HealthFlow: A Self-Evolving AI Agent with Meta Planning for Autonomous Healthcare Research

Yinghao Zhu[1,2,*], **Yifan Qi**[1,*], **Zixiang Wang**[1], **Lei Gu**[1], **Dehao Sui**[1], **Haoran Hu**[1], **Xichen Zhang**[3], **Ziyi He**[2], **Junjun He**[4], **Liantao Ma**[1,†], **Lequan Yu**[2,†]

[1] *Peking University*
[2] *The University of Hong Kong*
[3] *The Hong Kong University of Science and Technology*
[4] *Shanghai Artificial Intelligence Laboratory*

Abstract: The rapid proliferation of scientific knowledge presents a grand challenge: transforming this vast repository of information into an active engine for discovery, especially in high-stakes domains like healthcare. Current AI agents, however, are constrained by static, predefined strategies, limiting their ability to navigate the complex, evolving ecosystem of scientific research. This paper introduces HealthFlow, a self-evolving AI agent that overcomes this limitation through a novel meta-level evolution mechanism. HealthFlow autonomously refines its high-level problem-solving policies by distilling procedural successes and failures into a durable, structured knowledge base, enabling it to learn not just how to use tools, but how to strategize. To anchor our research and provide a community resource, we introduce EHRFlowBench, a new benchmark featuring complex health data analysis tasks systematically derived from peer-reviewed scientific literature. Our experiments demonstrate that HealthFlow's self-evolving approach significantly outperforms state-of-the-art agent frameworks. This work offers a new paradigm for intelligent systems that can learn to operationalize the procedural knowledge embedded in scientific content, marking a critical step toward more autonomous and effective AI for healthcare scientific discovery.

Keywords: large language model agent, multi-agent system, self-evolving, AI for healthcare

○ Code   ⬢ Dataset   ⊕ Project Page

1 Introduction

The rapid expansion of scientific literature, datasets, and knowledge bases has created an unprecedented opportunity for AI-driven discovery, particularly in high-stakes domains like healthcare [1]. A central challenge is developing intelligent systems that can navigate this complex information ecosystem to accelerate research. With the advent of Large Language Models (LLMs), a new paradigm of AI-driven scientific discovery has emerged, where autonomous agents are poised to revolutionize research workflows [2, 3, 4, 5, 6]. These agents promise to accelerate breakthroughs by automating complex data analysis, from hypothesis generation to navigating the vast, heterogeneous data landscapes of scientific literature and Electronic Health Records (EHRs) [7, 8, 9, 10]. By orchestrating sophisticated analytical pipelines, they offer the potential to uncover novel clinical insights at an unprecedented scale and pace [11, 12].

However, the capabilities of current AI agents are often limited because their high-level strategic frameworks are static and predefined. This is particularly detrimental in healthcare research, a domain characterized by open-ended problems, noisy data, and the need to dynamically adapt plans based on intermediate findings [12]. Existing agents can learn to refine the usage of a specific tool [13, 9] or improve a reasoning template for a sub-problem [8], but they operate within a fixed cognitive architecture. The overarching strategy that dictates how to decompose a new problem, orchestrate a multi-step workflow, and revise a plan remains hard-coded by human engineers and is outside the agent's learning scope. This can impose a significant limit on their autonomy; an agent can become highly efficient at executing a brittle or suboptimal strategy, but it cannot learn to devise a better one. This reliance on hand-crafted priors is contrary to a core lesson from AI's history: the consistent superiority of learned, general mechanisms over fixed, engineered solutions [14].

To address this limitation, we propose HealthFlow, an AI agent framework driven by meta-level strategic planning, learning, and self-evolution. HealthFlow transcends component-level optimization by treating every task as an

---

*Equal contribution. †Corresponding authors. ✉ yhzhu99@gmail.com, lqyu@hku.hk, malt@pku.edu.cn

experience from which to refine its own high-level management policies. Its architecture features a reflective loop where the entire execution trace of a task, including successes, failures, and corrections, is analyzed to synthesize abstract, structured knowledge. This knowledge, encapsulated as effective procedural patterns or critical data-handling warnings, directly reshapes the agent's future strategic choices, such as how to triage tasks or structure an analytical plan. Consequently, HealthFlow learns not just to execute tasks more effectively but to strategically manage the problem-solving process itself.

Furthermore, evaluating such advanced agent capabilities reveals a critical gap in existing resources. General-purpose agent benchmarks [15] lack the necessary domain specificity, while prevalent medical datasets [16, 17] are dominated by closed-ended question-answering tasks, failing to assess the complex data analysis and modeling skills central to clinical research [7, 18]. To address this, we introduce EHRFlowBench, a new benchmark comprising realistic, evidence-grounded data analysis workflows. These workflows are systematically mined from established scientific publications, effectively transforming unstructured research literature into a structured testbed for the community.

In summary, our contributions are threefold:

(1) We propose HealthFlow, an AI agent that introduces meta-level strategic learning. Its core innovation is an experience-driven evolution mechanism that refines the agent's high-level orchestration policies, a stark contrast to prior work focused on operational-level components.

(2) We introduce EHRFlowBench, a public benchmark for AI agent-driven analysis of electronic health records, featuring complex, realistic clinical research tasks derived from peer-reviewed literature to ground future research.

(3) We conduct comprehensive experiments showing that HealthFlow's self-evolving strategic approach yields significant improvements in task success, robustness, and efficiency compared to state-of-the-art agent frameworks.

Table 1. Architectural and evolutionary comparison of representative AI agent frameworks.

