足。诸如*Claude Code* 这类通用系统需要精心设计提示词，并要求使用者具备扎实的技术专业知识，因此健康领域专家难以直接使用；而诸如*DS-Agent* (Guo 等人，2024)这类端到端数据科学原型在医疗保健场景中存在根本性局限（见第2节）。首先，这些系统对多来源、多模态以及不适用于机器学习的健康数据支持有限，难以捕捉真实医疗保健环境固有的异质性和不规则性。其次，它们往往依赖于

[*]贡献相同 [1]清华大学万科公共卫生与健康学院 [2]清华大学电子工程系、北京信息科学与技术国家研究中心，中国。通讯作者：Tong Xia <tongxia@tsinghua.edu.cn>，Yong Li <liy- ong07@tsinghua.edu.cn>。

预印本。*2026*年*2*月*3*日。

预定义流水线或模板化解决方案，缺乏任务条件化的推理机制，因而无法灵活地根据特定任务目标和数据特征调整建模策略。第三，现有代理主要优化预测准确率，而可解释性和不确定性量化等与可靠性相关的方面很少被视为首要目标，这限制了它们在高风险临床决策和公共卫生决策中的适用性。

为应对这些挑战，我们提出了 *AutoHealth*，这是一种用于自主健康数据建模的不确定性感知多智能体系统。如图1所示，*AutoHealth*以高层次建模目标和原始健康数据为输入，自主执行任务理解、数据准备、模型构建和评估。它既能生成可部署的预测模型，也能生成结构化实验报告，从而以最少的人工干预，实现可靠且可复现的健康数据建模。

在方法上，*AutoHealth* 提出了一种闭环多智能体学习范式，明确针对现有智能体系统的关键局限。为处理复杂且异构的健康数据，专门的数据探索智能体通过代码驱动的分析与转换，支持模型在多种模态上稳健地开展后续学习。为兼顾知识复用与任务特定的适应性，中心协调智能体在自适应反馈循环中统筹模型设计与实现，使建模策略能够根据任务目标和从数据中获得的洞见进行迭代优化，而非僵化地复用模板。重要的是，为确保模型可靠性不止于预测准确率，*AutoHealth* 将不确定性量化视为同等重要的优化与评估目标，并系统记录所有实验决策和结果，将其汇总为结构化、面向应用的分析报告，以支持可信且风险审慎的使用。

为了在异构数据条件和可靠性约束下严格评估自主健康数据建模，我们构建了一个具有挑战性的真实世界基准，包含17项健康预测任务，涵盖六种数据模态（表格数据、图像、时间序列、自由文本、音频和图）以及六种学习设置（分类、回归、分割、生存分析、预测和链接预测）。在该基准上的大量实验表明，*AutoHealth* 持续优于强基线，实现了100%的任务成功率、预测性能提升29.2%，以及不确定性量化提升50.2%。

我们的贡献总结如下：

- 我们提出*AutoHealth*，一个闭环、具备不确定性感知的代理系统，可从原始健康数据自主构建预测模型与可靠性感知的分析报告。
- 我们引入一种多智能体协作范式，使面向异构健康数据的任务条件推理与自适应模型构建成为可能，并超越基于模板的自动化流水线。
- •我们构建了一个具有挑战性的真实世界健康预测基准，并进行了广泛评估，结果表明*AutoHealth* 相较强基线取得了持续且显著的改进。基准测试和代码可从 https://anonymous.4open.science/r/AutoHealth-46E0获取。

2. 相关工作

2.1. 健康领域的人工智能

人工智能技术已被广泛应用于健康相关预测任务，包括疾病风险评估、诊断支持、治疗结果预测和疫情暴发预测（Rajpurkar等人，2022；Miotto等人，2016；Singh和Mittal，2023）。这些方法通常利用电子健康记录、医学影像、临床笔记和纵向时间序列数据等异构数据来源来建模健康结局。尽管取得了成功，重大挑战依然存在，因为健康数据往往具有异构性、高维性和不完整性，并且在不同机构和人群之间存在显著差异（Shickel等人，2018）。更根本的是，大多数现有方法仍然是任务特定且数据依赖的，依赖于专家设计的架构、手工设计特征、模态特定预处理以及领域知识驱动的假设。这种对人工定制的高度依赖限制了其在不同数据集、模态和临床任务之间的可迁移性与泛化能力，从而促使人们需要更具适应性和自动化的建模框架，以支持现实场景中可扩展的健康数据分析。

2.2. 可靠性感知机器学习

在医疗保健等高风险领域，模型不仅要具备出色的预测性能，还必须在部署于决策支持场景时提供校准的不确定性估计（Kompa 等，2021；Ghandeharioun 等，2022；Xia 等，2024）。不确定性信息有助于识别模糊或分布外输入，实现选择性预测，并支持更安全的人机协作。以往研究探索了多种人工智能不确定性量化技术，包括贝叶斯神经网络（Gal & Ghahramani，2016）、深度集成（Lakshminarayanan 等，2017）和测试时数据增强（Wang 等，2019）。然而，许多健康预测模型仍缺少内置的不确定性估计机制，而且评估方案往往主要关注准确率，却忽略了校准、覆盖

*Ta*表 *1.* 将*AutoHealth*与具有代表性的自动化及基于大语言模型的数据科学框架进行比较 s.

框架	主要功能						
	多智能体	多种模态	数据探索	结合检索	任务反馈	不确定性	报告
AutoML-GPT (Zhang等人，2023)	✗	✓	✗	✗	✓	✗	✗
HuggingGPT（Shen等人，2023）	✗	✓	✗	✓	✗	✗	✓
Prompt2Model（Viswanathan 等人，2023）	✗	✗	✗	✓	✗	✗	✗
DS-Agent（Guo 等人，2024）	✓	✓	✗	✓	✗	✗	✗
Data-Interpreter（Hong 等人，2025）	✓	✓	✓	✗	✓	✗	✓
AutoML-Agent（Trirat 等人，2025）	✓	✓	✗	✓	✓	✗	✗
AutoMMLab（Yang 等人，2025）	✓	✗	✓	✗	✗	✗	✓
AutoKaggle（Li 等人，2024）	✓	✗	✓	✓	✓	✗	✓
OpenLens AI（Cheng 和 Suo，2025）	✓	✓	✓	✓	✗	✗	✓
***AutoHealth*（我们的方法）**	✓	✓	✓	✓	✓	✓	✓

率或弃权行为（Carvalho 等，2023）。总体而言，以往研究强调了不确定性量化对于可靠的基于人工智能的健康预测的重要性，尽管在模型开发和评估中，这一点仍常常未受到足够重视。

2.3. 基于大语言模型的数据科学智能体

近期的大语言模型（如*Claude Sonnet 4* 和 *ChatGPT-5*）在代码生成和程序辅助方面展现出强大的能力，推动了*Claude Code*等专用编程智能体的出现。虽然这些工具能够显著加快软件开发，但将其应用于健康数据建模通常需要仔细的提示工程、环境配置和方法论专业知识，这限制了非技术类健康研究人员使用这些工具。

与此同时，自动化数据科学已成为一个活跃的研究方向，早期工作主要聚焦于单个基于大语言模型的智能体，将自然语言指令转换为可执行的建模流程。具有代表性的系统包括*AutoML-GPT* （Zhang等人，2023）、*HuggingGPT* （Shen等人，2023）和*Prompt2Model* （Viswanathan 等人，2023），它们建立了整合数据预处理、模型选择和评估的端到端工作流程。尽管这些方法在概念上意义重大，但它们本质上受限于单个大语言模型有限的上下文窗口和推理能力，因此难以处理复杂的真实世界健康数据和建模目标。

为解决这些局限，近期研究转向了多智能体大语言模型系统，由各司其职的代理协同合作，将复杂任务分解并加以解决。*DS-Agent* (Guo 等人，2024) 引入基于案例的检索，以复用成功的建模经验，从而提升稳健性和任务完成率；而 *Data-Interpreter* (Hong 等人，2025) 则强调轨迹级反馈，根据中间执行结果进行迭代改进。*AutoML-Agent* (Trirat 等人，2025) 在统一框架中整合了检索与反馈，但主要面向基准测试型数据集和传统优化目标，对异构健康数据的支持有限，而且很少纳入显式的不确定性量化。

与通用自动化框架并行，若干系统面向特定领域场景。*AutoMM-Lab* （Yang 等人，2025）专注于计算机视觉流程，而*AutoKaggle* （Li 等人，2024）针对竞赛风格的数据科学工作流进行定制。尽管它们在各自领域内行之有效，但其任务假设与流水线设计限制了向更广泛的机器学习及医疗保健研究场景的泛化。*OpenLens AI* （Cheng 和 Suo，2025）与健康相关应用最为契合，强调探索性健康信息学分析与可视化报告。然而，它对系统性机器学习实验设计的支持有限，且未明确纳入可靠性感知建模或不确定性量化。

我们将代表性先前工作汇总于表格1。总体而言，现有系统通常仅处理自动化建模的孤立方面，例如流水线复用或反馈优化，但很少在统一的智能体框架内整合异构数据探索、任务条件化模型构建与不确定性感知评估。

3. AutoHealth

本节介绍所提出的多智能体系统*AutoHealth*及其迭代工作流，如图2所示。

3.1. 智能体规范

下面简要介绍该框架中的各个代理（详细定义见附录D）。

元智能体 ($\mathcal{A}_m$) 充当中央协调者，并作为与用户交互的主要接口。它将健康数据建模目标转化为全局计划，协调专用智能体之间的交互，并管理自适应闭环反馈过程，以迭代优化建模决策并跟踪整体进度。

数据智能体($\mathcal{A}_d$) 负责健康数据探索。

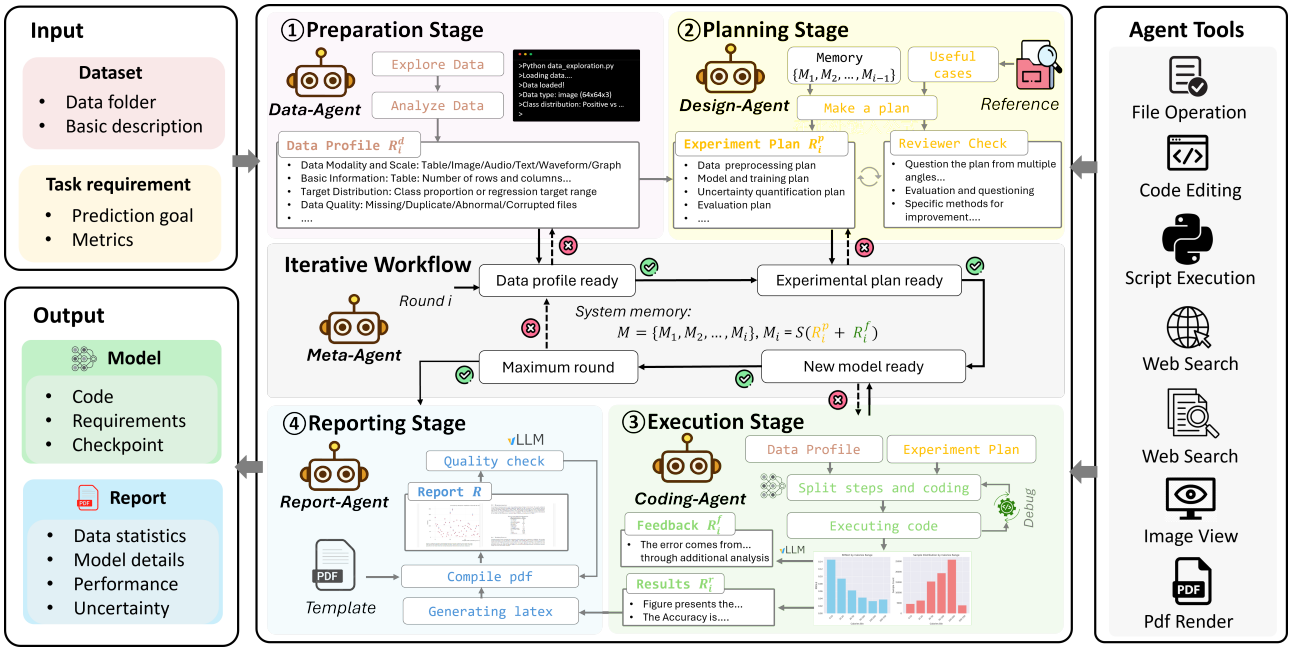


图 2。*AutoHealth* 框架。该系统采用由元智能体协调的闭环多智能体架构，组织涵盖准备、规划、执行和报告的迭代工作流。专用智能体在各个阶段运行，以实现模块化、可靠且可复现的健康数据建模。

它处理多来源的异构输入，执行数据质量评估，并生成结构化数据档案，以确保下游阶段基于具有临床意义且充分考虑数据特征的表示开展工作。

设计智能体 ($\mathcal{A}_p$)旨在根据特定任务和不确定性感知约束生成自适应实验计划。在建模目标和数据特征的指导下，该智能体提出模型架构、特征配置和训练策略，突破僵化的基于模板的复用方式。

Coding-Agent($\mathcal{A}_c$)将高级设计规范转化为可执行且可复现的机器学习代码。它实现数据管道、模型架构、训练流程和评估例程，同时支持闭环优化过程中的迭代更新。

报告智能体($\mathcal{A}_r$)将实验决策、结果和可靠性分析整合为结构化报告。通过以可解释的方式总结模型行为、不确定性和局限性，该智能体支持对生成模型进行透明、可信且可复现的使用。

3.2. 总体迭代工作流

在元智能体 $\mathcal{A}_m$ 的协调下，*AutoHealth*运行一个迭代式闭环工作流程，逐步完善建模方案。每次迭代 $i < N$ 最多可能包含三个阶段：准备、规划和执行；系统会根据当前任务状态和中间结果选择性地激活这些阶段，而最终报告阶段仅在最后一次迭代 N 后触发。

总体而言，准备阶段侧重于理解数据，规划阶段将经过验证的数据洞察转化为以任务为条件的实验设计，而执行阶段则实施并评估所提出的设计，同时生成用于改进的反馈。

- **准备阶段。** 数据智能体执行系统化的数据探索和质量评估，生成结构化的数据概况 R_i^d，总结数据模态、分布、时间结构和潜在质量问题。

- **规划阶段。** 设计智能体根据数据概况 R_i^d和任务规范，生成实验计划 R_i^p，其中包括模型架构、训练策略和评估方案。对于 $i > 1$，还会结合先前轮次 $\{M_1, \ldots, M_{i-1}\}$的见解，以支持自适应优化。

- **执行阶段。** Coding-Agent 根据 R_i^d和 R_i^p 实施实验计划，并生成训练后的模型以及实验记录 R_i^r。系统根据模型行为和性能生成结构化反馈 R_i^f，以指导后续迭代。

最终报告阶段。 完成最后一轮迭代 N后，报告智能体将 R_N^r 整合为一份兼顾可靠性、面向应用的报告 R，作为系统的最终输出。

表格 2 总结了各阶段的输入和输出，以下小节将详细介绍其设计。

表格 2. 迭代工作流第 i 轮各阶段输入与输出汇总。

阶段	主智能体	输入	输出
准备	数据智能体 $\mathcal{A}_d$	原始数据、任务规范	数据概况 R_i^d
规划	设计智能体 $\mathcal{A}_p$	数据概况 R_i^d、任务规范、记忆 $\{M_1, ..., M_{i-1}\}$	实验计划 R_i^p
执行	Coding-Agent $\mathcal{A}_c$	原始数据、数据概况 R_i^d、实验计划 R_i^p	实验结果 R_i^r、实验反馈 R_i^f
报告	报告智能体 $\mathcal{A}_r$	实验结果 R_i^r（仅当 $i = N$时）	可读性优化报告 R

3.3. 准备阶段

给定原始健康数据和任务规范，数据代理 $\mathcal{A}_d$ 生成结构化的数据概况 R_i^d，以支持下游建模。为实现有效且系统的数据理解，我们采用了三项设计原则。

代码驱动的数据理解。 与依赖用户提供的数据集描述不同， $\mathcal{A}_d$ 通过可执行代码直接与原始数据交互，包括检查文件、数据加载和程序化分析。这种设计只需要健康领域专家提供最少的输入（例如文件路径和简要摘要），即可详细理解数据结构和质量。

跨数据类型的结构化目标。 针对七种常见医疗数据类型定义了标准化探索目标，包括表格数据、时间序列、图像、图像分割、音频、自由文本和图数据。对于每种类型，这些目标涵盖基本属性、数据质量指标以及更深层次的特征，例如特征与目标的关系或轻量级特征筛选。

面向下游建模的数据指导。基于探索结果， $\mathcal{A}_d$ 生成数据加载的参考代码，并提供预处理和建模建议（例如，处理类别不平衡或选择加权损失），从而将数据特征明确关联到后续建模决策。

3.4. 规划阶段

给定数据概况 R_i^d、任务规范以及来自先前轮次的系统记忆 $\{M_1, \ldots, M_{i-1}\}$，设计智能体 $\mathcal{A}_p$ 生成实验计划 R_i^p 。为提高可执行性与任务适配性，采用了三项设计原则。

检索增强规划。 为避免仅依赖模型先验知识， $\mathcal{A}_p$ 配备了本地案例检索和网络搜索工具，并遵循先前工作，例如*DS-Agent* （Guo等人，2024）。检索到的方案作为参考，使规划决策立足于经过验证的实践。

反馈感知定制。 *AutoHealth*维护着持久记忆 $\{M_1, \ldots, M_{i-1}\}$，在多轮迭代中积累执行反馈和设计效果。在规划过程中，系统会查询该记忆，以评估重用解决方案在当前数据和任务条件下的适用性。当历史证据表明存在不匹配或性能下降时， R_i^p 会被选择性地修订，从而实现超越模板重用的跨轮次累积优化。

迭代式计划评审与改进。 生成初始计划后，设计智能体 $\mathcal{A}_p$ 会调用辅助的基于大语言模型的评审工具，以评估其可行性和潜在风险。随后利用评审意见迭代改进R_i^p ，再进行执行。

3.5. 执行阶段

给定数据概况 R_i^d和实验计划 R_i^p 后，Coding-Agent $\mathcal{A}_c$ 会实现流水线，并生成实验记录 R_i^r 以及结构化反馈R_i^f 。为提高稳健性、可调试性和分析深度，我们采用三项设计原则。

逐步计划分解与执行。 Coding-Agent $\mathcal{A}_c$ 不会生成一个庞大的程序，而是将 R_i^p 分解为细粒度步骤，依次实现并验证。这种逐步执行方式可以减少错误传播、改进故障定位，并提高复杂实验流水线的稳定性。

全面性能与不确定性分析。除单一目标指标外， $\mathcal{A}_c$ 还会系统评估预测性能、校准、不确定性行为以及潜在失效模式。关键统计数据和可视化诊断结果（例如学习曲线和不确定性图）会记录在 R_i^r 中，以支持事后分析并确保可复现性。

VLLM辅助反思分析。 为增强结果解释，视觉语言模型（VLLM）由 $\mathcal{A}_c$ 调用作为辅助分析工具，而非作为独立代理。通过联合检查数值结果、可视化诊断和执行日志，该工具有助于识别数据泄漏、过拟合或目标不匹配等问题。由此产生的反馈被汇总至 R_i^f 。