Framework	Benchmarked Tasks	Evolutionary Mechanism	Key Characteristics & Limitations
Biomni [12] *(Stanford)*	Biomedical data analysis (gene, drug, etc.)	✗ No evolution	Dynamically composes workflows but relies on a fixed set of tools, limiting its adaptability to tasks that require newly developed tools.
AFlow [19] *(ICLR 2025 Oral)*	Code generation, Math, QA	✓ MCTS-based workflow refinement	Frames workflow generation as a search problem is computationally intensive. Rely on learning abstract strategic knowledge from outcomes.
Alita [13] *(Princeton)*	Agent (general), math, medical VQA	✓ MCP creation & reuse	Creating new MCP tools but lacks a mechanism for refining its high-level strategic planning or workflow orchestration.
STELLA [9] *(Princeton)*	Biomedical QA	✓ Evolving template library & tool	Evolution is component-level, improving reasoning templates and the toolset. Core multi-agent coordination and strategy remain static.
AlphaEvolve [20] *(Google)*	Scientific computing, Algorithm optimization	✓ Evolutionary code mutation	Focus on optimizing a specific artifact (the code) based on a fixed evaluation metric, rather than evolving the agent's high-level strategic planning.
HealthFlow *(Ours)*	**Agentic health data science, medical QA**	**✓ Synthesizing experience (reflection, feedback, etc.)**	**Meta-level learning, allowing agents to evolve its own strategic planning from experience, moving beyond component-level optimization.**

2 Related Work

Predefined agentic workflows. Early and many contemporary agentic systems rely on human-designed workflows with fixed operational loops. Pioneering efforts like AutoGPT [21] and GPT4Tools [22] demonstrate autonomous task execution by chaining LLM calls, but their high-level strategy for task decomposition and execution is hard-coded. Subsequent systems such as HuggingGPT [23] and MetaGPT [24] introduce more sophisticated tool orchestration and multi-agent role-playing, yet their overarching operational sequences remain static. This paradigm extends to scientific domains. For instance, frameworks like SciToolAgent [25] provide more scientific tools, Robin [11] automates a predefined cycle of hypothesis generation and experimentation, and Biomni [12] executes complex biomedical tasks using a fixed "select-plan-execute" loop. While powerful, these systems are fundamentally constrained by their inability to learn from or adapt their high-level strategy.

Towards self-evolving agents. A more advanced line of research enables agents to evolve, but this evolution is typically confined to the operational level, improving specific components or artifacts within a fixed strategic framework. One approach focuses on refining tools and reasoning templates. For example, STELLA [9] and OriGene [8] improve their performance by expanding a library of tools and refining thought-process templates based on feedback. Other

methods focus on iterative self-improvement on a single output, such as self-reflection [26], Self-Refine [27], or optimizing a specific artifact, like the evolutionary code mutation in AlphaEvolve [20]. Frameworks like TextGrad introduce optimization by backpropagating LLM-generated feedback, offering a different paradigm for refinement [28]. As summarized in Table 1, despite their advances, these approaches share a common limitation: the agent learns to be a better tool-user or solution-reuser, but not a better strategic manager. The overarching meta-strategy remains predefined and outside the agent's learning scope.
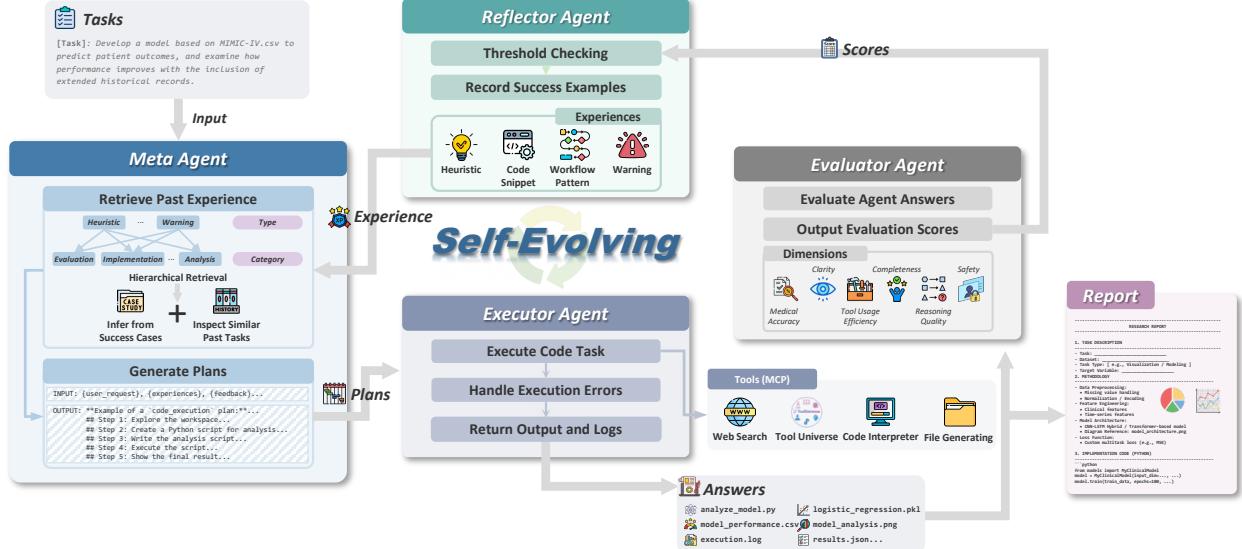

Figure 1. The self-evolving architecture of HealthFlow. The framework operates in a continuous learning loop. (1) A task is received by the meta agent, which generates a strategic plan by retrieving relevant past experiences. (2) The executor agent executes this plan using tools, producing results and detailed logs. (3) The evaluator agent assesses the execution, providing scores and feedback for immediate, short-term correction. (4) Upon successful completion, the reflector agent analyzes the entire process to synthesize abstract, structured experience (e.g., heuristics, workflow patterns). Experiences are stored in a persistent memory, augmenting the meta agent's strategic capabilities for future tasks and enabling the system's long-term, meta-level evolution.

3 Methodology

HealthFlow achieves autonomous healthcare research through a multi-agent architecture and a meta-level learning loop (Figure 1).

3.1 Formalism and System Overview

We formalize a research problem as a task T. The goal of HealthFlow is to produce a solution S that satisfies the requirements of T. This is achieved through a stateful, iterative process. At each turn i, the system generates a plan P_i, which is a sequence of actions $\{a_1, \ldots, a_k\}$. An action a can be a tool call or code execution. The execution of plan P_i produces a trace τ_i, which logs all actions, intermediate outputs, and errors. The system's strategic knowledge is stored in a dynamically growing experience memory, $\mathcal{M} = \{E_1, \ldots, E_N\}$. The core of HealthFlow is a learning process that updates $\mathcal{M}$ based on successful task completions, thereby improving its future planning capabilities.

3.2 A Collaborative Multi-Agent Architecture

To manage the cognitive complexity of research tasks, HealthFlow employs a team of four specialized agents, each responsible for a distinct phase of the problem-solving lifecycle: planning, execution, evaluation, and reflection.

Meta agent: strategic planner. The meta agent, A_M, serves as the cognitive orchestrator. Given a task T, the experience memory $\mathcal{M}$, and feedback f_{i-1} from a previous attempt (where f_0 is null), it devises a high-level strategic plan P_i. This process begins by retrieving a set of relevant experiences $\{E_k\} \subset \mathcal{M}$. These experiences augment the agent's context, guiding it to generate a more robust and efficient plan. Formally:

$$\{E_k\} = \text{Retrieve}(\mathcal{M}, T) \quad \text{and} \quad P_i = A_M(T, \{E_k\}, f_{i-1}) \tag{1}$$

Executor agent: transparent execution engine. The executor agent, A_E, is responsible for grounding the meta agent's strategic plan into concrete actions. It operates within a secure, sandboxed environment and translates plan P_i into a sequence of tool calls and code executions, producing a detailed execution trace τ_i:

$$\tau_i = A_E(P_i) \tag{2}$$

Crucially, the executor is a transparent "CodeAct"-style engine [29], ensuring that every decision, tool interaction, and intermediate result is auditable and reproducible. The executor integrates a wide array of tools via the Model Context Protocol (MCP) [30], including web search and access to ToolUniverse [10, 31], an ecosystem of machine learning models, datasets, and scientific packages.

Evaluator agent: short-term corrector. The evaluator agent, A_V, provides immediate, task-specific critique to enable a tight self-correction loop. After an execution attempt, it assesses the trace τ_i and any generated artifacts against the task requirements T:

$$(s_i, f_i) = A_V(\tau_i, T) \tag{3}$$

It produces a quantitative score s_i and qualitative, actionable feedback f_i. If s_i is below a predefined success threshold θ_{succ}, the feedback f_i is routed back to the meta agent to guide the generation of a revised plan P_{i+1}, initiating a new cycle of execution and evaluation.

Reflector agent: long-term knowledge synthesizer. The reflector agent, A_R, is the engine of HealthFlow's long-term, meta-level evolution. It is activated upon successful task completion (i.e., when $s_i \geq \theta_{succ}$). The reflector analyzes the entire history of the successful attempt, including initial failures and subsequent corrections documented in the traces $\{\tau_1, \ldots, \tau_i\}$. Its goal is to distill this procedural history into abstract, generalizable knowledge. This is achieved by instructing an LLM to synthesize insights according to a predefined JSON schema:

$$E_{new} = A_R(\{\tau_1, \ldots, \tau_i\}, T) \tag{4}$$

The output, a new set of experiences E_{new}, contains structured insights such as effective heuristics, reusable workflow patterns, or warnings about potential data pitfalls.

3.3   Meta-Level Evolution through Experience

HealthFlow adapts by transforming procedural execution into durable knowledge. We provide a formal proof for the efficacy of this evolutionary mechanism in Section C.

Synthesizing experience into structured knowledge. To ensure learned knowledge is actionable and retrievable, the reflector agent, A_R, synthesizes experiences into a structured format. An experience E is a record with key attributes: a type E_{type} in {heuristic, code_snippet, workflow_pattern, warning}; a categorical label $E_{category}$ (e.g., pediatric_care, EHR_data_preprocessing), and a content body $E_{content}$ containing the specific knowledge. After a successful task, A_R analyzes the execution trace to generate a set of new experiences, E_{new}. These structured experiences are then added to the memory: $\mathcal{M} \leftarrow \mathcal{M} \cup E_{new}$.

Experience-augmented planning and learning. The framework learns by continuously populating its experience memory $\mathcal{M}$. When a new task T' is presented, the meta agent A_M retrieves relevant prior knowledge using an LLM-based re-ranking strategy. In this process, all experiences from $\mathcal{M}$ are provided as context to an LLM, which then ranks them by semantic relevance to the task. The top-k ($k = 5$) experiences are incorporated into the prompt for A_M, providing a rich, contextual foundation for its initial plan. As $\mathcal{M}$ grows, A_M gains access to a broader and more refined set of strategies. The full algorithm is detailed in Section B.

Handling conflicting and evolving knowledge. The experience memory may accumulate advice that appears contradictory when viewed in isolation. HealthFlow treats this not as a flaw, but as a feature of rich, context-dependent knowledge. Seemingly conflicting heuristics often reflect valid strategies for different scenarios (e.g., distinct patient cohorts or data types). Our framework leverages the meta agent's contextual reasoning by retrieving a set of relevant, albeit potentially diverse, experiences. The LLM-based agent then dynamically synthesizes this advice, prioritizing or adapting strategies that are most applicable to the specific nuances of the current task, allowing for flexible, context-aware decision-making.

Knowledge bootstrapping in training mode. To address the cold-start problem, HealthFlow includes a training mode where it processes a curated set of problems with known reference solutions. In this mode, the evaluator uses the ground truth for critique, and crucially, the reflector is only permitted to synthesize experiences from tasks that are successfully validated against the reference. This supervised process populates the experience memory with verified, high-quality knowledge, bootstrapping the agent's strategic capabilities before it encounters new, unseen tasks.