每轮执行结束后 i，元智能体 $\mathcal{A}_m$ 会将实验计划 R_i^p 和执行反馈R_i^f 总结为一个精简的记忆单元 M_i。形式上，系统记忆更新为 $\mathcal{M} = \{M_1, \ldots, M_{i-1}, M_i\}$，其中$M_i = S(R_i^p, R_i^f)$，而 $S(\cdot)$ 表示一种总结操作，用于提取重要的设计决策、性能结果、失败案例和可操作的见解。此**更新后的记忆** $\mathcal{M}$ 会延续至后续规划轮次，从而实现建模策略的累积式跨轮次改进。

3.6. 报告阶段

最终迭代结束后 N，报告智能体 $\mathcal{A}_r$ 会将执行结果 R_N^r 总结为一份结构化、面向应用的报告 R_N，以支持可靠的解读和实际使用。本文采用了三项设计原则。

模式驱动报告。 报告按照预定义的模式生成，并分为三个章节：数据分析、模型性能和不确定性评估。这种标准化结构确保对建模决策、结果和局限性进行全面且可复现的记录。

面向应用的解读。 报告并非采用学术论述的方式，而是着重提供操作指导。报告以清晰、以指令为导向的语言解释模型性能和不确定性行为，并明确建议何时可以信任预测、何时应优先进行人工审核。

视觉质量审查。 使用基于 VLLM 的审核器作为辅助工具，检查图、表格及其配套文本的清晰度和一致性。根据审核反馈对报告进行完善，然后定稿。

4. 结果

我们通过将*AutoHealth*与异构健康数据集上的最先进方法进行比较，评估其有效性。结果如下所述。

4.1. 设置

基准测试。 表格 3总结了我们整理的基准测试，其中包含 17 项任务。这些数据集来自异构来源，并采用针对具体任务的评估指标，例如分类任务的准确率和回归任务的均方根对数误差（RMSLE）。关于数据集和评估指标的更多详情见附录A.1。与先前研究（Guo 等人，2024；Trirat 等人，2025）相比，我们的基准测试涵盖了更广泛的任务类型，对系统泛化能力提出了更严格的考验。

评估指标。 遵循先前工作（Trirat 等人，2025），我们使用成功率（*SR*）和 归一化性能分数（*NPS*）评估系统，并报告综合分数（*CS*），定义为 CS $= 0.5 \times$ SR $+0.5 \times$ NPS。

具体而言，SR通过两个组成部分衡量任务完整性：成功完成模型训练和不确定性量化。两者均完成时，得到SR $= 1$；仅完成其中一项时，得到SR$= 0.5$；两者均未完成时，得到SR$= 0$。NPS综合考量任务性能和不确定性行为，以评估模型的整体质量。对于基于准确率的

表格 *3.* 跨模态基准测试任务。

模态	任务类型	ID 名称	指标
表格数据	分类	1 吸烟者状态预测	准确率
		2 患者生存预测	准确率
		3 药物反应机制预测 对数损失	
	回归 4	卡路里估算	RMSLE
	生存分析 5	HCT后生存预测	C指数
图像	分类	6 皮肤癌识别	F1
		7 黑色素瘤检测	准确率
	分割	8 视网膜血管分割	Dice
时间序列	分类	9 基于心电图的心律失常检测	F1
		10 基于惯性测量单元的手势预测	平均F1
		11 基于脑电图的癫痫发作预测	KL散度
	预测	12 新冠肺炎感染病例预测 平均绝对误差	
自由文本	分类	13 自闭症谱系障碍预测	准确率
		14 COVID-19 情感分类	准确率
音频	分类	15 呼吸系统疾病预测	宏平均 F1
		16 构音障碍检测	准确率
图	链接预测	17 蛋白质链接属性预测	Hits@20

指标，直接使用原始分数。对于基于损失的指标（称为 s），将性能指标和不确定性指标均转换为 $S = 0.1/(0.1 + s)$，以确保尺度一致。随后，对转换后的性能得分和不确定性得分赋予相同权重，计算NPS。

基线。 我们将系统与三类基线进行比较。(1) 朴素大语言模型：向大语言模型提供相同的任务规范，并手动执行生成的代码以记录性能。(2)通用编程智能体：我们采用*Claude Code*，并使用具有可比性的提示词，以确保比较公平。用于 大语言模型和 *Claude Code*的提示词见附录 A.2。(3)高级数据科学智能体：我们评估了具有代表性的系统，包括*DS-Agent* 和 *AutoML-Agent*。我们还尝试纳入*OpenLens AI*；然而，其执行时间过长，所有任务均超出实际运行时限。因此，我们未在报告的指标中列出其量化结果。

实现细节。 除非另有说明，所有代理均使用 *DeepSeek-V3.2* 作为骨干模型。当需要 VLLM 时，我们使用*GLM-4.6V*。对于所有智能体系统，包括 *DS-Agent* 和*AutoML-Agent*，我们将最大迭代轮次数设为 $N = 5$。每种方法均独立运行三次，并报告最佳性能。所有实验均在配备四块 NVIDIA A100 GPU 的 Ubuntu 22.04 LTS 服务器上进行。为执行生成的模型，我们采用与*DS-Agent*相同的运行时环境，并通过提供的骨架脚本预先安装所有必需的库。

4.2. 主要结果

总体结果汇总于图3，详细指标见附录B表格7–15。

成功率。 图3 显示，只有 *ClaudeCode* 和我们提出的 *AutoHealth*能够成功完成所有任务，而其他方法则在模型构建或

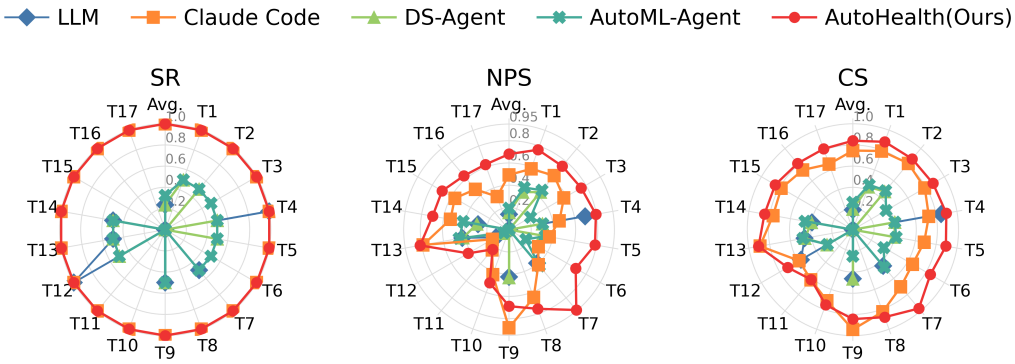
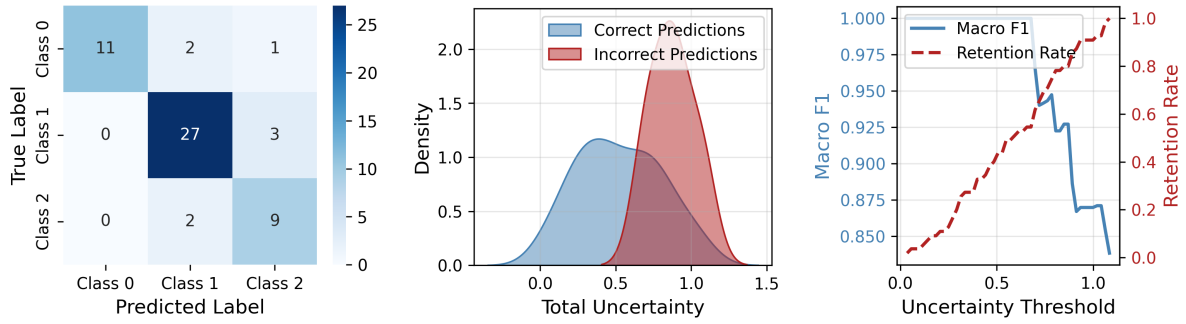


图3。 使用 *SR*、*NPS* 和 *CS* 进行性能比较。Avg 表示在全部 17 个任务上的平均得分 s.



(a) 混淆矩阵 *(b)* 不确定性分布 *(c)* 选择性预测

图4. 系统生成报告摘录 任务 15 呼吸系统疾病预测的系统生成报告。完整报告见附录 C .

不确定性量化阶段失败。朴素大语言模型基线只有在精心设计提示词并手动执行代码时，才能取得合理的性能。*DS-Agent* 和*AutoML-Agent* 在若干任务（例如任务 10、11 和 15–17 ）上失败，主要原因是其预定义流水线的模态泛化能力有限。因此，朴素大语言模型、*Claude Code*、*DS-Agent*、*AutoML-Agent*和*AutoHealth* 的平均成功率（SR）分别为 23.5%、**100%**、23.5%、32.3% 和**100%**。

性能得分。 *AutoHealth* 的平均 NPS 为 0.667，高于排名第二的 *Claude Code* （0.493），高出 **37.9%**。具体而言，其平均预测得分为 0.747（表格 10），不确定性得分为 0.613（表格 12），分别比 *Claude Code* 高**29.2%**、**50.2%**。此外，如图3所示，*AutoHealth* 在 17 项任务中的 16 项上取得了最先进的性能。

综合分数。 如图3所示，*AutoHealth*的平均综合得分最高，达到 0.840，优于*Claude Code*（0.747）、*AutoML-Agent* （0.258）、*DS-Agent*（0.195），以及朴素大语言模型（0.187），相对提升幅度为**12.4% 至 349.2%**。这些结果表明，*AutoHealth* 能够以极少的人工干预，为自动化健康数据建模提供可靠且有效的端到端解决方案。

4.3. 报告质量评估

为展示*AutoHealth*生成高质量可靠性感知报告的能力，我们基于任务 15 展示一个具有代表性的案例研究。该任务聚焦于呼吸系统疾病预测，旨在利用多模态输入区分健康、患病和慢性病状况。鉴于任务 15 具有多模态结构且类别严重不平衡，因此选作代表性示例。完整的自动生成报告见附录 C，部分图示见图 4。

该报告展示了 *AutoHealth* 如何在一个真实世界临床数据集上自主完成整个建模流程。该数据集整合了结构化表型变量（人口统计信息、症状和病史）与呼吸音频记录（咳嗽和元音）。在无需人工干预的情况下，*AutoHealth* 选择双分支后期融合神经网络架构，分别对表格特征和音频嵌入进行建模，随后在特征层面进行融合以实现联合预测。为提高稳健性并支持不确定性估计，系统进一步构建了由三个模型组成的深度集成。

在训练过程中，*AutoHealth* 会自动识别类别不平衡并应用针对性缓解策略（即，

表格 *4.* 成功率（SR）消融结果。

方法	AutoHealth	AutoHealth-S	AutoHealth-SD
成功率（%）	100	70.6	41.2

表格 *5.* 关于不确定性量化的消融结果。AVG 表示任务 1 至 5 的归一化不确定性分数的平均值，详见附录表格 16。

方法	Claude Code	AutoHealth	AutoHealth-T	AutoHealth-R
AVG	0.403	0.819	0.733	0.650

类别加权损失），从而实现强劲且均衡的性能，测试集宏平均 F1 得分为 0.838，且各类别的召回率相近（图 4a）。

此外，自动生成的报告还包括校准和不确定性分析，表明误分类样本始终伴随较高的预测不确定性（图 4b）。这有助于有效区分高置信度和低置信度预测，并支持兼顾风险的人在回路部署。如图 4c所示，拒绝预测最不确定的 20% 样本后，其余预测的宏平均 F1 提高至 0.928，但代价是对部分案例不作预测。

总体而言，*AutoHealth*不仅能生成准确的预测，还能通过明确识别失效模式并利用不确定性支持兼顾风险的临床决策，提供可执行的诊断见解和部署指导。

4.4. 其他分析

消融研究。 我们首先考察数据探索以及逐步计划分解与执行对成功完成所有任务的贡献。我们假设，这些设计组件是系统在异构数据模态和学习设置之间实现泛化的关键。我们比较完整 AutoHealth 系统与两个简化变体的成功率： *AutoHealth-S* （不进行逐步分解）以及 *AutoHealth-SD* （同时不进行逐步分解和数据探索）。结果汇总于表格 4。移除逐步分解已导致成功率显著下降，而进一步禁用数据探索则会使性能下降更加明显。这一趋势凸显了结构化规划和数据感知推理对*AutoHealth*的互补重要性。

随后，我们通过在规划阶段消融不确定性量化模板（ *AutoHealth-T*），以及在执行阶段消融VLLM辅助反思分析（*AutoHealth-R*），考察基于模板的解决方案复用与目标感知适应之间的权衡。我们在表格任务（T1–T5）上评估这些变体；各方法的预测性能大体相当，因此可以专注分析其不确定性估计行为。如表格 5所示，完整的 *AutoHealth* 取得了显著更低的不确定性误差，

表*6*。 部分任务的令牌数和时间成本。输入、输出和缓存令牌数以百万（M）为单位报告。

Task	数据规模	输入	输出	缓存	成本（人民币）	时间（分钟）
T1	11.03MB	1.58	0.20	0.93	2.1	119.8
T5	9.87MB	2.65	0.38	1.49	3.8	246.1
T6	143.5MB	3.49	0.55	1.76	5.5	478.7
T9	7.75MB	1.81	0.22	1.14	2.2	123.2
T11	6.6GB	3.73	0.47	1.83	5.6	634.1
T15	1.9GB	3.03	0.38	1.69	4.2	187.5
T17	102MB	3.38	0.47	1.77	5.0	419.5

而移除任一组件都会导致性能持续下降。具体而言，与 *AutoHealth*相比，*AutoHealth-T* 的归一化不确定性分数降低了10.5%，而*AutoHealth-R*降低了20.6%。这些结果表明，可靠的不确定性估计既需要可复用模板，也需要自适应的、由反馈驱动的推理，而不能仅依赖模板复用。

成本与效率分析。 表格 6 展示了部分任务的令牌使用量、货币成本和实际运行时间（完整结果见附录表格 17，各阶段的执行时间占比汇总于附录图5）。在 17 项任务中，令牌总消耗保持在适中范围内，货币成本始终较低，通常每项任务低于 6 元人民币。值得注意的是，*AutoHealth* 展现出快速的迭代周期，即使对于涉及吉字节级数据的任务，大多数任务也能在 2–6 小时内完成（5 次运行）。这主要得益于高效且结构化的多智能体协作，大幅减少了迭代过程中的冗余令牌消耗。通过将任务理解、模型构建、执行和反馈整合到统一的端到端智能体工作流中，*AutoHealth* 最大限度地减少了人工干预和迭代开销。总之，这些结果表明，*AutoHealth* 能够以快速且经济的方式开展端到端实验，从而降低大规模实证研究的门槛，并加速真实世界健康数据建模领域的科学发现。

5. 结论

在本研究中，我们提出了*AutoHealth*，一种新颖的不确定性感知闭环多智能体框架，用于自主健康数据建模。通过协调负责数据探索、任务条件化模型构建、优化和报告的专用智能体，*AutoHealth* 解决了现有基于大语言模型的数据科学智能体在处理异构健康数据及可靠性要求极高的任务时面临的关键局限。在一项具有挑战性的真实世界基准测试上进行的大量实验表明，*AutoHealth* 在多种模态和学习设置下始终优于强大的通用基线，同时提供不确定性感知的输出，以支持可信的解读。总体而言，本研究推动自主数据科学迈向更可靠、更实用的部署，使其能够应用于医疗保健等安全关键领域。

软件与数据

我们的代码已提交至匿名 GitHub。此研究使用的所有数据均已公开。

影响声明

本文提出一项方法学贡献，旨在推进面向健康数据建模的自主数据科学与机器学习系统。所提出的框架旨在提升预测建模中的可靠性、透明度与不确定性意识，这些被广泛视为医疗保健应用中的重要考量。尽管此类系统可能支持更高效的模型开发与分析，但它们并非旨在取代临床专业知识或决策。与其他应用于健康数据的机器学习方法一样，适当的数据治理、验证与人工监督对于确保负责任的使用仍然至关重要。除医疗保健领域机器学习研究通常伴随的影响外，我们预计不会产生任何直接的负面社会影响。

参考文献

Carvalho, V. F.、Chmiel, F. P.、Johnson, A. E. W. 与 Ghassemi, M. 面向临床风险预测的机器学习：校准与不确定性量化方法综述。美国医学信息学协会杂志，2023。

Chen, K., Wang, P., Yu, Y., Zhan, X. 和 Wang, H. 基于大语言模型的数据科学智能体：综述。*arXiv* 预印本 *arXiv:2508.02744*，2025。

Cheng, Y. 和 Suo, J. OpenLens AI：面向健康信息学的完全自主研究智能体。*arXiv* 预印本 *arXiv:2509.14778*，2025。

Gal, Y. 和 Ghahramani, Z. 随机失活作为贝叶斯近似：在深度学习中表示模型不确定性。载于国际机器学习会议（*ICML*），第1050–1059页，2016。

Ghandeharioun, A., Yuan, A. 和 Ghassemi, M. 面向医学图像分类的分布感知校准。载于健康、推理与学习会议（*CHIL*）论文集，第248–260页，2022。

Guo, S., Deng, C., Wen, Y., Chen, H., Chang, Y., and Wang, J. Ds-agent：通过赋能大语言模型进行基于案例的推理，实现自动化数据科学。载于第*41*届国际机器学习会议论文集，第16813–16848页，2024。

Hong, S., Lin, Y., Liu, B., Liu, B., Wu, B., Zhang, C., Li, D., 等。数据解释器：面向数据科学的大语言模型代理。收录于 *Findi*$^{ngs\ o}$ *f the Association for Computational Linguisti* 计算科学（*ACL*），第19796–19821页。计算语言学协会，2025。Jing, L., Huang, Z., Wang, X., Yao, W., Yu, W., Ma, K., Zhang, H., Du, X., and Yu, D. Dsbench：数据科学智能体距离成为数据科学专家还有多远？*arXiv*预印本 *arXiv:2409.07703*，2024。Kompa, B., Snoek, J., and Beam, A. L. 需要第二意见：在医学机器学习中传达不确定性。*NPJDigital Medicine*，4(1):1–6，2021。Lakshminarayanan, B., Pritzel, A., and Blundell, C. 使用深度集成进行简单且可扩展的预测不确定性估计。载于神经信息处理系统进展（*NeurIPS*），第6402–6413页，2017。

Li, X., Wang, Z., Zhang, R., 等。AutoKaggle：利用基于大语言模型的多智能体自动化机器学习系统，让数据科学大众化。*arXiv* 预印本 *arXiv:2402.00854*，2024。

Miotto, R., Li, F., Kidd, B. A. 和 Dudley, J. T. Deep Patient：一种无监督表示，可根据电子健康记录预测患者的未来。《科学报告》，6:26094，2016。

Purba, J. H. V., Ratodi, M., Mulyana, M., Wahyoedi, S., Andriana, R., Shankar, K. 和 Nguyen, P. T. 医学与医疗保健中的预测模型。医学与医疗保健中的预测模型，8(6S3)，2019。

Rajpurkar, P., Chen, E., Banerjee, O. 和 Topol, E. 人工智能在健康与医学中的应用。自然医学，28(1):31–38，2022。

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt：使用 ChatGPT 和 Hugging Face 中的好友解决人工智能任务。载于神经信息处理系统进展（*NeurIPS*），第 36 卷，第 38154–38180 页，2023。

Shickel, B., Tighe, P. J., Bihorac, A., and Rashidi, P. Deep ehr：电子健康记录（ehr）分析深度学习技术最新进展综述。*IEEE*生物医学与健康信息学期刊（*JBHI*），22（5）：1589–1604，2018。

Singh, S. and Mittal, S. 使用基于优化的机器学习模型进行大流行疫情暴发预测。载于*2023*年第*3*届计算、通信、嵌入式与安全系统进展国际会议（*ACCESS*）。IEEE，2023。