3.4 EHRFlowBench: A Benchmark for Agentic Healthcare Research

The development of increasingly sophisticated autonomous agents necessitates benchmarks that can rigorously assess their real-world problem-solving abilities. In the general domain, benchmarks like GAIA [15] have become a standard for evaluating agents on tasks requiring tool use, multi-step reasoning, and web navigation. However, a similar standard is conspicuously absent in healthcare research. We introduce EHRFlowBench, designed to mirror the complexity of real-world research challenges and facilitate the reproducible evaluation of advanced healthcare agents.

Our creation process is a two-stage, LLM-assisted and human-verified procedure. (1) *Candidate paper screening*: The process begins with the collection of 51,280 papers published between 2020 and 2025 from top-tier AI and data mining conferences (AAAI, ICLR, ICML, NeurIPS, IJCAI, KDD, WWW). To identify EHR-related papers, we use a majority-voting ensemble of LLMs to classify titles based on their relevance to AI applications on EHR data, resulting in 162 candidate papers. This is followed by manual review, yielding 118 highly relevant papers. (2) *Task extraction*: We then prompt an LLM to extract evidence-grounded tasks from these papers. The LLM is instructed to generate a detailed task description, a category, and a reference answer for each, resulting in an initial pool of 585 tasks.

We manually consolidate semantically similar task types into a final taxonomy of 10 major categories that cover the research lifecycle (Figure 2). During the final manual selection phase, we apply stratified sampling to ensure category diversity. We retain all tasks from categories with fewer than 10 instances. For larger categories, we select a representative subset of 10 to 20 tasks, guided by criteria such as diversity of modeling techniques and clinical objectives. This process results in a final benchmark of 110 tasks, with 100 for evaluation and 10 (one from each of the 10 categories) for the training set used by HealthFlow.
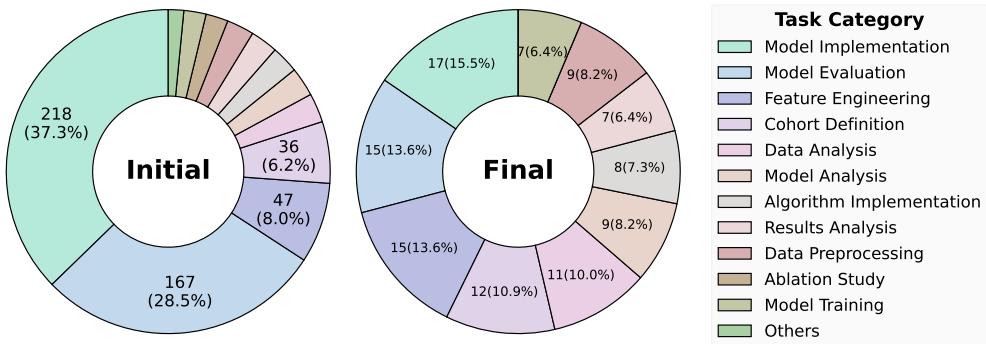

Figure 2. Task category distribution in EHRFlowBench. The initial distribution of 585 LLM-extracted tasks (left) is refined through manual curation and stratified sampling into a final set of 110 tasks across 10 core research categories (right), with irrelevant categories like "ablation study" being discarded.

4 Experimental Setups

4.1 Datasets, Tasks, and Evaluation

We define autonomous healthcare research as the ability to call tools, write code, and independently complete medical tasks for a given research question. To this end, we evaluate HealthFlow across five benchmarks assessing open-ended research capabilities (EHRFlowBench), complex EHR data analysis and modeling (MedAgentBoard [32]), medical knowledge reasoning (MedAgentsBench [33], HLE [34]), and tool-augmented clinical reasoning (CureBench [35]).

EHRFlowBench. Serving as the primary testbed for end-to-end, multi-step autonomous healthcare research, it consists of 110 complex tasks derived from peer-reviewed literature. Performance is assessed using an LLM-as-a-judge ensemble. For each task, the ensemble provides integer scores from 1 to 5 across three dimensions, which are then combined into a weighted final score for methodology soundness (70%), presentation (20%), and artifact quality (10%).

MedAgentBoard. MedAgentBoard [32] evaluates practical skills in handling structured EHR data. It includes 100 tasks on the MIMIC-IV [36, 37] and TJH [38] datasets, requiring a full analytical pipeline. Performance is primarily measured by task success rate, which is determined through manual human evaluation. A task is deemed successful if the agent's final generated artifacts fully and correctly satisfy all stated requirements. We also use an LLM to score solutions on a 1-to-5 scale across four key dimensions: (1) data extraction and statistical analysis, (2) predictive modeling, (3) data visualization, and (4) report generation.

MedAgentsBench. This benchmark [33] is used to evaluate the agent's foundational medical reasoning and knowledge retrieval capabilities, which are prerequisites for any meaningful healthcare research. We use a curated set of 100 challenging multiple-choice questions from its "hard set" to avoid ceiling effects. Performance is measured by accuracy.

Humanity's Last Exam (HLE). We use the medical subset of HLE [34] to assess reasoning on expert-level problems designed to be difficult for LLMs. Our test set includes 45 text-only questions from the "Biology/Medicine" category. Following the official protocol [39], performance is measured by binary correctness.

CureBench. To evaluate agentic tool-augmented reasoning in a clinically relevant context, we include CureBench [35], a benchmark for therapeutic decision-making. CureBench tasks models with complex reasoning over patients, diseases, and drugs, requiring the use of external biomedical tools (e.g., FDA databases, PubMed). We use a randomly selected set of 100 multiple-choice questions from the validation set, as the official test set lacks ground-truth labels. An additional 10 samples are randomly selected for HealthFlow's training. Performance is measured by accuracy.