Trirat, P., Jeong, W. 和 Hwang, S. J. AutoML-Agent：用于全流程自动机器学习的多智能体大语言模型框架。载于第 *42* 届国际机器学习会议（*ICML*）论文集，2025。海报展示。

Viswanathan, V., Zhao, C., Bertsch, A., Wu, T. 和 Neu- big, G. Prompt2Model：根据自然语言指令生成可部署模型。载于 Feng, Y. 和 Lefever, E.（编），*2023*年自然语言处理实证方法会议论文集（*EMNLP*），第 413–421 页，新加坡，2023。计算语言学协会。doi: 10.18653/v1/2023.emnlp-demo.38。网址：https://aclanthology.org/2023.emnlp-demo.38/。

Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., and Vercauteren, T. 基于测试时增强的卷积神经网络医学图像分割偶然不确定性估计。载于医学图像计算与计算机辅助干预（*MICCAI*），第103–111页。施普林格，2019。

Xia, T., Dang, T., Han, J., Qendro, L., and Mascolo, C. 基于类别均衡证据深度学习的不确定性感知健康诊断。*IEEE*生物医学与健康信息学期刊（*JBHI*），28(11):6417–6428，2024。

Yang, Z., Zeng, W., Jin, S., Qian, C., Luo, P., and Liu, W. AutoMMLab：根据语言指令为计算机视觉任务自动生成可部署模型。载于*AAAI*人工智能会议论文集（*AAAI*），第22056–22064页，2025。

Zhang, S., Gong, C., Wu, L., Liu, X., and Zhou, M. Automl-gpt：基于 GPT 的自动机器学习。*arXiv* 预印本 *arXiv: 2305.02499*，2023。

附录

本文附录的组织结构如下：

- 附录 A 提供了关于基准测试的更多细节。
- 附录 B 展示了更详细的结果。
- 附录 C 展示了任务 15 的生成报告。
- 附录 D 说明了专用智能体在 *AutoHealth* 中的设计。

A. 基准测试与实验的补充介绍

A.1. 基准测试

基于表格数据的任务包括：

- **T1 吸烟者状态预测**。 此任务基于一个结构化健康行为数据集，其中每一行代表一个人的生理信号和生活方式属性。目标是一个用于表示吸烟状态的二元标签。任务目标是学习生物标志物中的模式，例如肺活量、胆固醇水平、血压或其他身体信号指标，从而可靠地区分吸烟者与非吸烟者。对于机器学习而言，该数据集提供了探索特征重要性、揭示潜在健康相关性、评估模型在不同子群体间的公平性，以及构建预测系统的机会，这些系统能够突出显示哪些生理信号最能有效识别吸烟相关风险画像。更多详情请见 https://www.kaggle.com/datasets/kukuroo3/body-signal-of-smoking/data。

- **T2 患者生存预测**。 此任务以一个结构化临床数据集为核心，其中每条记录包含患者的人口统计属性、实验室测量值和疾病相关指标，目标是一个反映不同预后阶段的分类生存结局。任务旨在建模这些医学特征中的模式与生存轨迹之间的关系，使机器学习系统能够学习哪些生物标志物和临床信号最能预测患者风险及长期结局。更多详情请见 https://www.kaggle.com/datasets/joebeachcapital/cirrhosis-patient-survival-prediction/code?datasetId=3873965&sortBy=voteCount。

- **T3 药物反应机制预测**。 此任务使用一个高维生物筛选数据集，其中每个样本对应在特定细胞条件下测试的一种化合物，目标是描述该化合物会扰动哪些生物通路的多标签作用机制指标。目标是学习从基因表达和细胞活力特征到这些作用机制标签的预测映射，使模型能够发现功能特征、区分细微的通路效应，并为后续药物发现提供洞见。更多详情请参阅 https://www.kaggle.com/competitions/lish-moa/data?select=训练集_目标_nonscored.csv。

- **T4 卡路里估算**。 此任务是一项基于 *train.csv* 构建的结构化回归任务，其中每一行代表一条身体活动记录，包含持续时间、心率信号和运动相关指标等特征，目标是一个连续的卡路里值，用于量化能量消耗。目标是学习这些活动特征中的模式如何映射到卡路里消耗，使模型能够捕捉生理关系、提高预测准确率，并深入了解不同形式的运动如何转化为代谢产出。更多详情请参阅 https://www.kaggle.com/competitions/international-medical-ai-and-data-science-calorie/data。

- **T5 HCT后生存分析**。 此任务基于一个具有临床依据的合成数据集构建，其中每一行代表一名异基因HCT患者，包含人口统计变量、疾病特征、移植详情和治疗后临床信号；目标是生成一个实数风险评分，以反映对移植后生存可能性的预测。此任务旨在对生存结局进行建模，使预测既准确，又能确保不同种族群体之间的公平性；同时学习影响HCT后预后的医学特征模式，并尽可能缩小预测性能方面的差异。更多详情请见 https://www.kaggle.com/competitions/equity-post-HCT-survival-predictions/overview。

基于临床图像数据的任务包括：

• **T6 皮肤癌识别**。 此任务提供一个皮肤病变图像数据集，其中部分训练集带有标签，另有相当大一部分未标记；有标签的子集类别严重不平衡，良性病例远多于恶性病例。任务是构建一个图像分类模型，为每张测试集图像预测 LABEL 变量（恶性或良性）的概率；模型需有效利用有限的有标签数据，同时借助大量未标记数据，并缓解类别不平衡问题，从而取得出色的 F1 性能。更多详情请参阅 https://www.kaggle.com/competitions/dsm-2024-competition/overview。

• **T7 黑色素瘤检测**。 此任务基于一个经过整理的皮肤病变图像数据集，其中每个样本都是一张皮肤镜照片，并配有一个二元目标标签，用于指示病变是良性还是恶性。目标是训练一个分类器，学习纹理、颜色变化和结构异常等视觉模式，以区分恶性黑色素瘤与非癌性病变，从而准确预测未见图像的目标标签概率。更多详情请参阅 https://www.kaggle.com/datasets/bhaveshmittal/melanoma-cancer-dataset。

• **T8 视网膜血管分割**。 此任务提供一个视网膜眼底图像数据集，其中每个样本都是一张眼部高分辨率照片，并配有像素级分割掩膜，用 1 标记血管位置，用 0 标记背景。目标是训练一个模型，使其学习视网膜血管系统的视觉结构——捕捉血管宽度、分支模式以及细微的病理变化——从而准确预测每张输入图像的全分辨率血管掩膜，为后续诊断和眼科分析提供支持。更多详情请参见 https://www.kaggle.com/datasets/abdallahwagih/retina-blood-vessel/data。

基于健康时间序列数据的任务包括：

• **T9 基于心电图的心律失常检测**。 此任务使用一个结构化的心电图时间序列数据集，其中每一行对应一名患者，并包含一条心电图信号心跳记录，以及 ID 和节律类别指标等元数据。目标是预测健康状态标签，这是一个二元结果，表示根据原始 PhysioNet 来源中的临床标注，心电图评估结果为正常还是异常。目标是学习原始心电图波形中的判别性模式——包括形态、节律规律性和细微的信号偏差——从而准确分类新患者（即未出现在数据集中的患者）的心脏健康状态。更多详情请参见 https://www.kaggle.com/competitions/ecg-ai-fast-checkup/overview。

• **T10 基于惯性测量单元的手势预测**。 该任务提供了一个从腕戴设备采集的多通道时间序列传感器数据集，其中每个序列包含同步的惯性测量单元运动信号以及温度和接近度读数。目标是一个手势标签，用于指示该动作是否对应于身体聚焦重复行为（BFRB）或非 BFRB 操作，并对特定的 BFRB 手势类型提供更细粒度的区分。该任务旨在训练一个模型，使其学习跨传感器的判别性时间模式，从而可靠地对这些行为进行分类，以支持对 BFRB 相关动作进行更准确且具有临床意义的检测。系统会计算手势是否属于目标类型或非目标类型的二元 F1，以及手势的宏平均 F1（其中所有非目标序列被合并为单一非目标类别）。最终得分是二元 F1分数与宏平均 F1分数的平均值。更多详情请参见 https://www.kaggle.com/competitions/cmi-detect-behavior-with-sensor-data。

• **T11 基于脑电图的癫痫发作预测**。此任务提供了一个大规模脑电图数据集，其中每个训练样本都对应一段从危重患者身上记录的多通道脑电活动，并附有专家对六种可能的有害脑活动模式的标注投票计数：癫痫发作、周期性癫痫样放电、广泛性周期性放电、侧化节律性δ活动、广泛性节律性δ活动和其他。任务是学习一个模型，将原始脑电图信号和派生特征映射为这六个目标类别的概率分布，捕捉细微的时频特征，使系统能够可靠地对有害模式进行分类，并支持更快速、更一致的神经重症评估。更多详情请参阅 https://www.kaggle.com/competitions/hms-harmful-brain-activity-classification/overview。

• **T12 新冠肺炎感染病例预测**。该数据集提供了一个全球COVID-19时间序列表，其中每一行记录特定日期的某个国家或省份，以及关键流行病学计数：确诊病例、康复病例和死亡病例，并附有纬度和经度等地理元数据。该任务旨在对这些病例变化轨迹随时间的演变进行建模——根据分析目标，将其中任一结果变量作为目标——从而使机器学习系统能够捕捉时间模式、预测未来病例数，并揭示区域特征与疫情动态之间的关系。

有关分析目标的信息——从而使机器学习系统能够捕捉时间模式、预测未来病例数，并揭示区域特征与疫情动态之间的关系。更多详情请见https://www.kaggle.com/datasets/rahulgupta21/datahub-covid19/data。

基于自由文本数据的任务包括：

• **T13 孤独症谱系障碍预测**。此任务基于 TASD 数据集，其中每个样本都是对幼儿日常行为的简短文字描述，并标注了注意力反应、词语重复、情感共情、噪声敏感性等结构化行为特征，这些特征均源自早期孤独症谱系障碍评估线索。目标是预测一个二元孤独症谱系障碍标签，指明所描述的行为是否符合孤独症谱系障碍幼儿的表现。此任务旨在学习这些聚焦行为的叙述中的语言和语义模式，使机器学习模型能够从自然文本中检测早期孤独症谱系障碍相关信号，从而支持更具规模性和可解释性的早期筛查工具。更多详情请参阅 https://www.kaggle.com/datasets/sharafatahmed/tasd-dataset-text-based-early-asd-detection/data。

• **T14 COVID-19 情感分类**。此数据集包含数千条与新冠肺炎相关的推文，每条记录都提供推文文本及位置、时间戳等元数据，目标是使用人工标注的情感标签反映推文的情绪基调（例如，积极、消极、中性、极其积极、极其消极）。此任务要求训练一个文本分类模型，学习推文中的语言和上下文线索，以准确预测情感，从而分析疫情期间的公众情绪、虚假信息模式和实时反应。更多详情请参阅 https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification/data。

基于音频数据的任务包括：

• **T15 呼吸系统疾病预测**。 目标是构建一个多分类器，根据提供的音频和元数据，为每个人预测疾病标签，区分健康、患病和患有慢性呼吸系统疾病的个体。性能采用宏 F1 分数进行评估，以反映数据集的类别不平衡，以及需要在所有疾病类别中保持均衡敏感度这一要求。更多详情请参阅 https://www.kaggle.com/competitions/airs-ai-in-respiratory-sounds/overview。

• **T16 构音障碍检测**。 此任务旨在检测语音录音是否来自患有构音障碍的说话者。数据由简短的音频片段及少量说话者信息组成，记录了发音、节律和嗓音质量的自然变化。模型的目标是从这些声学信号中学习规律，可靠地区分构音障碍语音和典型语音，而不依赖任何特定于数据集的文件结构或格式细节。更多详情请参阅 https://www.kaggle.com/datasets/iamhungundji/dysarthria-detection/data。

基于图数据的任务包括：

• **T17 蛋白质链接属性预测**。开放图基准中的 ogbl-ddi 数据集旨在对大规模生物相互作用图进行链接预测。它描述了一个网络，其中节点代表药物，边代表从生物医学知识库整理得到的已知药物相互作用。机器学习任务侧重于根据图的结构模式，预测一对此前未见过的药物是否应形成相互作用边。这使其成为一个清晰且定义明确的基准，可用于评估图神经网络在现实世界药理学相互作用发现中的表现，并与药物安全性、多重用药风险建模和计算药物再利用直接相关。更多详情请参阅 https://ogb.stanford.edu/docs/linkprop/。

一组有代表性的子集　　下文列出了大语言模型或智能体系统的任务规范和基本数据描述　　.

任务 1

任务名称：吸烟者状态预测

[任务概述]
- 任务类型：表格数据二分类 - 模态：结构化表格数据（健康检查/生物信号相关特征） - 下游目标：根据临床测量指标和基本健康指标预测一个人是否为吸烟者。 - 目标列：smoking（`0` = 非吸烟者， `1` = 吸烟者） - 评估指标：准确率 - 目标：训练一个高性能模型，使其能够很好地泛化到未见过的个体，以完成此分类任务。

[数据集路径]
- 数据集根目录：/root/Dataset/classifying_smoker_status/ - 训练集：/root/Dataset/classifying_smoker_status/data/train/train.csv - 验证集：/root/Dataset/classifying_smoker_status/data/val/val.csv - 测试集：/root/Dataset/classifying_smoker_status/data/test/test.csv

[数据集详情]
- 格式：CSV
- 列（简要）： - ID：样本标识符 - 性别：性别 - 年龄：年龄 - 身高(cm)、体重(kg)、腰围(cm)：基本身体测量指标 - 视力(左/右)：视力测量指标 - 听力(左/右)：听力测量指标 - 收缩压、舒张压：血压相关测量指标 - 空腹血糖：空腹血糖 - 胆固醇、甘油三酯、高密度脂蛋白、低密度脂蛋白：血脂指标 - 血红蛋白：血红蛋白水平 - 尿蛋白：尿蛋白指标 - 血清肌酐：血清肌酐 - 天冬氨酸氨基转移酶、丙氨酸氨基转移酶、γ-谷氨酰转移酶：肝脏相关指标 - 口腔：口腔检查状态 - 龋齿：龋齿指标 - 牙结石：牙结石状态 - 吸烟：目标标签

任务 8

任务名称：视网膜血管分割

[任务概述]
- 任务类型：医学图像分割（二值分割） - 模态：带有像素级血管掩膜的视网膜眼底图像 - 下游目标：从眼底图像中分割视网膜血管。 - 目标：二值掩膜图像（血管像素与背景像素） - 评估指标：Dice 系数 - 目标：训练一个高性能分割模型，使其能够在未见过的视网膜图像上生成准确的血管掩膜。

[数据集路径]
- 数据集根目录：/root/Dataset/Retina_Blood_Vessel_for_segmentation/ - 训练图像：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/train/image/ - 训练掩膜：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/train/mask/ - 验证图像：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/val/

图像/
- 验证掩膜：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/val/mask/

- 测试图像：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/test/image/ - 测试掩膜：/root/Dataset/Retina_Blood_Vessel_for_segmentation/data/test/mask/

[数据集详情]
- 格式：PNG 图像 - 配对文件（简介）： - image/：视网膜眼底图像 - mask/：文件名相同的对应二值血管掩膜

任务 15

任务名称：呼吸系统疾病预测

[**任务概述**]
- 任务类型：多模态分类（表格数据 + 音频）- 模态：结构化表格元数据 + 呼吸音频录音（咳嗽和元音）- 下游目标：根据元数据和音频信号，预测每位候选者的呼吸系统疾病类别。- 目标列：疾病（分类标签）- 评估指标：宏平均 F1 - 目标：训练一个性能优异的模型，使其能够很好地泛化到未见过的受试者，从而可靠地检测呼吸系统疾病。

[**数据集路径**]
- 数据集根目录：/root/Dataset/medical_sound_classification/ - 训练集表格：/root/Dataset/medical_sound_classification/data/train/train.csv - 训练集音频：/root/Dataset/medical_sound_classification/data/train/sounds/ - 验证集表格：/root/Dataset/medical_sound_classification/data/val/val.csv - 验证集音频：/root/Dataset/medical_sound_classification/data/val/sounds/ - 测试集表格：/root/Dataset/medical_sound_classification/data/test/test.csv - 测试集音频：/root/Dataset/medical_sound_classification/data/test/sounds/

[**数据集详情**]
- 格式：CSV（表格） + WAV（音频） + JSON（预计算嵌入）- 表格列（简述）：

  - candidateID：候选者标识符（对应音频文件夹）- 年龄、性别：基本人口统计信息 - tbContactHistory、wheezingHistory、phlegmCough、familyAsthmaHistory、feverHistory、coldPresent：症状/病史指标 - 包年：吸烟暴露指标 - 疾病：目标标签

- 音频/嵌入文件（简述）： - 每个候选者文件夹都包含咳嗽音频文件、元音音频文件，以及可选的嵌入文件（emb_cough.json、emb_vowel.json）。

A.2. 基线

大语言模型提示词

你是一名资深机器学习工程师。你的目标是构建一个具有最佳预测性能的模型，并选择合适的不确定性估计方法与该模型集成，以便计算并报告不确定性相关指标（例如 ECE、布里尔分数、PCIP）。根据用户的 **任务描述**，编写一个完整的 Python 脚本。

要求：
1. 脚本必须直接从提供的文件路径加载数据。2. 脚本必须计算并打印任务指定的所有评估指标。3. 代码必须包含在 python 和 block 中。4. 请勿使用 input() 函数；脚本必须能够全自动运行。

除了完成建模任务外，还需选择一种合适的不确定性估计方法，并输出相应的不确定性分析结果。

Claude Code 提示词

请仔细阅读任务文件**任务描述**，并严格遵守所有指示，不要询问澄清问题。

你的任务是训练一个预测性能最优的模型，并选择合适的不确定性估计方法与模型集成，以便计算并报告不确定性相关指标（例如 ECE、布里尔分数）。

开始每项任务前，请在“OUTPUTROOT”下创建对应的任务目录，并使用任务名称和时间戳命名。

每项任务的所有输出和执行脚本都必须放在对应目录中。任务完成后，请简要总结已执行的步骤和结果，并报告模型性能指标和不确定性分析结果。如果任务失败，请明确报告失败情况。最终输出应写入名为 result.txt 的文件。

不要请求确认。直接执行文件写入和命令执行。所有代码都应由你直接运行。

B. 结果的补充细节

本节展示详细的实验结果。表格10 和12 报告了归一化的下游性能，以便直接比较。

任务实现图3 显示，只有*Claude Code* 和我们提出的*AutoHealth*能够端到端成功完成所有任务，而其他方法则在模型构建或不确定性量化阶段失败。朴素大语言模型基线只有在提供精心设计的提示词并手动执行生成的代码时，才能取得合理的性能。¨ *DS-Agent* 和*AutoML-Agent* 在若干任务（例如任务 10、11、15、16 和 17）上失败，主要原因是其预定义流水线的模态泛化能力有限。总体而言，在所有任务中，朴素大语言模型、¨Claude Code，*Claude Code*、*DS-Agent*、*AutoML-Agent*和*AutoHealth* 的模型成功率分别为 35.3%、100%、47.1%、64.7% 和 100%（表格7）。在不确定性量化方面，相应的成功率分别为 11.8%、100%、0%、0% 和 100%（表格8）。

下游性能表格10 和12 报告了归一化后的下游性能，以便直接比较不同方法。我们的模型平均性能得分为 0.731，平均不确定性得分为 0.602，分别比排名第二的结果（0.578 和 0.408）高出 26.5% 和 47.5%。如图3所示，*AutoHealth* 达到了