4.2 Baseline Methods

We compare HealthFlow against a set of representative baselines: (1) general LLMs: DeepSeek-V3 [40], DeepSeek-R1 [41]; (2) medical LLMs: HuatuoGPT-o1 [42], MedGemma [43]; (3) multi-agent collaboration frameworks: MedAgents [44], MDAgents [45], ColaCare [46]; (4) general agent frameworks: AFlow [19], Alita [13]; and (5) biomedical agent frameworks: Biomni [12], STELLA [9].

4.3 Implementation Details

EHRFlowBench curation and evaluation. The LLM ensemble for paper filtering includes DeepSeek-V3, DeepSeek-R1, and Qwen3-235B (Qwen3-235B-A22B-Instruct-2507 [47]). Task extraction is performed by DeepSeek-V3. The EHRFlowBench's LLM-as-a-judge evaluation ensemble consists of DeepSeek-V3, DeepSeek-R1, Claude-4-Sonnet [48], Kimi-K2 [49], and GLM-4.5 [50]. The MedAgentBoard's LLM judge is DeepSeek-V3. We use bootstrapping on all test set samples 100 times to report the mean and standard deviations. Metrics of success rate and accuracy are multiplied by 100 for readability purposes.

Hardware and software configuration. All experiments are conducted on a Mac Studio M3 Ultra with 512GB of RAM. HuatuoGPT-o1 and MedGemma are deployed locally using LMStudio. All agent-based baselines are powered by DeepSeek-V3 using their official implementations (except for Alita, for which we use the OpenAlita community implementation [51]). We configure external tools, including web search with the Serper search API and access to ToolUniverse, under MCP. HealthFlow is developed in Python 3.12. To ensure transparency and facilitate potential local deployment in healthcare settings, HealthFlow's meta, evaluator, and reflector agents use DeepSeek-V3 by default. For

Table 2. Main evaluation results of HealthFlow and its variants on five medical agent benchmarks. We compare our framework against baselines and present an enhanced version, *HealthFlow + ToolUniverse*, which integrates a broader set of external tools. For fair comparison, all agent frameworks are powered by DeepSeek-V3 by default. The ablation study demonstrates the contribution of key components: *w/o Feedback* is a version without reflection or evaluation; *w/o Experience* disables the experience memory; and *w/o Training* omits initial knowledge bootstrapping on EHRFlowBench and CureBench (the two benchmarks with curated training samples). The best result in each column is in bold.

Category	Methods	EHRFlowBench (LLM Score ↑)	MedAgentBoard (Success Rate % ↑)	MedAgentsBench (Accuracy % ↑)	HLE (Accuracy % ↑)	CureBench (Accuracy % ↑)
General LLM	DeepSeek-V3	2.65±0.03	3.70±2.28	8.42±2.29	2.33±2.45	86.20±3.57
	DeepSeek-R1	2.78±0.03	3.16±1.64	39.03±4.33	6.44±3.34	87.57±3.45
Medical LLM	HuatuoGPT-o1	1.83±0.06	0.00±0.00	19.10±3.58	6.91±3.97	78.31±4.26
	MedGemma	2.17±0.07	1.90±1.34	16.07±3.87	8.49±4.49	78.02±4.55
Multi-agent Collab.	MedAgents	1.76±0.06	0.00±0.00	20.00±3.64	2.33±2.45	86.15±3.80
	MDAgents	1.89±0.09	4.29±2.19	20.16±4.09	2.33±2.45	83.86±3.76
	ColaCare	2.04±0.07	1.16±1.04	21.58±3.54	0.00±0.00	85.95±3.62
General Agent	AFlow	3.31±0.06	4.90±2.11	30.30±4.46	0.00±0.00	81.95±3.80
	Alita	2.77±0.07	9.48±3.27	23.97±4.07	4.49±3.31	85.18±3.92
Biomedical Agent	Biomni	2.22±0.06	45.61±4.51	22.72±3.87	4.16±3.35	81.68±3.58
	STELLA	2.39±0.07	38.46±4.61	26.97±4.60	7.11±3.72	85.98±3.78
Ours	**HealthFlow + ToolUniverse**	**3.98±0.06**	**81.89±3.87**	**30.68±4.28**	**9.13±4.52**	**90.29±3.17**
	HealthFlow	3.82±0.07	66.09±5.06	28.08±4.51	4.96±3.34	88.31±3.31
	w/o Feedback	2.78±0.07	42.63±4.48	21.52±3.70	2.33±2.45	87.20±3.66
	w/o Experience	3.63±0.08	57.59±5.46	25.59±4.25	6.44±3.34	87.83±2.91
	w/o Training	3.80±0.07	—	—	—	88.01±3.59

our main experiments, HealthFlow's executor backend integrates ToolUniverse and is also powered by DeepSeek-V3. To provide a robust and standardized execution environment for baselines, other HealthFlow variants default to a Claude Code [52] backend powered by the Kimi-K2 model. During a task, HealthFlow employs an internal self-correction loop; the evaluator agent assigns a score on a 10-point scale, and a score below 6.0 triggers a retry, with a maximum of 3 attempts per task.

5 Experimental Results and Analysis

5.1 Main Results and Ablation Study

Comparison against diverse baselines. As shown in Table 2, HealthFlow substantially outperforms all baselines across tasks, especially on EHRFlowBench and MedAgentBoard, which test end-to-end data analysis and modeling. The integration of ToolUniverse provides a substantial performance uplift, demonstrating the value of a rich, accessible scientific tool ecosystem. On knowledge-intensive, tool-light benchmarks like MedAgentsBench, HLE, and CureBench, HealthFlow's performance is competitive. As expected, non-agentic LLMs and multi-agent frameworks that lack code execution capabilities struggle on these tasks; they are unable to complete the multi-step analytical pipelines. Biomedical agent frameworks like Biomni and STELLA outperform general agents on MedAgentBoard, likely benefiting from specialized, built-in tools that provide relevant priors for healthcare data analysis.