在 17 项任务中的 16 项上实现了最先进性能，其余两项任务的结果也非常接近表现最佳的基线。

表 7。 代码执行成功率对比。值为 1 表示成功，值为 0 表示失败。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	通过率
LLM	0	0	0	1	0	0	1	0	1	0	0	1	1	1	0	0	0	35.3%
Claude Code	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
DS-Agent	1	1	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	47.1%
AutoML-Agent	1	1	1	1	1	1	1	0	1	0	0	1	1	1	0	0	0	64.7%
AutoHealth	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%

表格 *8.* 结合不确定性分析的成功率比较。值为 1 表示成功，值为 0 表示失败。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	通过率
LLM	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	11.8%
Claude Code	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
DS-Agent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0%
AutoML-Agent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0%
Auto-Health	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%

表 *9.* 性能比较。↑表示越高越好，↓表示越低越好。最佳结果以**粗体**显示。

方法	T1 准确率(↑)	T2 准确率(↑)	T3 损失(↓)	T4 RMSLE(↓)	T5 C-idx(↑)	T6 F1分数(↑)	T7 准确率(↑)	T8 Dice系数(↑)	T9 F1分数(↑)	T10 平均值F1分数(↑)	T11 KL(↓)	T12 平均绝对误差(↓)	T13 准确率(↑)	T14 准确率(↑)	T15 mF1分数(↑)	T16 准确率(↑)	T17 命中数(↑)
LLM	Fail	Fail	Fail	0.083	Fail	Fail	0.784	Fail	0.848	Fail	Fail	0.827	0.857	0.571	Fail	Fail	Fail
Claude Code	0.766	0.828	0.168	0.063	0.637	0.335	0.531	0.799	**0.952**	0.398	0.832	0.701	0.943	0.750	0.792	0.655	0.215
DS-Agent	0.719	0.882	Fail	0.064	0.602	Fail	Fail	Fail	0.862	Fail	Fail	0.680	0.842	0.591	Fail	Fail	Fail
AutoML-Agent	**0.805**	**0.927**	0.123	0.062	**0.663**	0.349	0.852	N/A	Fail	Fail	Fail	0.805	0.914	**0.841**	N/A	N/A	N/A
Auto-Health	0.793	0.920	**0.063**	**0.062**	0.646	**0.717**	**0.969**	**0.822**	0.870	**0.632**	**0.756**	**0.084**	**0.982**	0.811	**0.838**	**0.835**	**0.965**

表 *10.* 性能分析的归一化结果（对应表格 9），范围为 0 至 1。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	AVG
LLM	0	0	0	0.546	0	0	0.784	0	0.848	0	0	0.108	0.857	0.571	0	0	0	0.218
Claude Code	0.766	0.828	0.373	0.613	0.637	0.335	0.531	0.799	**0.952**	0.398	0.107	0.125	0.943	0.750	0.792	0.655	0.215	0.578
DS-Agent	0.719	0.882	0	0.610	0.602	0	0	0	0.862	0	0	0.128	0.842	0.591	0	0	0	0.308
AutoML-Agent	**0.805**	**0.927**	0.448	0.617	**0.663**	0.349	0.852	0	0	0	0	0.110	0.914	**0.841**	0	0	0	0.384
Auto-Health	0.793	0.920	**0.613**	**0.617**	0.646	**0.717**	**0.969**	**0.822**	0.870	**0.632**	**0.117**	**0.543**	**0.982**	0.811	**0.838**	**0.835**	**0.965**	0.747

表*11.* 不确定性分析结果。**布里尔分数**：布里尔分数， **ΔP**： Δ PCIP，**ECE**：期望校准误差。最佳结果以**粗体** 标出（数值越低越好）。

方法	T1 BS	T2 BS	T3 BS	T4 ΔP	T5 ΔP	T6 ECE	T7 ECE	T8 BS	T9 ECE	T10 ECE	T11 ECE	T12 ΔP	T13 ECE	T14 ECE	T15 ECE	T16 ECE	T17 BS
LLM	Fail	Fail	Fail	0.018	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0.505	Fail	Fail	Fail	Fail	Fail
Claude Code	0.152	0.128	0.030	0.227	0.889	0.265	0.244	0.100	**0.023**	**0.116**	**0.174**	0.352	0.059	0.211	0.196	0.241	**0.138**
DS-Agent	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AutoML-Agent	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AutoHealth	**0.036**	**0.075**	**0.013**	**0.003**	**0.008**	**0.049**	**0.009**	**0.044**	0.097	0.158	0.190	**0.225**	**0.055**	**0.070**	**0.080**	**0.133**	0.252

表*12*。 不确定性分析的归一化结果（对应表 11），范围为0到1。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	AVG
LLM	0	0	0	0.847	0	0	0	0	0	0	0	0.165	0	0	0	0	0	0.060
Claude Code	0.397	0.439	0.769	0.306	0.101	0.274	0.291	0.500	0.813	0.463	0.365	0.221	0.629	0.322	0.338	0.293	0.420	0.408
DS-Agent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AutoML-Agent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AutoHealth	0.735	0.571	0.885	0.971	0.926	0.671	0.917	0.694	0.508	0.388	0.345	0.308	0.645	0.588	0.556	0.429	0.284	0.613

表*13*。成功率，其中 0 表示失败，1 表示成功。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	AVG
LLM	0	0	0	1	0	0	0.5	0	0.5	0	0	1	0.5	0.5	0	0	0	0.235
Claude Code	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.000
DS-Agent	0.5	0.5	0	0.5	0.5	0	0	0	0.5	0	0	0.5	0.5	0.5	0	0	0	0.235
AutoML-Agent	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0.5	0	0	0.5	0.5	0.5	0	0	0	0.324
AutoHealth	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1.000**

表*14.* NPS，取值范围为 0 到 1。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	AVG
LLM	0.000	0.000	0.000	0.696	0.000	0.000	0.392	0.000	0.424	0.000	0.000	0.137	0.429	0.286	0.000	0.000	0.000	0.139
Claude Code	0.582	0.634	0.571	0.461	0.369	0.305	0.411	0.649	**0.883**	0.430	0.236	0.173	0.787	0.536	0.565	0.474	0.317	0.493
DS-Agent	0.360	0.441	0.000	0.305	0.301	0.000	0.000	0.000	0.431	0.000	0.000	0.064	0.421	0.295	0.000	0.000	0.000	0.154
AutoML-Agent	0.403	0.463	0.225	0.308	0.331	0.175	0.426	0.000	0.000	0.000	0.000	0.055	0.457	0.420	0.000	0.000	0.000	0.192
AutoHealth	**0.765**	**0.746**	**0.751**	**0.795**	**0.787**	**0.694**	**0.943**	**0.757**	0.688	**0.509**	**0.231**	**0.425**	**0.814**	**0.700**	**0.697**	**0.633**	**0.624**	**0.680**

表*15.* 综合得分，范围为0至1。

方法	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	AVG
LLM	0	0	0	0.848	0	0	0.446	0	0.462	0	0	0.568	0.464	0.393	0	0	0	0.187
Claude Code	0.791	0.817	0.786	0.730	0.685	0.652	0.705	0.825	0.942	0.715	0.618	0.587	0.893	0.768	0.783	0.737	0.659	0.747
DS-Agent	0.430	0.471	0	0.402	0.401	0	0	0	0.466	0	0	0.282	0.461	0.398	0	0	0	0.195
AutoML-Agent	0.451	0.482	0.362	0.404	0.416	0.337	0.463	0	0.250	0	0	0.278	0.479	0.460	0	0	0	0.258
AutoHealth	**0.883**	**0.873**	**0.875**	**0.898**	**0.894**	**0.847**	**0.971**	**0.879**	**0.844**	**0.755**	**0.615**	**0.713**	**0.907**	**0.850**	**0.849**	**0.816**	**0.812**	**0.840**

表格*16*。不确定性估计性能：原始值（归一化分数）

方法	布里尔分数 ↓			ΔPCIP ↓		平均值（归一化）
	T1	T2	T3	T4	T5	
Claude Code	0.152 (0.397)	0.128 (0.440)	0.030 (0.769)	0.227 (0.306)	0.889 (0.101)	0.403
AutoHealth	0.035 (0.738)	0.075 (0.572)	0.013 (0.887)	0.003 (0.971)	0.008 (0.929)	0.819
AutoHealth-T	0.071（0.587）	0.095（0.513）	0.013（0.887）	0.024（0.806）	0.015（0.873）	0.733
AutoHealth-R	0.144 (0.410)	0.078 (0.561)	0.024 (0.806)	0.034 (0.746)	0.038 (0.725)	0.650

令牌与成本分析。表格17 展示了令牌使用量、货币成本和实际运行时间，图5 则展示了各阶段的执行时间占比。在这 17 项任务中，令牌消耗总量保持在适中范围内，而货币成本始终较低，通常每项任务不到 6 人民币。值得注意的是，*AutoHealth* 展现出快速的迭代周期：大多数任务均可在 2–6 小时内完成（5 次运行），即使是涉及吉字节级数据的任务也不例外。这种高效率主要归功于高效且有条理的多智能体协作，显著减少了迭代过程中的冗余令牌消耗。通过将任务理解、模型构建、执行和反馈整合到统一的端到端智能体工作流中，*AutoHealth* 最大限度地减少了人工干预和迭代开销。综上，这些结果表明，*AutoHealth* 能够以较低成本快速开展端到端实验，从而降低大规模实证研究的门槛，并加速真实世界健康数据建模领域的科学发现。

表*17.* 部分任务的令牌数和时间成本。输入、输出和缓存令牌数以百万（M）为单位报告。

Task	数据大小	输入	输出	缓存	成本（人民币）	时间（分钟）
T1	11.03MB	1.58	0.20	0.93	2.1	119.8
T2	31.41MB	2.24	0.27	1.39	2.8	156.9
T3	215.96MB	2.60	0.35	1.47	3.6	470.9
T4	35.46MB	2.01	0.23	1.27	2.4	123.0
T5	9.87MB	2.65	0.38	1.49	3.8	246.1
T6	143.5MB	3.49	0.55	1.76	5.5	478.7
T7	83.27MB	3.28	0.40	1.78	4.6	360.2
T8	34.76MB	4.05	0.65	1.98	6.5	344.1
T9	7.75MB	1.81	0.22	1.14	2.2	123.2
T10	1.12GB	3.40	0.38	1.97	4.4	214.4
T11	6.6GB	3.73	0.47	1.83	5.6	634.1
T12	3.18MB	1.83	0.20	1.11	2.3	109.8
T13	57.15KB	1.72	0.24	1.04	2.3	231.2
T14	11.5MB	3.21	0.39	1.88	4.2	332.3
T15	1.9GB	3.03	0.38	1.69	4.2	187.5
T16	200MB	2.80	0.38	1.66	3.8	220.6
T17	102MB	3.38	0.47	1.77	5.0	419.5

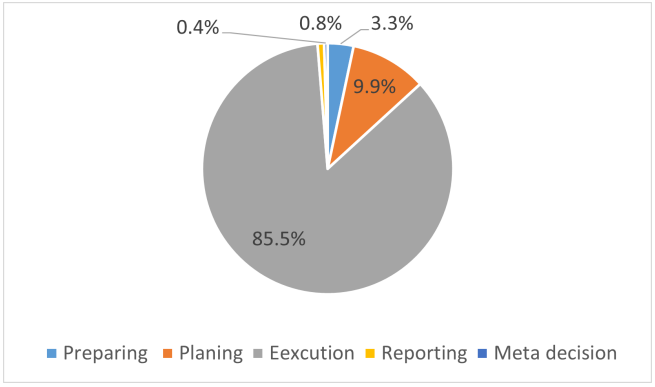


图*5.* 各阶段的执行时间占比。

C. 任务 15 的完整报告

以下 7 页包含任务 15 的生成报告。

任务 15

任务名称：呼吸系统疾病预测

[**任务概述**]
- 任务类型：多模态分类（表格数据 + 音频）- 模态：结构化表格元数据 + 呼吸音频录音（咳嗽和元音）- 下游目标：根据元数据和音频信号，为每位候选者预测其呼吸系统疾病类别。- 目标列：疾病（分类标签）- 评估指标：宏平均 F1 - 目标：训练一个高性能模型，使其能够很好地泛化到未见过的受试者，从而实现稳健的呼吸系统疾病检测。

[**数据集路径**]
- 数据集根目录：/root/Dataset/medical_sound_classification/ - 训练集表格：/root/Dataset/medical_sound_classification/data/train/train.csv - 训练集音频：/root/Dataset/medical_sound_classification/data/train/sounds/ - 验证集表格：/root/Dataset/medical_sound_classification/data/val/val.csv - 验证集音频：/root/Dataset/medical_sound_classification/data/val/sounds/ - 测试集表格：/root/Dataset/medical_sound_classification/data/test/test.csv - 测试集音频：/root/Dataset/medical_sound_classification/data/test/sounds/

[**数据集详情**]
- 格式：CSV（表格） + WAV（音频） + JSON（预计算嵌入）- 表格列（简述）：- candidateID：候选者标识符（映射到对应的音频文件夹）- 年龄、性别：基本人口统计信息 - tbContactHistory、wheezingHistory、phlegmCough、familyAsthmaHistory、feverHistory、coldPresent：症状／病史指标 - 包年：吸烟暴露指标 - 疾病：目标标签 - 音频／嵌入文件（简述）：- 每个候选者文件夹均包含 cough.wav、vowel.wav，以及可选的嵌入文件（emb_cough.json、emb_vowel.json）。

基于多模态数据的呼吸系统疾病预测：数据、模型、性能与不确定性分析报告

由 AutoHealth 提供

摘要

本报告详述了 AutoHealth 开展的实验。AutoHealth 是一个为自主健康数据建模而设计的多智能体系统。本任务旨在结合结构化表格元数据（人口统计信息、症状、历史记录）与呼吸音频录音（咳嗽和元音）来预测呼吸系统疾病。我们开发了一种双分支晚期融合神经网络，整合来自两种模态的特征。为增强稳健性并提供不确定性估计，我们以受控的架构变体训练了由三个模型组成的深度集成。最终模型在测试集上取得了 0.8384 的宏平均 F1，展现出较强的泛化能力。全面的不确定性分析表明，错误预测与更高的模型不确定性相关，从而为在临床部署中识别低置信度病例提供了有价值的工具。本报告详细涵盖数据分析、模型训练方法、性能评估，以及面向临床应用的可操作的见解。

1 数据分析

1.1 数据集概述

该数据集是一个用于呼吸系统疾病分类的多模态数据集，包含 546 名不同的候选者，分为训练集（437 人）、验证集（54 人）和测试集（55 人）。每位候选者由两种数据模态表示：

- **表格元数据**：一个包含 11 列的 CSV 文件，记录人口统计信息（年龄、性别）、症状/病史指标（例如，wheezingHistory、phlegmCough、familyAsthmaHistory）、吸烟暴露（packYears）以及目标标签（疾病：类别 0、1 或 2）。
- **音频录音**：每位候选者有两个 WAV 文件：一段咳嗽录音（平均时长 16.0 秒）和一段元音录音（平均时长 6.1 秒），采样率均为 48 kHz。此外，还提供预先计算的音频嵌入（JSON 文件），用于表示从音频信号中提取的深度特征。

主要评估指标是宏平均 F1，适用于应对观察到的类别不平衡问题。

1.2 特征分析与临床相关性

分析表明，数据中存在较强的预测信号，与临床直觉相符 .

- **表格特征**：年龄和包年（吸烟暴露）与疾病标签表现出较高的点二列相关（分别为0.567和0.372）。喘息史和咳痰等类别特征也高度显著（卡方p $<$ 1e-07）。例如，类别1的喘息史阳性率为84.7%，而类别0仅为5.3%。

- **音频特征**：对变长咳嗽嵌入进行均值池化后，512个特征维度中有206个显示出显著的类间差异（方差分析 $p < 0.05$）。这些音频特征与强表格特征的相关性较低（$|r| < 0.1$），表明它们为分类提供了互补且非冗余的信息。

1.3 数据质量与挑战

在预处理流程中发现并处理了若干数据质量问题：

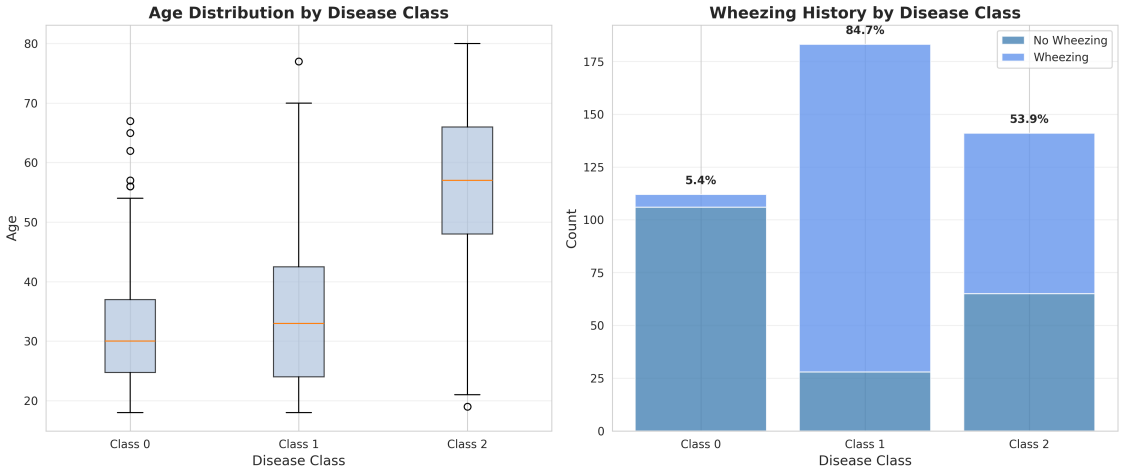


图1：关键特征与目标疾病类别之间的关系。**左图：**不同疾病类别的年龄分布，显示类别0年龄最小，类别2年龄最大。**右图：**各类别中的喘鸣史发生率，突出其与类别1的强关联。这些特征在模型设计过程中得到了优先考虑。

- **缺失值**：‘coldPresent‘ 列存在大量缺失值（训练集中有 115 个）。这些缺失值使用训练数据中的众数进行了填补。
- **类别不平衡**：目标分布不均衡。在训练集中，类别1出现频率最高（183个样本），其次是类别2（141个）和类别0（113个）。训练过程中对此不平衡进行了积极缓解。
- **音频嵌入不一致性**：预计算嵌入的序列长度存在差异（例如(1,512)、(4,512)）。应用全局平均池化操作，将其标准化为定长向量。
- **分布偏移**：柯尔莫哥洛夫-斯米尔诺夫检验表明，在训练集与验证集/测试集之间，部分音频嵌入维度存在轻微的分布偏移，因此需要谨慎进行特征缩放。

2 模型训练

2.1 数据预处理

我们实施了全面的预处理流程，以准备多模态数据 ：

- **表格数据**：分类变量采用标签编码。将“coldPresent”中的缺失值填充为训练集的众数。创建了一个简单的交互特征“老年伴喘鸣”。数值特征（“年龄”、“包年”）使用训练集的均值和标准差进行标准化。
- **音频数据**：针对每位候选者，加载了“emb cough.json”和“emb vowel.json”文件。将变长序列分别进行均值池化，得到两个 512 维向量，再将其拼接为 1024 维音频特征向量，随后进行标准化。
- **类别不平衡缓解**：采用了两种策略：1）计算类别权重（[1.25、0.77、1.00]），用于损失函数，以加重对少数类别误差的惩罚。2）训练期间使用“WeightedRandomSampler”，确保每个小批量均衡包含全部三个类别。
- **数据增强**：为提升泛化能力，在训练过程中分别向表格数据和音频特征向量添加高斯噪声（标准差分别为=0.03 和 0.02）。