Further illustrating this advantage, a head-to-head comparison in Figure 3 shows HealthFlow achieving a dominant win rate against other leading frameworks. On the highly complex and open-ended tasks of EHRFlowBench, Health-Flow's superiority is clear. Its robust performance is confirmed on MedAgentBoard, which focuses on structured EHR data analysis pipelines, where it maintains a significant lead even against strong, domain-specific baselines.

Ablation on core components. Our ablation study in Table 2 quantifies the contribution of HealthFlow's core components. Removing the feedback loop entirely (*w/o Feedback*), which includes both the evaluator and reflector agents, leads to a significant performance drop on EHRFlowBench (from 3.82 to 2.78) and MedAgentBoard (from 66.09% to 42.63%). This result underscores a key finding: an initial plan is rarely perfect, and the ability to iteratively critique and refine the workflow is fundamental to success. Disabling only the long-term experience memory (*w/o*
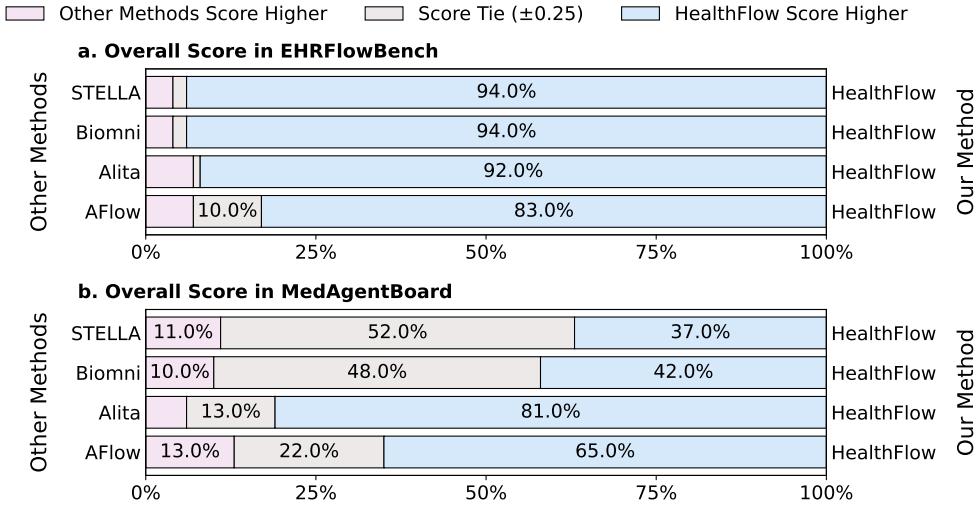
Figure 3. Head-to-head performance of HealthFlow against leading agent frameworks on (a) EHRFlowBench and (b) MedAgent-Board. Each bar shows the distribution for all tasks in a direct comparison against a specific baseline. Outcomes are categorized as a tie if the score difference is ≤ 0.25.

Experience) also degrades performance (from 3.82 to 3.63 on EHRFlowBench), demonstrating that while short-term correction is critical, the accumulation of strategic knowledge provides a durable advantage. Finally, omitting the initial knowledge bootstrapping (*w/o Training*) results in a minor performance decrease on EHRFlowBench (from 3.82 to 3.80). This indicates that while pre-populating the memory with verified knowledge is beneficial, HealthFlow's ability to learn on the fly from new tasks remains a powerful mechanism for adaptation.

5.2 Further Analysis

Impact of LLM backbones and executors. Table 3 reveals that the choice of underlying LLM is critical. The framework's architecture separates frontend reasoning models (meta, reflector, evaluator) from the backend coding model (executor). Using a more powerful reasoning model like DeepSeek-R1 as the frontend significantly boosts performance, highlighting that better strategic planning directly translates to better outcomes. Conversely, a capable strategy is ineffective without a reliable backend. Substituting the executor with a less proficient Qwen3-Coder causes a near-total performance collapse on MedAgentBoard (from 66.09% to 6.04%). Failure analysis reveals that Qwen3-Coder frequently misinterpreted file system instructions, failing to locate input data and causing tasks to terminate prematurely. This starkly illustrates that a robust execution engine demands exceptional instruction-following fidelity, not just code generation. The best performance is achieved with our native backend powered by DeepSeek-V3 and integrated with ToolUniverse due to pairing an execution backend that is not only a capable coder but also part of a rich, domain-aware tool ecosystem, confirming that optimal performance requires synergy between high-level strategy and execution capability.

Fine-grained evaluation across task categories. To better understand HealthFlow's capabilities, we conduct a fine-grained analysis across key dimensions on EHRFlowBench and MedAgentBoard (Table 4). On EHRFlowBench, HealthFlow excels in method soundness (3.72) and artifact generation (3.96), demonstrating its ability to both devise sound research plans and produce high-quality outputs like statistical code, trained models, and visualizations. On MedAgentBoard, HealthFlow shows superior performance on the most complex, integrative tasks: predictive modeling (4.21) and report generation (4.10). While a specialized baseline like STELLA performs competitively on routine sub-tasks such as data extraction and visualization, HealthFlow's advantage in the end-to-end modeling process highlights the value of its adaptive meta-planning capabilities for holistic problem-solving.

Table 3. Impact of different LLM backbones on HealthFlow performance. We evaluate variants by changing the frontend (meta, reflector, evaluator agents) and backend (executor agent).

Frontend (Meta/Reflector)	Backend (Executor)	EHRFlowBench (LLM Score ↑)	MedAgentBoard (Success Rate % ↑)
DeepSeek-V3	Claude Code (Kimi)	3.82±0.07	66.09±5.06
DeepSeek-R1	Claude Code (Kimi)	3.84±0.07	81.64±3.77
Kimi-K2	Claude Code (Kimi)	2.41±0.10	79.53±4.54
Qwen3-235B	Claude Code (Kimi)	3.53±0.09	79.03±4.41
DeepSeek-V3	Qwen3-Coder (Qwen)	2.64±0.07	6.04±2.34
DeepSeek-V3	Ours (DeepSeek-V3)	**3.98**±0.06	**81.89**±3.87

Table 4. Fine-grained performance breakdown on EHRFlowBench and MedAgentBoard. Scores are the mean and standard deviation from multiple LLM judges on a 1-5 scale. This consolidated view shows HealthFlow's strong performance across both high-level research methodology (EHRFlowBench) and practical health data science tasks (MedAgentBoard).

| Methods | EHRFlowBench | | | MedAgentBoard | | | |
	Method Soundness (70%)	Presentation Quality (20%)	Artifact Generation (10%)	Data Extraction & Stats	Predictive Modeling	Data Visualization	Report Generation
AFlow	3.22±0.95	3.99±0.58	2.60±1.12	3.67±1.20	3.84±0.73	3.64±1.63	3.79±0.97
Alita	2.64±1.04	3.49±0.81	2.19±1.01	3.24±1.40	2.70±1.13	3.00±1.34	2.79±1.06
Biomni	2.04±1.08	2.72±0.94	2.43±1.11	4.62±0.69	3.40±0.72	4.22±0.82	4.03±0.71
STELLA	2.17±1.03	2.92±0.93	2.80±1.09	**4.68**±0.65	3.28±1.13	**4.75**±0.43	3.15±1.13
HealthFlow	**3.72**±1.03	**4.15**±0.67	**3.96**±0.98	4.58±0.70	**4.21**±0.68	**4.75**±0.43	**4.10**±0.85

Dynamics of experience synthesis and retrieval. An analysis of HealthFlow's learning loop reveals a dynamic adaptation to task structure (Figure 4). For the complex, open-ended tasks in EHRFlowBench, the agent prioritizes synthesizing abstract `heuristics` (31.1% of experiences) but most frequently retrieves concrete `code_snippets` (1.91 average per task). This indicates a strategy of grounding high-level plans with actionable code for novel problems. Conversely, for the structured pipelines in MedAgentBoard, the agent most often retrieves `heuristics` (1.91 average per task), suggesting a focus on refining existing workflows. Across both benchmarks, synthesized knowledge is overwhelmingly concentrated in the `Clinical & Medical Analysis` category (Figure 5), confirming that learning is sharply focused on core domain challenges. This interplay allows HealthFlow to build a contextually relevant knowledge base, optimizing for strategic guidance on routine tasks and reusable code for novel ones.
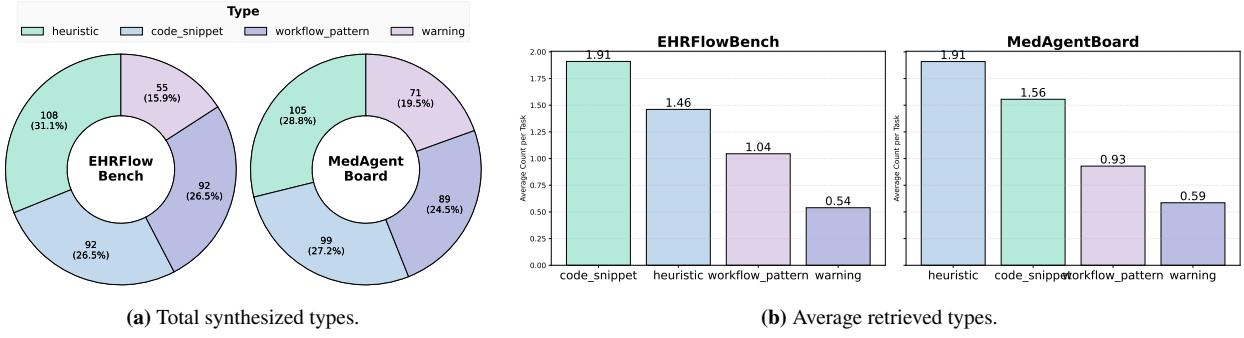

(a) Total synthesized types.   (b) Average retrieved types.

Figure 4. Distribution of experience synthesis and retrieval across EHRFlowBench and MedAgentBoard benchmarks.
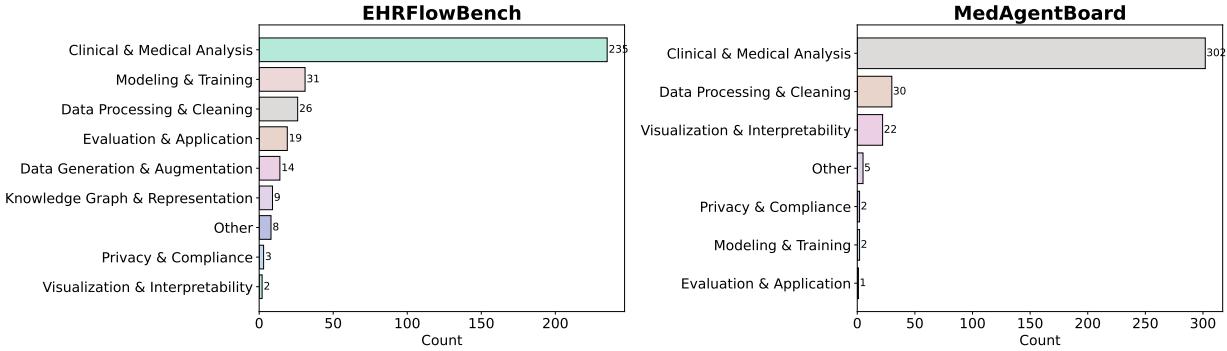
Figure 5. Distribution of synthesized experience categories.