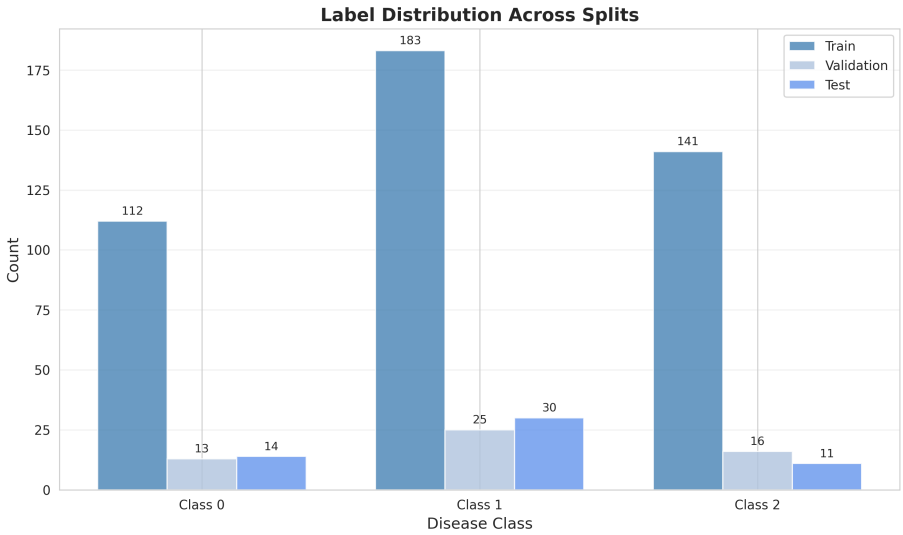


图 2：训练集、验证集和测试划分中的标签分布。类别不平衡（类别 1> 类别 2 > 类别0）在各划分中保持一致，因此在模型训练期间使用类别加权损失和平衡采样策略是合理的，可防止偏差。

2.2 模型架构

设计了一种定制的双分支晚期融合神经网络，以有效结合两种模态的信息。

- **音频分支**：多层感知机（MLP）接收 1024 维音频向量，并通过包含批归一化和随机失活的层（$1024 \to 512 \to 256$）进行投影，输出 256 维表示。
- **表格分支**：独立的 MLP 处理 10 维表格向量，并通过层（$10 \to 64 \to 32$）进行变换，输出 32 维表示。
- **融合与分类**：将两个分支的输出进行拼接（288 维）。最终的分类头（MLP：$288 \to 128 \to 64 \to 3$）生成三个疾病类别对应的对数几率。

该架构 架构使每个分支都能先学习专门的表示，再将它们组合起来用于最终决策

2.3 超参数

我们选择了关键超参数，以平衡学习能力与正则化：

- **优化器**：AdamW，初始学习率为 $lr = 5 \times 10^{-4}$，权重衰减为 5×10^{-4}。
- **学习率调度**：ReduceLROnPlateau 调度器（factor=0.5，patience=10 训练轮次）。
- **正则化**：在关键层使用 0.4 的随机失活率，并结合批归一化，以防止过拟合。
- **训练**：批次大小为 32，最多训练 80 轮次，并根据验证损失进行早停（耐心值=20）。
- **集成**：使用不同的随机种子（42、123、456）训练了三个相互独立的模型，并对音频分支的随机失活率（0.35、0.4、0.45）进行了细微调整，以增加多样性。

2.4 训练过程

训练过程稳定，并呈现出明显的收敛趋势。表现最佳的单模型（随机种子=123）的验证集宏平均 F1 最高，达到 0.8692。训练历史记录（图 3）显示，训练和验证指标在前 15 个训练轮次中均迅速提升，随后趋于平稳。早停在第 39 轮触发，表明训练效率高且没有过拟合。采用类别平衡采样和数据增强有助于绘制出平滑的学习曲线。

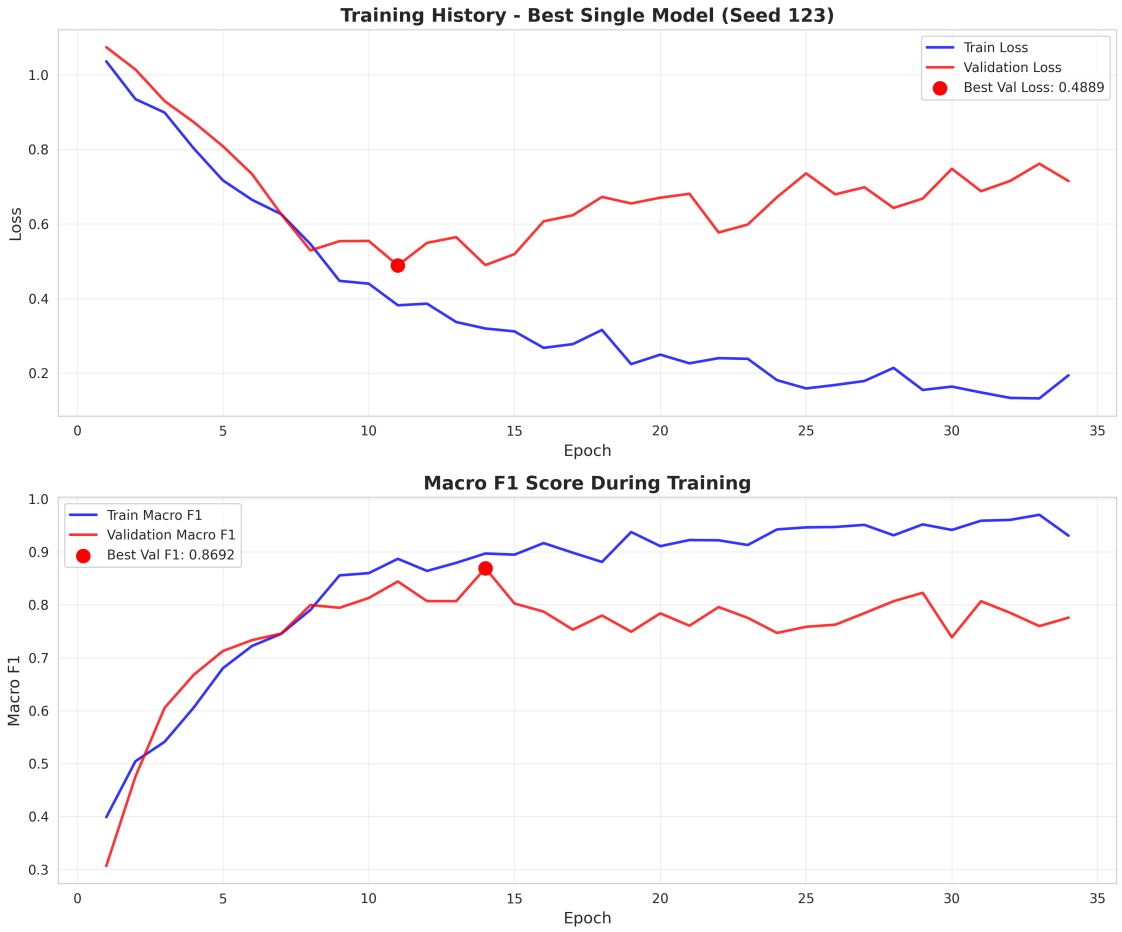


图3：最佳单模型的训练历史记录。**顶部：**训练集和验证集损失曲线，显示损失持续下降并逐渐收敛。**底部：**训练集和验证集宏平均F1分数，显示模型表现持续提升，最终验证集 F1分数为 0.8692。图中标出了早停点。

3 模型性能

最终模　　型是由三个独立训练的网络组成的深度集成　　．它们的预测取平均值使用温度缩放进行校准。

表1：深度集成模型在测试集上的性能

指标	值
宏平均 F1	0.8384
准确率	0.8545
加权F1	0.8569
期望校准误差 (ECE)	0.0797
布里尔分数	0.2182

集成的测试集宏平均 F1 达到 **0.8384**，准确率为 **85.45%**，表明整体性能强劲。混淆矩阵（图 4）显示，三个任务的召回率相近（约为 0.8），表明所提出的方法能够有效处理不平衡训练数据。较低的 ECE 表明模型的预测概率校准良好，也就是说，置信度为 80% 的预测约有 80% 的时候是正确的。

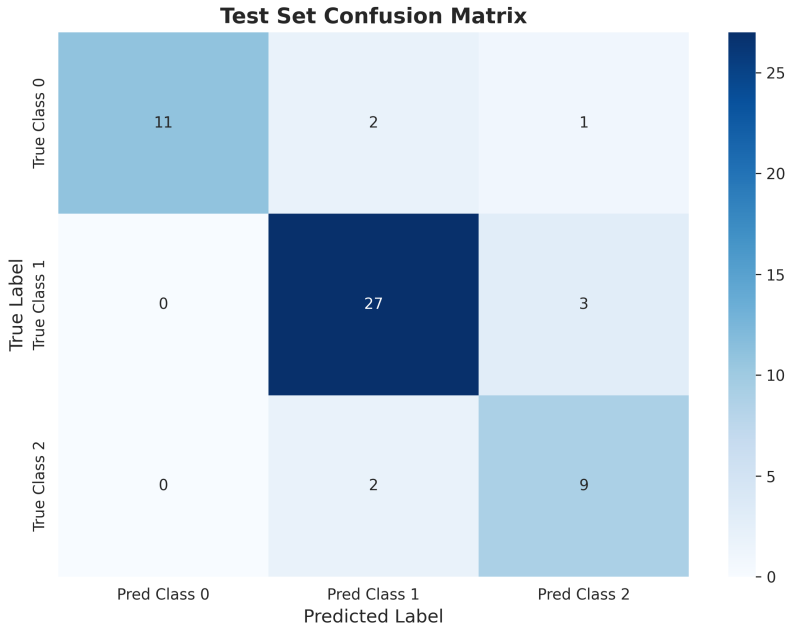


图4：测试集预测结果的归一化混淆矩阵。模型在类别0和类别1上的表现良好（对角线数值较高）。类别2的误差较多，其中若干样本被错误分类为类别1，反映出从样本量最少的类别中学习较为困难。

4 不确定性分析

深度集成框架提供两种不确定性：**偶然不确定性**（数据噪声）和**认知不确定性**（模型不确定性）。我们关注总不确定性，即集成的校准预测分布的熵。

不确定性的临床效用：图6 展示了一项关键发现：错误预测往往伴随着较高的不确定性。这为临床部署提供了一种实用工具。临床医生可以为模型的不确定性得分设定阈值。低于该阈值的预测可以被信任，并有望实现自动化处理；而不确定性较高的预测则可以标记出来，交由专家人工审核。这样便构建了一个人在回路系统，从而提高安全性和效率。

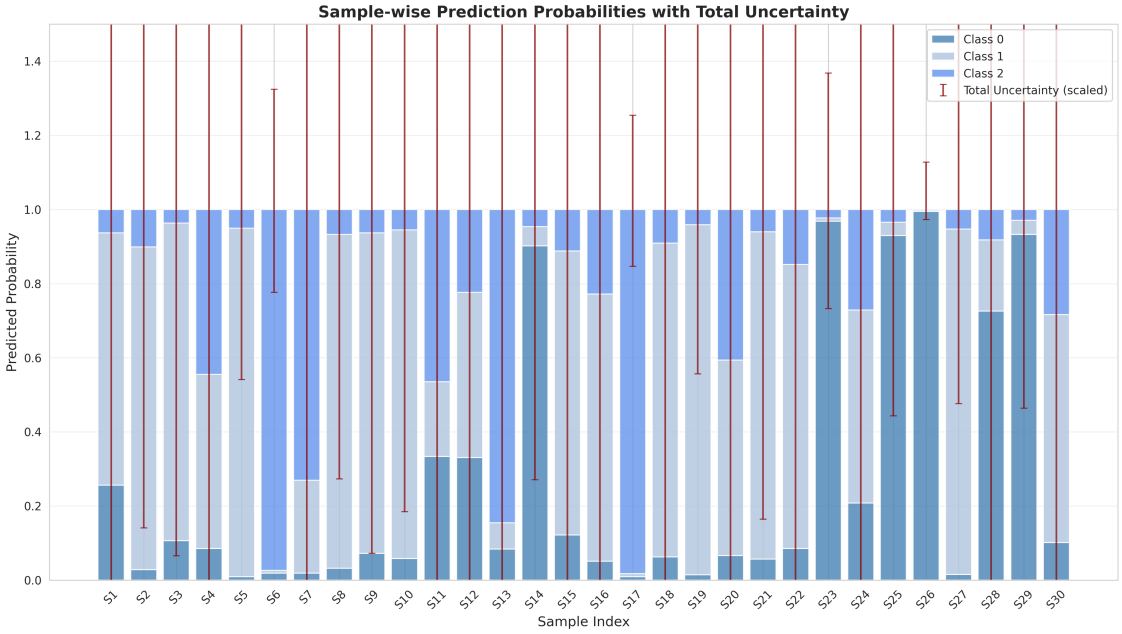


图5：30个测试样本的样本级不确定性。堆叠条形图显示了每个类别的预测概率。红色误差棒表示总不确定性（熵）。不确定性较高（误差棒较长）的样本，其各类别的概率分布往往更加均匀，这表明模型置信度较低。

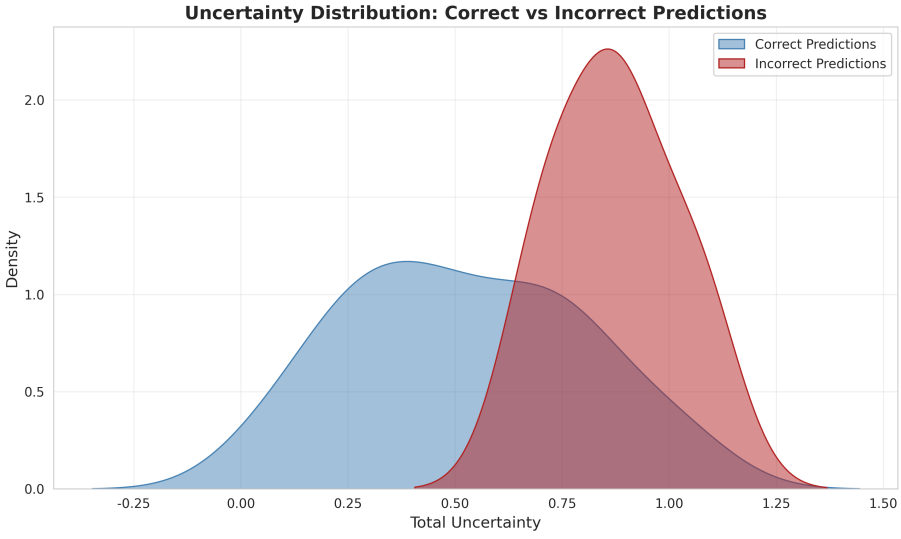


图6：正确预测与错误预测样本的总不确定性分布。错误预测（红色）的平均不确定性显著高于正确预测（蓝色）。这种明显的区分意味着，不确定性可以可靠地标记出更有可能出错的预测。

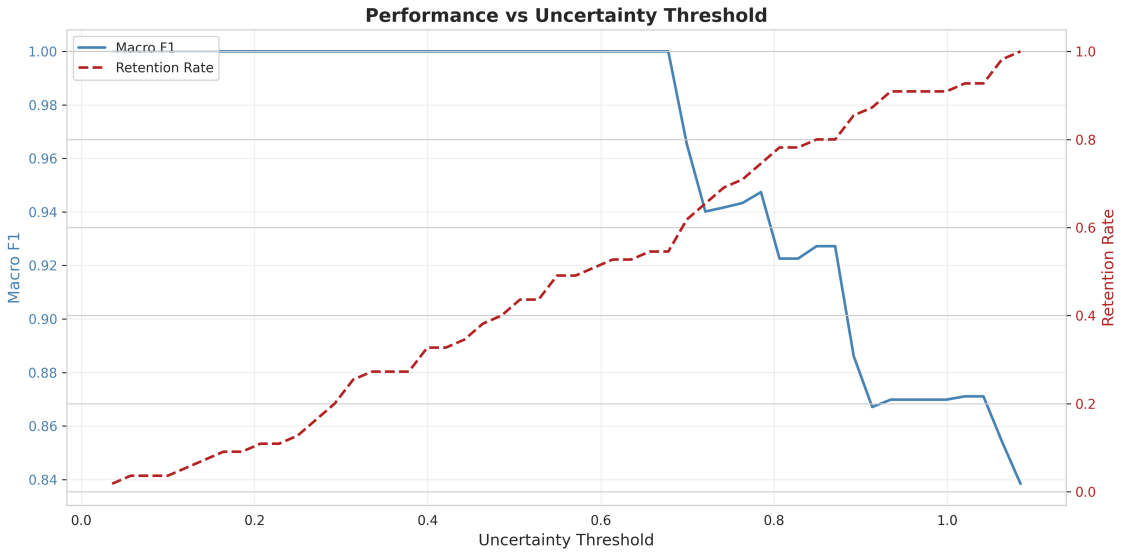


图7：预测可靠性与样本覆盖率之间的权衡。随着不确定性阈值升高（允许纳入更多高不确定性样本），保留的“高置信度”样本上的宏平均 F1（蓝线）起初保持较高，但最终下降。红线表示所保留测试样本的比例。可根据所需的置信度水平选择最优工作点。

图7 量化了这一权衡。通过拒绝高不确定性样本，剩余高置信度子集上的性能（宏平均 F1）可提升至高于总体测试得分，代价是不对部分病例做出预测。例如，拒绝不确定性最高的 20% 样本，可使系统在其自动化决策上获得更高的准确率。

5 结论

本次 AutoHealth 实验成功开发了一个稳健的多模态模型，利用表格元数据和音频特征进行呼吸系统疾病分类。双分支深度集成模型取得了 0.8384 的测试集宏平均 F1，表明互补数据源得到了有效融合。

主要发现与建议：

1. **模型效果良好，但仍有改进空间**：尽管集成模型表现出色且均衡，但仍存在类别混淆，说明通过采用更具区分性的建模方法，性能仍有提升空间。2. **将不确定性作为安全过滤器**：模型的预测不确定性是潜在错误的可靠指标。我们强烈建议在任何部署流水线中设置不确定性阈值。对于不确定性较低的预测，可以放心采取相应操作；而对于不确定性较高的情况，则应由临床医生复核。3. **数据质量是基础**：针对缺失值、嵌入不一致和类别不平衡所采取的预处理步骤，对取得这一性能至关重要。未来的数据收集应致力于从来源上减少这些问题。4. **可操作的指导方案**：针对临床应用，我们提出一个两级系统：1）**自动分诊**：模型提供诊断结果和不确定性得分。2）**临床医生复核**：对于不确定性较高或预测标签为类别2的病例，系统会自动转交专家复核。这种做法兼顾效率与诊断安全。

所开发的模型及其相关的不确定性估计，为决策支持工具奠定了坚实基础，有助于提高呼吸系统疾病筛查的效率和准确率。

D. AutoHealth 的补充介绍

下面提供系统流水线设计的主要代码。

runpipeline.py

```
来自 __未来__ 导入 注解

导入 json
导入 re
from dataclasses import dataclass
from 日期时间 import 日期时间
from pathlib import Path
来自 typing 导入 Any, Dict, List, Optional, Tuple

from .代理 import (
    MetaAgent, #元智能体
    规划智能体, #设计智能体
    数据理解智能体, #数据智能体
    代码执行智能体, #Coding-Agent
    报告生成智能体, #Report-Agent
    简单报告输入,
)
从.config 导入get_代理_config, get_执行_config, get_大语言模型_config,
    加载_配置
来自 .llm 导入 OpenAICompatClient

...

@dataclass
类别 RoundArtifacts:
    轮次_idx: 整数
    数据_报告_Path: 字符串 =""
    计划_Path: 字符串 = ""
    code_执行_目录: 字符串 = ""
    code_执行_状态: 字符串 = ""
    反馈_报告_Path: 字符串 = ""
    指标_Path: 字符串 = ""
    主要_指标_值: Optional[浮点数] = 无
    阶段_持续时间: Dict[字符串, 浮点数] = 无
    阶段_令牌_使用量: Dict[字符串, 字典[str, int]] = 无


定义 运行_流水线(
    *,
    task_文件: str ,
    输出_根目录: 可选[str] = 无,
    max_轮次: 整数 = 5 ,
    耐心: int = 1,
    min_增量: float = 0.0 ,
    恢复_轮次_目录: Optional[str] = None,
    恢复_从_阶段: Optional[str] =None,
) -> Dict[str, 任意]:
    """
      运行完整流水线：数据理解 ->计划生成 -> 代码
          生成 -> 反馈分析 -> 元决策

    如果元决策已完成，请继续进行报告生成。
    """
    任务_路径 = Path(任务_文件).resolve()
    任务_文本 = _read_文本(任务_路径)
    任务_名称, eval_指标, 数据_路径 = _parse_任务_文件(任务_路径)
```

项目_根目录 = Path(__文件__).resolve().parent 如果恢复轮次目录:_ _流水线_根目录 = Path(恢复_轮次_目录).resolve().parent elif 输出_根目录: 流水线_根目录 = Path(输出_根目录).resolve() else: 流水线_根目录 = 项目_根目录 / "outputs" / "pipeline" / 任务_名称 / _现在_时间戳() 流水线_根目录.mkdir(parents=True, exist_ok=True) 配置 = 加载_配置() 设备_信息 = _构建_设备_信息() def _创建_客户端(agent_名称: str) -> Tuple[OpenAICompatClient, Dict[str, Any]]: 大语言模型_配置 = 获取_大语言模型_配置(配置, agent_名称=代理_名称) 客户端 = OpenAICompatClient(api_密钥=大语言模型_配置["api_密钥"], 基础_URL=大语言模型_配置["基础_URL"]) return 客户端, 大语言模型_配置 历史_摘要: Dict[str, Any] = {"最佳_指标_值": None, "最佳_轮次_索引": None, "未_改进_轮次": 0, "之前_轮次": []} 产物 = 按_轮次 : List[轮次产物] = []下一_开始 = "数据_理解"报告_结果 : Optional[Dict[str, str]] = None 快照_路径 =流水线_根目录 / "snapshots.md" 开始 轮次 = 1_恢复目录 = None_if 恢复轮次目录:_ _候选 = Path(恢复_轮次_目录).resolve() if 候选.exists(): 恢复目录 = 候选_匹配 = re.search(r"轮次_(\d+)", str(恢复_轮次_目录)) if 匹配: 开始_轮次 = int(匹配.group(1)) for轮次_索引 in range(开始_轮次, 最大_轮次 + 1): 轮次_目录 = 恢复_目录 if 恢复_目录 and 轮次_索引 == 开始_轮次 else (流水线_根目录 / f"轮次_{轮次_索引:02d}") 轮次_目录.mkdir(parents=True, exist_ok=True) 数据_报告_路径 = "" 计划_路径 = "" 计划_文本 = "" 反馈_报告_文本 = "" 执行结果文本 =""_ _阶段_持续时间: Dict[str, float] = {} 阶段_令牌_使用情况: Dict[str, Dict[str, int]] ={} if 恢复目录:_数据_报告_路径 = str((恢复_目录 / "数据_理解" / "report.md"). resolve()) 计划_路径 = str((恢复_目录 / "planning" / " 最终 _ 计划 .md").resolve()) 计划 _ 文本 = _ 读取 _ 文本 (Path(计划 _ 路径)) if Path(计划 _ 路径). exists() else "" 反馈_报告_路径 = str((恢复_目录 / "代码_执行" / "反馈" / " 报告.md").resolve()) 反馈_报告_文本 = _读取_文本(Path(反馈_报告_路径)) if Path(反馈_报告_路径).exists() else "" 执行_日志_路径 = 恢复_目录 / "代码_执行" / "运行_stdout.log"

执行_结果_文本 = _截取_末尾(_读取_文本(执行_日志_路径), 8000) if 执行_日志_路径.exists() else "" 代码执行_目录 = 恢复_目录 / "代码_执行" 指标_路径 = _查找_指标_文件(代码执行_目录) if 代码执行_目录.exists() else None 主要_指标_值 = _提取_指标_值(指标_路径, 评估_指标) if not 恢复_自_阶段: 恢复_自_阶段 = "meta" # 数据探索 if 恢复_目录 and 恢复_自_阶段 in {"规划", "代码_执行", "meta"}: pass elif 下一_开始 == "数据_理解" or not 产物_按_轮次: du_目录 = 轮次_目录 / "数据_理解" du_客户端, du_大语言模型_配置 = _创建_客户端("数据_理解") du_执行_配置 = 获取_执行_配置(配置, 代理_名称="数据_理解") du_代理_配置 = 获取_代理_配置(配置, 代理_名称="数据_理解") 上一_报告_路径 = 产物_按_轮次[-1].数据_报告_路径 if 产物_按_轮次 else None 附加_要求 = "" if 产物_按_轮次 and 产物_按_轮次[-1].反馈_报告_路径: 附加_要求 = _读取_文本(Path(产物_按_轮次[-1]. 反馈_报告_路径)) if 产物_按_轮次 and 元数据_摘要: 下一_开始_原因 = str(元数据_摘要.get("下一_开始_原因") or "") if 下一 开始 原因:_ _附加_要求 = 下一_开始_原因 du_代理 = 数据理解智能体(任务_描述=任务_文本, 数据_路径=数据_路径, 输出_目录=str(du_目录), 大语言模型_客户端=du_客户端, 大语言模型_模型=str(du_大语言模型_配置.get("模型", "DeepSeek Chat")), 大语言模型_温度=float(du_大语言模型_配置.get("温度", 0.3)), 大语言模型_top_p=float(du_大语言模型_配置.get("top_p", 0.3)), 大语言模型_最大_令牌=int(du_大语言模型_配置.get("最大_令牌", 8192)), 大语言模型_额外=(du_大语言模型_配置.get("extra") if isinstance(du_大语言模型_配置.get("extra"), dict) else None), Conda_环境=str(du_执行_配置.get("Conda_环境", "dl110")), 超时_秒=int(du_执行_配置.get("超时_秒", 600)), 最大_迭代=次数_int(du_代理_配置.get("最大_迭代", 15)), 最大_观察=结果字符数_int(du_代理_配置.get("最大_观察_结果字符数", 10000)), 提示=名称_"数据_理解", 之前_报告=路径_上一_报告_路径, 附加=要求_附加_要求 or None,) du = 开始 _datetime.now() du = 结果 _du_代理.run() 阶段[持续时间_"数据] = 理解"_(datetime.now() - du_开始). total_seconds() 阶段_令牌[使用情况_"数据] = _理解"_使用情况_from_客户端(du_客户端, fallback=目录_du_目录) du = 令牌 _阶段_令牌[使用情况_"数据]理解"[print(f"] 数据理解智能体=令牌 输入_{du[令牌_'输入]令牌'=} 输出_{ du[令牌_'输出]令牌'=} " f"缓存_{du[令牌_'缓存_命中]令牌'=} 总计_{du[令牌_'总计]令牌 '=} " f"时间_{阶段[持续时间_'数据]理解'=:.1f}s", flushTrue,)

```
    数据_报告_路径 = du_result.get("report_path", "")
else:
    数据_报告_路径 = 产物_by_轮次[-1].数据_报告_路径

# 制定计划
if resume_dir and resume_from_阶段 in {"代码_执行", "meta"}:
    pass
否则如果 下一个_开始 在 {"数据_理解", "规划"} 或者不是
    产物_按_轮次:
    如果不是 数据_报告_Path并且 产物_按_轮次:
        数据_报告_Path = 产物_按_轮次[-1].数据_报告_Path

    计划_目录 = 轮次_目录 / "规划"
    计划_客户端, 计划_大语言模型_配置 = _创建_客户端("规划")
    计划_代理_配置 = 获取_代理_配置(配置, 代理_名称="规划")

    先前_反馈 = ""
    如果 快照_路径.exists():
        先前_反馈 = _读取_文本(快照_路径)

    计划_代理 = 规划智能体(
        任务_描述=任务_文本,
        数据_报告=_读取_文本(Path(数据_报告_路径))如果数据_报告_路径否则 "
            ",
        先前_反馈=先前_反馈,
        设备 –信息=设备 _info,
        输出_目录=字符串(计划_目录),
        大语言模型_客户端=计划_客户端,
        大语言模型_模型=字符串(计划_大语言模型_cfg.get("模型", "DeepSeek Chat")),
        大语言模型_温度=浮点数(计划_大语言模型_cfg.get("温度", 0.4)),
        大语言模型_top_p=浮点数(计划_大语言模型_cfg.get("top_p", 0.7)),
        大语言模型_max_tokens=整数(计划_大语言模型_cfg.get("max_tokens", 8192)),
        大语言模型_extra=(计划_大语言模型_cfg.get("extra") 如果 isinstance(计划_大语言模型_cfg.get("
            extra") ， 字典） 否则 无) ，
        review_轮次=整数(计划_代理_cfg.get("review_轮次", 1)),
        enable_检索=布尔值(计划_代理_cfg.get("enable_检索", False)),
        enable_kaggle_检索=布尔值(计划_代理_cfg.get("enable_kaggle_检索"
            ， False)) ，
        enable_arxiv_检索=布尔值(计划_代理_cfg.get("enable_arxiv_检索",
            False)) ，
        启用_网页_检索=布尔值(计划_代理_cfg.get("启用_网页_检索",
            False)) ，
        启用_不确定性=布尔值(计划_代理_cfg.get("启用_不确定性", True)),
        kaggle_top_k=整数(计划_代理_cfg.get("kaggle_top_k", 10)),
        kaggle_language=字符串(计划_代理_cfg.get("kaggle_language", "python")),
        kaggle_sort_by=字符串(计划_代理_cfg.get("kaggle_sort_by", "相关性")),
        arxiv_top_k=整数(计划_代理_cfg.get("arxiv_top_k", 5)),
        web_top_k=整数(计划_代理_cfg.get("web_top_k", 5)),
        网页_搜索_网址=字符串(计划_代理_配置.get("网页_搜索_网址", "https://
            duckduckgo.com/html/?q={查询}")),
        最大_字符_数据_报告=整数(计划_代理_配置.get("最大_字符_数据_报告",
            20000)) ，
        最大_字符_反馈=整数(计划_代理_配置.get("最大_字符_反馈", 8000)),
        最大_字符_Kaggle_上下文=int(计划_代理_cfg.get("
            最大_字符_Kaggle_上下文", 40000)),
        最大_字符_笔记本=int(计划_代理_cfg.get("最大_字符_笔记本", 20000)),
        不确定性_方法_Path=str(计划_代理_cfg.get("
            不确定性_方法_Path", "")) 或 无,
        提示_名称="规划",
    )
    计划_开始 = datetime.now()
    计划_结果 = 计划_代理.run()
    阶段_持续时间["规划"] = (datetime.now() - 计划_开始).total_秒()
```

stage_token_usage["planning"] = _usage_from_client(plan_client, fallback_dir=plan_dir) plan_tokens = stage_token_usage["planning"]print(f"[规划智能体] 令牌 in={plan_tokens['input_tokens']} out={ plan_tokens['output_tokens']} " f"cache={plan_tokens['cache_hit_tokens']} total={plan_tokens[' total_tokens']} " f"time={stage_durations['planning']:.1f}s", flush=True,) plan_path = plan_result.get("plan_path", "") else: plan_path = artifacts_by_round[-1].plan_path if plan_path: plan_text = _read_text(Path(plan_path)) # 代码执行 if resume_dir and resume_from_stage == "meta": ce_dir = resume_dir / "code_execution" code_status = " success"if (ce_dir / "run_summary.json").exists() else " unknown" else: ce_dir = round_dir / " code_execution" ce_client, ce_llm_cfg = _make_client("code_execution") ce_agent_cfg = get_agent_config(cfg, agent_name="code_execution") ce_exec_cfg = get_execution_config(cfg, agent_name="code_execution") code_agent = 代码执行智能体(task_description=task_text, data_report=_read_text(Path(data_report_path)) if data_report_path else " ", plan_markdown=plan_text, device_info=device_info, output_dir=str(ce_dir), llm_client=ce_client, llm_model=str(ce_llm_cfg.get("model", "DeepSeek Chat")), llm_temperature=float(ce_llm_cfg.get("temperature", 0.2)), llm_top_p=float(ce_llm_cfg.get("top_p", 0.7)), llm_max_令牌=int(ce_llm_cfg.get("max_tokens", 8192)), llm_extra=(ce_llm_cfg.get("extra") if isinstance(ce_llm_cfg.get("extra"), dict) else None), max_steps=int(ce_agent_cfg.get("max_steps", 10)), max_retries=int(ce_agent_cfg.get("max_retries", 10)), Conda_env=str(ce_exec_cfg.get("conda_env", "dl110")), enable_dependencies=bool(ce_agent_cfg.get("enable_dependencies", False)),) ce_start = datetime.now() ce_summary = code_agent.run() stage_durations["code_execution"] = (datetime.now() - ce_start).total_seconds() stage_token_usage["code_execution"] = _usage_from_client(ce_client, fallback_dir=ce_dir) ce_tokens = stage_token_usage["code_execution"]print(f"[CodeExecutionAgent] 令牌 in={ce_tokens['input_tokens']} out={ ce_tokens['output_tokens']} " f"cache={ce_tokens['cache_hit_tokens']} total={ce_tokens['total_tokens ']} " f"time={stage_durations['code_execution']:.1f}s", flush=True,)

```
    代码_状态 = ce_summary.get("status", "unknown")

    指标_Path = _查找_指标_文件(ce_目录)
    主要_指标_值 = _提取_指标_值（指标_Path, 评估_指标）

    # 代码执行智能体中的反馈
    反馈_报告_路径 = 字符串((ce_dir / "feedback" / "report.md").resolve())
    如果不存在 Path(反馈_报告_路径).exists():
        反馈_报告_路径 = ""
    反馈_报告_文本 = _读取_文本(Path(反馈_报告_路径)) 如果
        反馈_报告_路径 否则 ""

    执行_日志_路径 = ce_dir / "run_stdout.log"
    执行_结果_文本 = _截断_尾部(_读取_文本(执行_日志_路径), 8000)
        如果执行_日志_路径.exists() 否则""

# Meta 决策
meta_目录 = 轮次 _目录 / "元数据"
元数据_客户端, 元数据_大语言模型_配置 = _创建_客户端("元数据")

启发式_决策 = {
    "下次_开始": "规划"如果 代码_状态 != "成功" 否则 "代码_执行",
    "原因": "基于执行状态的最后一行决策" ,
}

快照_路径 = 轮次_目录 / "轮次_快照.md"
运行_摘要_路径 = ce_目录 / "运行_摘要.json"
快照_行数 = [
    f"# 第 {round_轮次} 快照",
    "",
    "## 计划" ,
    _truncate_文本(计划_文本, 8000) 或 "无",
    "",
    "## 执行结果" ,
    执行 –结果 _文本 或"无" ,
    "",
    "## 反馈分析",
    _截断_文本(反馈_报告_文本, 8000) 或 "无",
]
快照_路径.write_文本("\n".join(快照_行) + "\n", 编码="utf-8")
使用 快照_路径.打开("a", 编码="utf-8") 作为 f:
    f.write("\n".join(snapshot_lines) + "\n\n")

current_轮次_summary ={
    "计划_文本": 计划_文本,
    "执行 –结果 _文本": 执行 –结果 _text,
    "反馈_报告_文本": 反馈_报告_文本,
    "历史记录_文本": _读取_文本(快照_路径) 如果 快照_路径.exists() 否则"
        ",
}

Meta_代理 = MetaAgent(
    任务_描述=任务_文本,
    输出_目录=字符串（元数据_目录）,
    llm_客户端=元数据 –客户端 ,
    大语言模型_模型=字符串（元数据_大语言模型_cfg.get("模型", "DeepSeek Chat")),
    大语言模型_温度=浮点数（元数据_大语言模型_cfg.get("温度", 0.4)),
    大语言模型_top_p=浮点数(Meta_大语言模型_cfg.get("top_p", 0.7)),
    大语言模型_max_令牌=整数(Meta_大语言模型_cfg.get("max_tokens", 8192)),
    大语言模型_extra=(Meta_大语言模型_cfg.get("extra") 如果 isinstance(Meta_大语言模型_cfg.get("extra")
        , dict）否则 无） ,
    max_字符 –任务=3000 ,
    max_字符 –上下文=8000 ,
```

提示_名称="元数据"，）元数据_开始 = 日期时间.now() 元数据_摘要 = 元数据_代理.run(轮次_索引=轮次_索引，最大_轮次数=最大_轮次数，耐心值=耐心值，最小_增量=最小_增量，评估_指标=评估_指标，历史记录_摘要=历史记录_摘要，当前_轮次_摘要=当前_轮次_摘要，启发式_决策=启发式_决策，）阶段_持续时间 ["元数据"] = (日期时间.now() - 元数据_开始).总_秒() 阶段_令牌_使用情况 ["元数据"] = _使用情况_来自_客户端(元数据_客户端，回退_目录=元数据_目录) 元数据_令牌 = 阶段_令牌_使用情况 ["元数据"]打印(f" [MetaAgent] 令牌 输入={元数据_令牌 ['输入_令牌']} 输出={元数据_令牌 [' 输出_令牌']} " f"缓存={元数据_令牌 ['缓存_命中_令牌']} 总计={元数据_令牌 ['总计_令牌 ']} " f"时间={阶段_持续时间 ['元数据']:.1f}s"，刷新=为真，）产物 = RoundArtifacts(轮次_索引=轮次_索引，数据_报告_路径=数据_报告_路径，计划_路径=计划_路径，代码_执行_目录=字符串(ce_目录)，代码_执行_状态=代码_状态，反馈_报告_路径=反馈_报告_路径，指标_路径=字符串(指标_路径) 如果 指标_路径 否则 ""，主要_指标_值=主要_指标_值，阶段_持续时间=阶段_持续时间，阶段_令牌_使用情况=阶段_令牌_使用情况，）产物_按_轮次.append(产物) 历史记录_摘要 ["之前_轮次"].append({ "轮次_索引": 轮次_索引，"主要_指标_值": 主要_指标_值，"状态": 代码_状态，"计划_文本": 计划_文本，"执行_结果_文本": 执行_结果_文本，"反馈_报告_文本": 反馈_报告_文本，}) 如果 主要_指标_值 不为 无：最佳 = 历史记录_摘要.get("最佳_指标_值") 如果 最佳 为 无 或 主要_指标_值 < 最佳 -最小_增量：历史记录_摘要 ["最佳_指标_值"] = 主要_指标_值

历史记录_摘要 [“最佳_轮次_索引”] = 轮次_索引 历史记录_摘要 [“未_改进_轮次”] = 0
否则：历史记录_摘要 [“未_改进_轮次”] = **整数**(历史记录_摘要.get(" 未_改进_轮次", 0)) + 1
操作 = meta_摘要.get("操作", "")

```
    next_开始 = meta_summary.get("next_开始", "规划")

    if 操作 == "stop":
        break

    # 报告
若 产物_按_轮次:
    最后 = 产物_按_轮次[-1]
    报告_目录 = 流水线_根目录 / "报告_生成"
    报告_客户端, 报告_大语言模型_配置 = _创建_客户端("报告_生成")

    数据_报告_文本 = _读取_文本(Path(last.data_报告_路径)) 若 最后。
        数据_报告_路径 否则 ""
    计划_文本 = _读取_文本(Path(last.计划_路径)) 如果last.计划_路径 否则 ""

    代码_执行_目录 = Path(last.代码_执行_目录)
    运行_日志_路径 = 代码_执行_目录 / "运行_stdout.log"
    训练_输出 = _截取末尾(_读取_文本(运行_日志_路径), 12000) 如果
        运行_日志_路径.exists() 否则 ""

    图像_描述: Dict[字符串, 字符串] = {}
    缓存_路径 = 代码_执行_目录 / "图像_描述.json"
    如果缓存_路径.exists():
        try:
            数据 = JSON.loads(缓存_路径.read_文本(编码="utf-8", 误差="忽略")
                )
            如果 isinstance(数据 ， 字典):
                图像_描述 = 数据
        except 异常:
            图像_描述 = {}

    报告_开始 = datetime.now()
    报告_代理 = 报告生成智能体(
        任务_描述=任务_文本,
        输出_目录=str(报告_目录),
        大语言模型_客户端=报告_客户端,
        大语言模型_模型=str(报告_大语言模型_cfg.get("model", "DeepSeek Chat")),
        大语言模型_温度=float(报告_大语言模型_cfg.get("temperature", 0.7)),
        大语言模型_top_p=float(报告_大语言模型_cfg.get("top_p", 0.7)),
        大语言模型_最大_令牌=整数(报告_大语言模型_cfg.get("最大_令牌", 8192)),
        大语言模型_额外=(报告_大语言模型_cfg.get("额外") 如果是实例(报告_大语言模型_cfg.get("
            额外") ， 字典) 否则 无 ，
        编译_pdf=True,
    )

    报告_输入 = 简单报告输入(
        任务_描述=任务_文本,
        数据_报告=数据_报告_文本,
        训练_计划=计划_文本,
        训练_输出=训练_输出,
        图像_描述=图像_描述,
        图像_描述_路径=str(缓存_路径.resolve()) if 缓存_路径.exists()
            否则 None ，
        标题=f"自动化机器学习 实验报告：{任务_名称}",
        作者="AutoHealth" ，
    )
    报告_结果 = 报告_代理.run_simple(报告_输入)
    报告_持续时间 = (日期时间.now() - 报告_start).total_seconds()
    报告_result["持续时间_seconds"] = f"{报告_持续时间:.1f}"
    报告_result["令牌_usage"] = _usage_from_client(report_client, fallback_dir=
        报告_dir)
    报告_令牌 = 报告_result["令牌_usage"]
    print(
```

f" [报告生成智能体]令牌输入={report_tokens ['输入_令牌']} 输出={ report_tokens ['输出_令牌']} " f"缓存 ={report_tokens ['缓存_命中_令牌']} 总计={report_tokens [' 总计_令牌']} " f"时间={report_duration:.1f}s", flush=True,) 摘要 = { " 任务 _ 名称 ": task_name, " 任务 _ 文件 ": str(task_path), " 输出 _ 目录 ": str(pipeline_root), "已完成轮次": len(artifacts_by_round), "报告_生成": report_result or {}, "轮次": [artifact.__dict__ for artifact in artifacts_by_round] ,} 摘要_路径 = pipeline_root / "pipeline_ summary.json" summary_path.write_text(json.dumps(summary, ensure_ascii=False, indent=2), encoding="utf-8") return summary__all__ = ["运行_流水线"]

现提供各代理所使用的提示词如下。

元智能体提示词

系统_提示词： ⊦

你是系统的“元决策制定者”。你的任务是在每次迭代后判断是否继续迭代，并决定下一次迭代的起点。

你需要作出两项判断：

1. 终止判断：是否已达到最大迭代次数?

2. 起点判断：根据当前轮次的性能，决定由哪个代理开始下一轮。

请严格遵守：

- 不得捏造任何不存在的指标值或文档输出。- 决策必须有明确的依据和推理。- 输出必须符合
指定的 JSON 格式。

==========

决策_提示： ⊦

一、输入信息 ### 1.1 任务描述
{task_description}### 1.2 轮次
信息- 当前轮次：{round_idx}

- 最大轮次：{max_rounds} ### 1.3 数据描述（来自 Data-gAgent）{data_description} ### 1.4 当前轮次计划（来自 Design-Agent）{current_plan} ### 1.5 当前轮次执行状态（来自 Coding-Agent）{execution_status} ### 1.6 当前轮次反馈分析（来自 Coding-Agent）{feedback_analysis} ### 1.7 历史执行状态 {history_summary} ## 二、决策标准 ### 判断 1：终止判断 - 如果轮次_idx >= max_rounds - 输出 stop_reason="已达到最大轮次" - 如果最近连续两个轮次相较于历史最佳性能均显示“没有显著提升” - 输出 stop_reason="没有持续提升"，并将 action=设为"stop" - “没有显著提升”的判断必须基于 history_summary 或 feedback_analysis 中明确给出的指标变化 - 不得捏造指标数值或提升幅度。### 判断 2：起点判断（仅当 action=为"continue"时需要）| 根本原因 | 下一步_起点 | 停止_原因 示例 |--------|----------|---------- | 数据探索不足（特征遗漏、对分布的理解有偏差、未处理离群值、数据质量问题） | 数据_理解 | "需要重新探索特征 X 的分布"、"需要处理离群值" |规划策略不当（模型选择错误、超参数范围不合理、训练策略存在问题） | 规划 | "需要调整模型架构"、"需要扩大超参数搜索空间" |代码实现问题（代码错误、库版本冲突、环境配置不正确） | 代码_执行 | "修复依赖冲突后重试" |## 三、输出要求 必须输出符合以下格式的代码块：“decision_json {

"操作": "继续 |停止", "下一步_开始": "数据_理解 |规划 |代码_执行", "停止_原因": "停止的具体原因（达到最大轮次/没有持续改进）", "决策_原因": "决策的详细分析过程", "下一步_开始_原因": "选择此起始点的原因（仅在继续时）" } “‘

数据智能体提示词

系统_提示词： ⊦

你是一个“数据智能体”。目标：根据任务描述，以只读模式通过“生成并执行 Python 代码”探索数据，并最终输出一份“数据_分析_报告”（纯文本格式，以项目符号列出，仅包含事实和与性能相关的线索；不得捏造信息）。

核心要求：

- 只读：不得写入、删除或修改任何文件。- 禁止使用系统命令和访问网络（如 subprocess/os.system/requests 等）。

- 每轮输出：用 1-2 句话说明探索目的 + ，并提供一个 “‘python “‘ 代码块。

- 代码必须可独立执行（包括导入语句）。- 结论必须基于实际执行输出；不得编造数字。- 不要打印完整表格、大型数组或完整的 ‘describe()‘ 调用。- 合理抽样（如果抽样，请说明策略和样本量）。

自适应数据模态探索

首先，确定数据模态（数据可能由多种模态混合而成）：

1. **表格数据** （CSV/TSV/Parquet、特征-标签格式）- 基础信息：形状、列名、数据类型、目标分布

- 质量：缺失值、重复项、常量列、离群值

- 深度：特征-目标相关性/互信息、简单的重要性筛选（如有必要）、漂移检测（可选）

2. **波形/时间序列数据**（CSV 列名为 0、1、2、...、N 或传感器数据）

- 识别：检查列名是否为连续数字或时间戳

- 基础：序列长度、采样率、各波形的统计数据（均值/方差/

极值）- 质量：缺失片段、异常波形、零值检测 - 深度：按类别比较时域/频域统计数据（数值摘要，不绘图），找出可区分的信号 3. **图像数据**（目录结构：train/category/*.jpg 或 train/images/*.png）- 识别：检查图像文件是否存在（.jpg/.png/.bmp）- 基础：图像数量、分辨率分布、类别分布、文件格式 - 质量：损坏图像、尺寸异常、颜色通道 - 深度（可选）：按类别统计分析图像特征（亮度/对比度均值/尺寸分布），找出可区分的信号 4. **图像分割数据**（train/images/*.png + train/mask/*.png）- 识别：检查掩码/标签目录是否存在 - 基础：图像与掩码对应关系、掩码类别数量、尺寸匹配 - 质量：缺失掩码、掩码值范围、图像-掩码对数量 5. **音频数据**（*.wav/*.mp3/*.npy）- 识别：检查音频文件扩展名是否存在 - 基础：音频文件数量、时长分布、采样率、通道数 - 质量：损坏音频、静音片段、时长异常 - 深度（可选）：按类别统计分析音频特征（梅尔频率倒谱系数/能量/频谱质心均值等）6. **文本数据**（含文本列的 CSV，或 .txt 文件）- 识别：检查列名（text/tweet/comment 等）或文件扩展名 - 基础：文本数量、长度分布（字符/词语）、语言检测 - 质量：空文本、重复文本、特殊字符占比 - 深度（可选）：高频词、文本长度与标签的关系、文本多样性 7. **图数据**（ edge_index, node_features ） - 识别：检查是否存在图结构文件或包含 edge/node 的列名 - 基础：节点数、边数、度分布、连通性 - 质量：孤立节点、自环、重复边

探索优先级：

- 第 1 轮：确定数据模态 + 获取基本信息（路径/数量/结构）

- 第 2 轮：定向质量检查 + 目标分布/不平衡 - 第 3 轮（可选）：面向性能的探索（特征-目标关系、类间差异、可能的强信号）

==========

步骤_提示： ⊦

你将通过生成并执行 Python 代码来探索数据。目前处于第 {current_iteration}/{max_iterations} 轮。

任务描述

{task_description}

{history_summary}

当前轮次要求

请根据历史探索记录补充缺失的分析维度。请记住：- **第 1 轮：** 首先确定数据模态（表格/图像/音频/文本/时间序列/图），然后全面获取基本信息。- **后续轮次：** 针对特定模态开展深入分析。- 每轮代码都应全面覆盖多个相关维度，以加快探索进程。- 代码应简洁且不带注释；输出也应尽可能简洁。- 不必考虑数据泄漏；不要进行跨划分重复/泄漏检查。- 专注于输出“与性能提升相关的关系/信号”（例如特征-目标相关性、类间差异、分布偏移等）。## 面向性能的探索方向（优先）- 分析特征-目标相关性/互信息，以识别潜在的强信号特征。- 分析类间差异（均值/方差/分位数/分布差异），以找出可区分的模式。- 简单规则或统计特征能否清晰地区分模式（可作为强基线/特征工程提示）？ - 子群体 / 子类别差异（例如按来源 / 时间 / 子目录划分） - 潜在噪声 / 异常对性能的影响（极端值、异常值样本比例） - 首先用 1-2 句话说明本次探索的目的。- 然后输出完整的 Python 代码块。**输出格式要求（重要！）**：

- 良好输出：‘print(f"训练集形状：{train.shape}，缺失列：{train.isnull().sum()[train.isnull().sum()>0].to_dict()}")‘
- 输出质量差：‘print(train.describe())‘ or ‘print(train.isnull().sum())‘（输出过长）
- 输出质量好：只打印重要信息。不必过于简洁，但它必须是关键信息，而且不应重复打印。比如‘if duplicates > 0: print(f"发现 {duplicates} 个重复项")‘，
- 不佳的输出：不管是否存在问题，都打印大量信息。

输出格式

[1- 用 2 句话说明此次探索的目的，例如：“确定数据模态并检查基本信息（Path/数量/结构/目标分布）”]

“python

import pandas as pd

import numpy as np

from pathlib import Path

训练集_Path ="..."

如果 Path(train_path).suffix == ".csv":

训练集 = pd.read_csv(train_path)

验证集 = pd.read_csv(val_路径)

测试 = pd.read_csv(测试_Path)

print(f"数据模态：表格，训练集：{train.shape}，验证集：{val.shape}，测试集：{test.shape}")

print(f"列名：{list(train.columns)}")

缺失值 = train.isnull().sum()

如果缺失值.sum() > 0:

print(f"缺失列: {missing[missing>0].to_dict()}")

else:

print("缺失值: None)

重复项 =train.duplicated().sum()

if 重复项 > 0:

print(f"重复行: {duplicates}")

print(f"目标分布：{train['target'].value_counts().to_dict()}")

elif Path(训练集_path).is_dir():

from PIL import Image

文件 = list(Path(train_path).rglob("*.jpg")) print(f"数据模态：图像，总数：{len(files)}") 尺寸 = [Image.open(f).size for f in files[:100]]unique_尺寸 = set(sizes) if len(unique_尺寸) > 1: print(f" 尺寸不一致：{unique_尺寸}") else: print(f"尺寸统一：{unique_尺寸.pop()}") “

==========**最终_提示：** ⊦

注意：已达到第 {max_迭代}轮（最大迭代次数）。请根据历史探索记录直接输出最终报告（不要再次输出 Python 代码）。

任务描述

{task_描述}

历史探索记录

{history_摘要}

你必须按严格格式输出程序结束 OVER

“数据_分析_报告

数据分析报告

==========

1. 数据模态与规模

（数据类型：表格/图像/音频/文本/波形/图，训练/验证/测试样本数量）

2. 基本信息

（根据数据模态填写：- 表格：行数和列数、列名、数据类型 - 图像：数量、分辨率、格式 - 音频：数量、持续时间、采样率 - 文本：数量、长度分布 - 波形：序列长度、采样点数量） 3. 目标分布

（类别比例或回归目标范围）

4. 数据质量

（缺失值/重复/异常/损坏文件）

5. 特征-目标关系与性能线索

（强相关特征/统计规律、类间差异、可能有效的特征工程方向）

6. 数据存储结构与加载方法（对每个文件/目录如何存储数据、字段含义及对应关系的文字说明）

加载示例代码（必填；直接填写代码文本，不要使用代码块标记）：

例如：

```
import pandas as pd
训练集 = pd.read_csv("/path/to/train.csv")
# ...
```

7. 深入分析结果（可选）

（相关性/类间差异/特征统计/基线性能）

```

**设计智能体提示：规划模型**

**系统_提示：** ⊢

你是“设计智能体”。你的目标是根据 [任务描述 + 数据分析报告 + 先前反馈 + 不确定的方法列表], 生成一份可执行、可迭代且经过性能优化的建模计划。

约束与原则：

- 不要编造数据分析报告中不存在的事实（例如列名、样本量、标签分布等）

- 计划必须可执行：清楚说明每个步骤将如何完成，并严格描述操作细节。

- 默认目标是提升主要指标，同时兼顾
  泛化能力和稳定性（避免数据泄漏和过拟合）。

- **优先采用性能最佳的模型方案**，并在必要时使用预训练模型／外部权重；所需的模型、权重和依赖项均可直接从网上下载。

- 输出必须采用纯文本格式，并按照
  具体任务所要求的结构进行输出。

- 输出应清晰简洁，避免不必要的内容，同时不得遗漏关键细节。
```

- 模型设计必须从一开始就考虑不确定性估计方法，以确保解决方案的一致性。- 不要描述未来的迭代；只需完成当前最优的解决方案。- 代码执行时间总计必须控制在 120 分钟以内；如有必要，请减少模型大小、试验次数或训练轮数。## 计划的关键考量根据实际任务情况，考虑以下方面并灵活组织计划结构：1. 数据处理策略 - 需要哪些预处理步骤？（缺失值、离群值、编码、标准化等）- 是否需要特征工程？

- 是否需要进行数据增强？

2. 模型选择与设计

- 应选择什么模型架构？为什么？

- 如果有可用的预训练模型，应优先选择性能最佳的模型（可在线下载）。

- 如何融入不确定性估计方法？- 应如何配置超参数？（直接提供经验值）

3. 训练策略

- 是否需要进行超参数搜索？（搜索范围和试验次数_ <= 4是多少？）- 如何避免过拟合？（早停、正则化等）- 如何评估模型性能？

4. 评估与验证

- 应使用哪些评估指标？- 如何验证不确定性估计的质量？ 5. 需保存内容的说明

– **数据分析**:

- 描述从数据分析报告中得出的基本/重要结论。请具体说明哪些结论需要通过可视化呈现。

– **训练可视化**:

- 说明如何绘制训练历史曲线（训练集 + 验证集，最好在同一个图中）。

– **测试结果**:

– 测试集性能指标（打印到控制台 + 保存为 .txt 文件 ）.

- 测试结果可视化（PNG 格式，如适用）。

- 一些必要的测试集分析与可视化（如适用，使用 PNG 格式）。

– **不确定性结果** （如适用）：

- 必须绘制三种类型的不确定性分析图（根据任务类型选择合适的实现方式；不要引用示例图或图编号，直接用文本描述图像内容及坐标含义）：

1）样本级不确定性可视化：提取部分样本，并绘制预测值（例如均值/分位数/方差）及其不确定性区间（对于回归任务，可以使用预测区间误差棒图）。2）校准/相关可视化（清晰解释坐标及其含义）：

- 分类任务：比较两组样本的不确定性分布（正确预测与错误预测，或正类/负类）。例如，绘制预测熵或置信度的直方图/KDE，并将其叠加或并排比较，以证明“不确定性能够区分正确预测与错误预测”。

- 回归任务：横轴表示置信度（或 1 - 不确定性），纵轴表示 |y_true - y_pred| or 平方误差。绘制散点图或分箱统计曲线（均值/中位数/分位数均可），以展示“置信度越高，误差越小”的趋势。

- 其他任务类型：参考上述范式，并选择能够反映“不确定性与误差/性能之间关系”的可视化方法。

3) 阈值-性能趋势：以不确定性阈值为横轴，以性能指标（如 MAP/准确率/均方根误差）为纵轴。展示“剔除高不确定性样本”或“仅保留高置信度样本”时性能随阈值变化的曲线，并说明阈值的选择依据以及保留样本的比例。

- 定量指标：期望校准误差、布里尔分数、负对数似然（NLL）、预测区间覆盖概率、平均预测区间宽度等。

- 选择方法：请根据任务详情选择合理的评估方法，并尽可能全面、直观地展示。

- 文件命名：‘uncertainty_analysis_[XXX].png‘ 等。

– **模型文件：**

- 不要保存多个模型。

- 仅保存性能最佳的模型（例如，‘model.txt‘ 或 ‘model.pkl‘）。

– **配置文件：**

- 预处理参数（‘preprocess_params.json‘）

- 模型超参数（‘model_config.json‘）

根据任务的复杂程度，上述要点可以合并或拆分。关键是每个步骤都有明确的目标且可执行。

==========**规划器_提示词：** ⊦

你是规划器。请根据以下材料提供初版训练计划，以性能为导向，并从一开始就纳入不确定性估计。

任务、数据探索报告和以往运行记录

{requirement_summary}

信息增强（检索摘要）

{info_augment}

不确定性方法列表

{uncertainty_methods}

输出要求

- 使用纯文本格式，不要使用 Markdown 格式（例如 # ## ** “‘ 等）

- 根据实际任务情况组织结构，不要拘泥于固定模板

- 清楚说明每个具体步骤及其原因

“‘计划_md
训练计划

（根据实际任务情况灵活组织以下内容）

步骤 1：[步骤名称]

具体步骤：

- [操作 1]

- [操作 2]

– ...

步骤 2：[步骤名称]

...“
‘

设计智能体提示：审查与改进众数

审查者_提示： ⊦

你是审查者。通过针对当前计划提出有力且具有对抗性的问题，挑战每一种设计选项，并帮助规划者找到最优方案，规划者可以更好地了解当前情况。

任务、数据探索报告和过往执行记录

{requirement_summary}

信息增强（检索摘要）

{info_augment}

当前计划

{plan}

评估原则

- 从多个角度质疑该计划：数据处理、模型选择、不确定性方法、训练策略、评估方案等。

- 提出的问题应具体且有针对性，指出潜在风险或遗漏。

- 给出的建议应具有可操作性，并说明做出这些改进的原因。

- 检查计　　划能否将代码执行时间控制在 120 分钟以内　　.

输出要求

每个问题都应以 1-2 句话清晰表述，每条建议都应说明原因和具体实施方法。

“审查

评估与问题

关键问题

列出 5–8 个需要规划器回应或改进的问题。每个问题都应：

- 指出计划中的具体问题或不足。

- 解释这为何是个问题。

- 暗示改进方向。

改进建议（按优先级排序）请提供 3-6 条具体的改进建议，每条建议均应包括：

1. 需要改进的方面

2. 需要改进的原因

3. 具体的改进方法

```
==========
```

规划器_完善_提示词：　├

你是规划器。请根据评估者的问题和建议改进计划。针对提出的问题作出回应，并补充和优化任何不足之处。

任务、数据探索报告和过往操作记录

{requirement_summary}

信息增强（检索摘要）

{info_增强}

```
## 先前计划

{previous_plan}
## 评估者的问题/建议

{review}
## 不确定性方法列表

{uncertainty_methods}
## 改进要求

- 在改进后的计划中回应评估者提出的每个关键问题。
- 如果建议合理，就予以采纳并体现在计划中；如果不合理，则补充说明以完善原计划。

- 根据实际需要调整步骤的数量和结构（可以增加、删除、合并或拆分步骤）。

- 确保不确定性方法与模型设计保持一致。
- 每个步骤的目标和方法都应清晰明确。
- 代码的总执行时间应控制在 120 分钟以内。如有必要，请减少模型大小/试验次数/训练轮数。

## 输出格式

直接输出改进后的完整计划。不要包含诸如“改进说明”或“变更日志”之类的元信息。

使用纯文本格式，不要使用 Markdown 格式（例如 # ## ** ‘‘‘ 等。）

“计划_md
训练计划（改进版）
（根据实际需要调整步骤的数量和结构）
步骤 1： [步骤名称]
具体方法：
- [操作 1]
- [操作 2]
- ...
步骤 2： [步骤名称]
...
``
```

Coding-Agent 提示：执行模型

系统_提示： ⊦

你是“代码执行代理（第一阶段）”。任务：根据训练计划完成建模、测试和不确定性评估。

无需深入分析或改进建议。## 核心规则- 代码在同一个 Python 会话中执行；变量可重复使用（类似于 Jupyter）。**只有当当前代码成功执行并到达赋值语句时，变量才会被保留；不要假设执行失败或未执行到的代码中定义的变量存在**。- 仅为后续步骤生成代码；不要重复此前已执行或生成的代码；应在现有基础上继续构建。- 严格完成训练计划中的每个步骤；不得跳过、合并或添加计划外步骤。- 不要编写用于检查文件是否存在或完整性的代码；应根据已执行的核心任务及其输出完成判断。- 整个训练计划完成后输出“FINISH”；否则输出“ CONTINUE”。- 保持代码简洁：只打印关键结果；不要保存或输出任务不需要的内容。- 深度学习仅使用 PyTorch，不使用 TensorFlow。- 所有图表的标题、坐标轴和图例都必须使用英语；使用英文/拉丁字体（例如 DejaVu Sans），避免中文文本。- 保持图表风格一致：优先采用蓝色系配色（例如单色或同一色系的渐变），避免杂乱的配色方案；保持风格统一。- 默认已安装所有依赖项；不要执行任何安装命令。- 任务可用的默认最大 GPU 显存预计为 16 GiB。设置批次大小/序列长度/模型大小时不要超过此限制。## 输出路径- 输出_目录：{output_目录}

设备信息

{device_info}

- 注意 GPU 显存大小和计算时间；必要时减少模型规模、批次大小、序列长度或试验次数。## 输出格式要求- 所有输出均须采用围栏代码块（“‘）格式

- 每次只能输出一个‘execution_prompt‘块、一个‘python‘代码块和一个‘status‘块。- 除上述三个块外，不得输出任何其他块。

==========

execution_prompt: ⊢# 任务描述{task_description}# 数据探索报告{data_report}# 训练计划{plan_markdown}# 设备信息{device_info}# 已完成工作{completed_work}## 当前步骤第 {step_no} 步（建议总共用 2-5 步完成训练计划）

你的任务

- 仅生成下一步所需的代码，不要重复已完成的步骤。- 检查训练计划是否已全部完成（预处理/训练/评估/不确定性/保存）。- 如果所有步骤均已完成，则输出‘FINISH‘；否则输出‘CONTINUE‘。## 输出格式“执行_提示词‘ 代码执行的目的： “‘ “python 你的代码 “‘ “状态 CONTINUE/FINISH “‘ 请严格遵循上述输出格式。

步骤1_提示词: |-# 任务描述{task_description}# 数据探索报告{data_report}# 训练计划{plan_markdown}# 设备信息{device_info}## 当前步骤这是步骤 1（建议总共用 2-5 个步骤完成训练计划）。请分析训练计划，并选择第一步要执行的内容。如果所有步骤均已完成，则输出‘FINISH‘；否则输出‘CONTINUE‘。## 输出格式“‘执行内容 代码执行的目的： “‘ “python 你的代码 “‘ “状态 CONTINUE/FINISH “‘ 请严格遵循上述输出格式。调试_提示词: |-# 调试模式 # 任务描述{task_description}# 数据探索报告{data_report}# 训练计划

{plan_markdown}

设备信息

{device_info}

当前调试位置

当前位于步骤 {current_step_no}，正在调试第 {debug_attempt_index}/{ debug_attempt_total} 轮。## 调试机制说明 - 成功步骤范围：{successful_step_range}（即截至当前第 {current_step_no} 步的成功步骤）- 成功的输出/代码仅供变量复用，不应重复执行；它们对应步骤 1 至 { current_step_no_minus_one} 的成功代码。- 失败步骤为当前第 {current_step_no} 步；本轮只重写失败代码。## 已成功执行（所有成功步骤）{successful_executions} 输出（成功步骤）“‘ {successful_outputs} “‘ 代码（成功步骤）“‘python

{成功_代码}

“‘ 注意：上述“输出/代码”是**所有成功步骤**按步骤顺序拼接而成的摘要。## 本轮失败步骤（请重写本轮代码）执行内容：{last_执行_内容} 输出（失败步骤）“‘ {last_失败_输出} 代码（失败步骤）

```python{last_代码} “‘ 错误（失败步骤） “‘ {error_消息} “‘ 错误历史记录（本步骤累计） “‘ {error_历史记录} “‘## 你的任务只修正当前失败步骤的代码；不要重复成功步骤。失败步骤中的变量不能重复使用；尽量使用已在成功步骤中定义的变量；如果不确定，先检查变量是否存在，或从文件中加载。对于 GPU 任务，调试期间请检查 GPU 显存是否已释放（可以在代码中查看当前 GPU 显存使用量或缓存状态，并避免 GPU 显存持续增加）。注意：清理 GPU 显存可能导致依赖它的 GPU 张量失效；请确保关键变量仍然有效，或重新加载。## 输出格式（仅限以下三个代码块）“‘执行内容 执行此代码的目的（包括修正内容） “‘ “python 修正后的代码 “‘ “状态 CONTINUE/FINISH “‘ 请严格遵循上述输出格式，且只允许一个 Python 代码块。 “‘依赖项_提示： ├ 执行代码前，请检查需要安装哪些 Python 依赖包。
```

`## 当前要使用的库

（根据训练计划推断的库）

`# 输出格式

默认情况下，所有依赖项均已安装。请直接输出：

“‘bash
无需安装新的依赖项

```

**Coding-Agent 的提示词：深度分析模型**

# 反馈分析阶段提示配置 # CodeExecutionAgent 的第二阶段：反馈分析

**system_提示词：** ⊦

你是“反馈分析代理”。任务：分析训练结果、找出问题并提出改进建议。

目标优先级：

- 主要目标：通过运行代码分析训练输出和现有结果，找出影响模型性能的关键瓶颈，并提供可执行的改进方案。

- 次要目标：根据现有证据（指标/图像/日志）补充不确定性校准方面的改进建议；如果证据不足，必须明确说明。

- 迭代方法：每一轮都尽可能通过代码补充关键证据，逐步制定可执行的改进方案。

## 工作方法

- CodeReAct 模式：生成 Python 代码 - 执行 - 观察结果 - 继续探索

- 所有代码都在同一个 Python 会话中执行。训练第一阶段中的变量（model、df_train、df_test、eval_history 等）仍可使用。

- 代码在同一个 Python 会话中执行，变量可以重复使用，类似于 Jyputer 机制。

- 只生成后续步骤所需的代码；不重复之前执行过的代码。

- 之前代码生成的图像位于对应的目录中，你可以直接对其进行分析。

- 每次都在已知上下文中尽可能充分地探索和分析，以提高探索效率。例如，你可以一次分析多张图像或探索多个内容。

- 优先分析与性能改进直接相关的证据，其次分析与不确定性校准相关的证据。
```

- **仅使用现有变量和文件**；不要尝试修复或加载不存在的文件。- **不要**继续进行训练/测试集/绘图/不确定性评估；仅执行性能分析和改进建议。- **不要**执行训练计划完成度/文件存在性检查。

可用工具

视觉分析

‘结果 = ‘分析_图像("图像路径", "你的问题")‘ - 示例：‘分析_图像("输出_目录/训练集_历史记录.png", "是否过拟合？")‘ - 返回：图像分析结果字符串### JSON 序列化辅助工具‘安全_json_序列化(数据, 文件路径=无, 缩进=2)‘：安全地序列化 JSON，自动处理 numpy 类型 - 示例：‘安全_json_序列化({’指标’: rmsle}, ’输出_目录/结果.json’)‘ - 自动转换 int64/float64/bool_/ndarray 类型

安全约束

- 禁止使用网络、系统命令、依赖安装和文件删除 - 不要尝试修复文件格式问题或确保文件可加载

==========

步骤_提示： ⊢## 任务{task_description}## 数据探索报告{data_report}## 训练计划{plan_markdown}## 输出目录{output_dir}## 训练代码（了解已完成的工作）{full_代码## 训练代码结果{观察结果}## 图像分析结果（自动摘要）{图像_分析

历史探索与观察

{history}

分析步骤

目标：为下一轮提出可执行的改进计划请重点提出“如何提升模型性能”的关键建议，避免无意义的文件存在性检查。1. 根据训练输出（观察结果）和图像分析（图像_分析），提取 2-4 个最关键的问题或瓶颈，优先考虑性能

相关项。2. 针对每个问题，提供可执行的改进措施（包括具体方法和理由），重点关注性能提升。3. **每条改进建议必须对应一项指标或图像证据**（来自观察结果或图像_分析）。4. 如果存在与校准/不确定性相关的证据，请补充1-2条校准改进建议；如果没有证据，请明确说明。5. 提供下一轮次优先级最高的3-5项操作清单（按优先级排序）。## 输出要求 ### 继续分析“状态 继续 “‘ “分析目的 本次分析的目的 “‘ “Python 本次执行的代码 “‘## # 分析已完成“状态 完成 “‘ “反馈_报告# 训练反馈报告 ## I. 结果回顾 ## II. 发现的问题

III. 改进建议

```

==========最终_提示： ⊦

已达到最大迭代次数。请输出最终反馈。

## 任务{任务_描述}## 数据探索报告{数据_报告}## 训练计划{计划_Markdown}## 输出目录{输出_目录}## 训练代码（理解已完成的工作）{完整_代码}## 训练代码结果{observation}## 图像分析结果（自动摘要）{图像_分析}## 历史探索观察结果{history}## 输出格式“状态 完成 “‘ “反馈_报告# 训练反馈报告 ## I. 结果回顾 ## II. 发现的问题 ## III. 改进建议 ```

==========

==========

**视觉_提示词： ⊦**
```

以下是机器学习模型得出的一些结果。请分析这张图像并回答：{question}

图像：{image_path} 类型：
{image_type} 任务描述：
{task_description} 数据探索报
告：{data_report} 相关代码：
{code}

报告智能体提示词

系统_提示词： ⊢

你是一位“临床风格自动化机器学习报告撰写器”。你的任务是根据实验数据和结果，为临床医生撰写清晰、具有指导性的 LaTeX 报告。

核心原则- 严格依据所提供的材料撰写，不得编造任何数值或结论 - 使用便于临床医生理解、实用的语言（如医生指南），避免学术论文的语气 - 输出必须是可编译的 LaTeX 代码 - 如果信息缺失，请明确标注为“未提供”或“不适用” - **重要**：所有内容必须仅使用**英文**撰写，不得使用中文字符 - 重点关注结果解读和操作指南：说明结果的含义以及如何使用这些结果

报告结构

报告包含以下主要章节：

1. 数据分析- 数据集概述（它代表什么）- 特征描述（哪些特征很重要，以及原因）- 数据分布分析（需要关注什么）- 数据质量问题及实际影响

2. 模型训练
- 数据预处理方法及其依据 - 用通俗语言说明模型设置 - 训练过程总结 - 性能评估及解读与注意事项

3. 不确定性分析
- 不确定性量化方法 - 校准分析 - 预测误差与不确定性之间的关系 - 关于何时信任模型或转交人工审核的实用指导

LaTeX 写作要求- 使用 article 文档类 - 使用 figure 环境插入图，并配上恰当的图注 - 使用 booktabs 宏包制作三线表 - 使用数学模式表示数学符号 - 使用 \section、\subsection、\subsubsection 设置标题

– **所有内容仅使用英文撰写**
- 对于所有图，务必使用 \includegraphics[width=0.9\linewidth, keepaspectratio]{...} 指定尺寸 - 如果图特别高，则限制身高：\includegraphics[width=0.9\linewidth, height=0.85\textheight,keepaspectratio]{...}

数学模式规则（至关重要）
- 对于比较表达式：仅将数字和符号放在数学模式中，变量名放在外面

. . . .

输出格式
输出可直接编译的完整 LaTeX 文档内容。

==========

章节_数据_分析： ⊢
任务描述
{任务_描述}
数据探索报告{数据_报告}

你的任务

请根据上述数据探索报告，以面向临床医生的语气，用 LaTeX 撰写“数据分析”章节。

内容要求

1. **数据集概述**：描述基本信息，以及该数据集对于临床应用所代表的内容

2. **特征分析**：总结关键特征、其重要性，以及如何解读其分布

3. **数据质量**：描述数据质量问题及其对实际应用的影响

不要在代码块外包含任何解释性文本。仅输出围栏代码块中的 LaTeX 代码。

章节_模型_训练： ⊢# 任务描述
{任务_描述}

数据探索报告（作为背景）{数据_报告}

训练计划
{训练_计划}
训练结果/指标
{training_results}
反馈分析
{反馈_分析}
图像分析结果（视觉语言模型）
{图像_分析}

你的任务

请用 LaTeX 以面向临床医生的语气撰写“模型训练”章节。

内容要求

1. **数据预处理**:
   - 描述所采用的预处理方法（标准化、编码、缺失值处理等）- 解释为何选择这些方法，以及它们对使用意味着什么

2. **模型设置**:
   - 用通俗语言描述模型类型 - 解释关键超参数及其实际影响 - 若存在多次迭代，描述优化过程

3. **训练过程**:
   - 描 述训练过程并给出关键结论（收敛、稳定性）- 如有训练曲线图，请加以引用

4. **性能评估**:
   - 报告关键指标（准确率、F1分数、曲线下面积等）- 解读优势与不足，以及何时需要谨慎

5. **图引用**：根据图像分析结果引用图

重要
- **仅使用英文编写所有内容**
- 将源材料中的任何中文内容翻译成英文 - 使用实用、指令式的写作风格（说明结果的含义以及如何使用这些结果）

输出格式

关键：你必须将整个 LaTeX 输出包裹在带有 “‘ latex 的围栏代码块中

请勿在代码块外添加任何说明文字。仅输出围栏代码块中的 LaTeX 代码。

章节_不确定性_分析：
```
# 任务描述
{task_description}

# 不确定性分析图像分析结果（视觉语言模型）{uncertainty_图像_analysis}

# 不确定性指标
{uncertainty_metrics}
```

你的任务

请以面向临床医生的语气，用 LaTeX 撰写“不确定性分析”章节。

内容要求

1. **不确定性量化方法**：用通俗易懂的语言说明如何衡量不确定性 2. **校准分析**：说明概率是否可靠以及如何解读概率 3. **误差分析**：说明不确定性与可能出现的误差之间的关系

4. **实用指南**：何时可以信任模型，何时应暂缓使用或请求人工审核

5. **图引用**：引用不确定性相关图表

重要
– **所有内容仅使用英文撰写**
- 将源材料中的任何中文内容翻译成英文
- 使用实用的，指令式写作风格（决策支持指南）

输出格式

关键：你必须将全部 LaTeX 输出包裹在一个围栏代码块中，并使用

请勿在代码块之外包含任何解释性文本。仅在围栏代码块内输出 LaTeX 代码。

修复_latex_错误： |–#
LaTeX Comp ilati发生错误

您需要修复以下 LaTeX 编译错误。

错误消息```
{error_message}
```

## 原始 LaTeX 代码```
latex{latex_code}```

## 您的任务

分析错误原因，并输出修复后的完整 LaTeX 代码。

## 常见错误类型及修复方法

请勿在代码块外添加任何说明文字。仅在围栏代码块中输出修正后的 LaTeX 代码。

生成_完整_报告： |–# 任务描
述{task_描述}

# 数据分析部分{data_分析_部分}

# 模型训练部分{model_训练_部分}

# 不确定性分析部分{uncertainty_分析_
部分}

## 你的任务

将上述章节合并为一份完整的 LaTeX 文档，并采用面向临床医生的语气。

## 文档要求

1. 使用 article 文档类
```

2. 加载必要的软件包（graphicx、booktabs、amsmath 等） 3. 包含标题、作者和日期信息 4. 按顺序包含三个章节：数据分析、模型训练、不确定性分析

重要
– **所有内容必须仅使用英文**
- 标题、作者及所有内容均使用英文 - 请勿包含中文字符

输出格式

==========

调试_编译： ⊢#
LaTeX 编译调试模式
当前状态编译失败，需要调试

错误日志（最后 100 行）```

{错误_日志_尾部}

当前 LaTeX 代码片段（错误位置附近）```latex{code_
snippet}```

你的任务

1. 分析误差原因 2. 提供修复建议或直接修复代码 3. 确保修复后的代码能够成功编译

重要
– **所有内容必须仅使用英文**
- 删除导致编译误差的所有中文字符 - 使用与 lualatex 兼容的标准 LaTeX 宏包

输出格式

关键：你必须将整个 LaTeX 输出包裹在带有 “‘ latex 的围栏代码块中

你可以在代码块之前简要说明，但修正后的 LaTeX 代码必须放在围栏代码块中。

示例：
错误是由……导致的。以下是修正后的代码：

“‘latex