Case studies. Two case studies illustrate HealthFlow's capabilities. The first, a data visualization task from MedAgentBoard, demonstrates the practical impact of meta-level learning. As shown in Figure 6, when tasked with visualizing the correlation between systolic and diastolic blood pressure, HealthFlow's meta agent retrieves relevant prior knowledge, including a critical heuristic about handling outliers in clinical data. This experience informs a robust plan that incorporates proactive data validation, such as setting clinically plausible axis limits, a step that is not explicitly requested but is essential for meaningful analysis.

The outcome of this experience-driven approach is shown in Figure 7. HealthFlow produces a clean, interpretable visualization (b) that aligns closely with the reference answer (a). In stark contrast, competing agents like Biomni (c) and STELLA (d), which lack this adaptive planning mechanism, fail to perform the necessary validation. Their resulting plots are rendered uninterpretable by outliers that compress the primary data distribution, highlighting a critical failure mode for agents that operate without domain-aware, learned strategies.
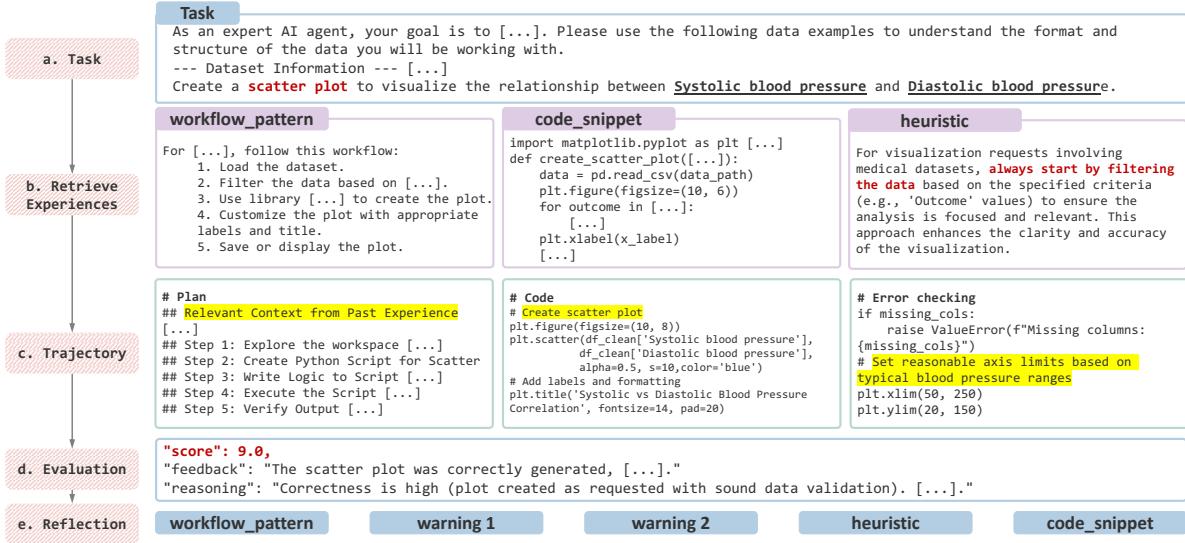

Figure 6. An illustration of HealthFlow's workflow on a data visualization task from MedAgentBoard (#56). The agent retrieves past experiences (b) to inform its strategic plan (c), leading to a high-quality, domain-aware output. The entire trajectory is then analyzed to synthesize new, generalizable knowledge through reflection (e), completing the self-evolution loop.

A second case from EHRFlowBench demonstrates versatility on more complex research tasks. Without a provided dataset, HealthFlow autonomously simulates realistic data, implements a specified CNN-LSTM architecture, and executes a full training and evaluation workflow. It successfully produces all required artifacts, including a training loss plot, final accuracy metrics, and saved model checkpoints. These cases show that HealthFlow's ability to learn and
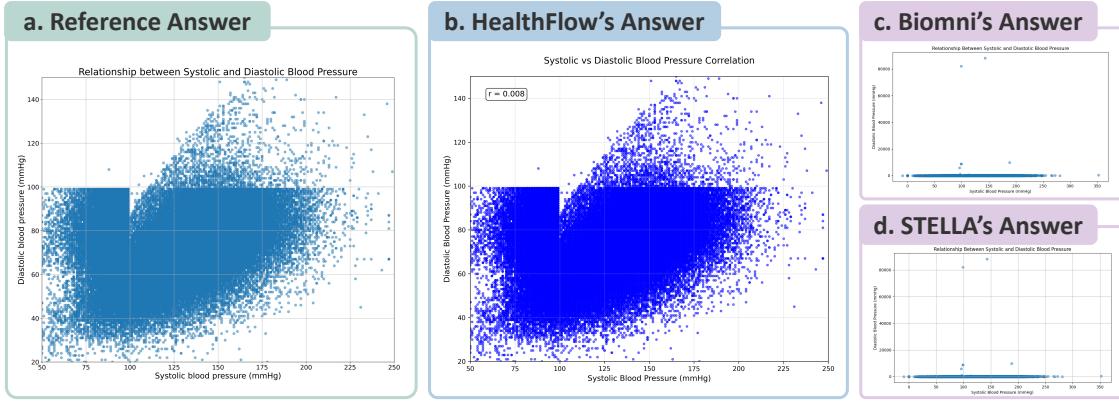
10

Figure 7. Comparison of outputs for a data visualization task on MedAgentBoard (#56). HealthFlow (b) performs essential, experience-driven data validation, producing a result comparable to the reference answer (a). In contrast, frameworks like Biomni (c) and STELLA (d) fail to handle outliers, rendering their visualizations uninterpretable. Alita failed to generate the required image.

apply strategic knowledge enables it to conduct robust, multi-step research that adapts to domain-specific nuances.

Human evaluation. To complement our automated metrics, we conduct a rigorous human evaluation to assess the practical utility and quality of the generated solutions. We enlist 12 expert evaluators, comprising PhD and MD students from disciplines such as AI for healthcare, biostatistics, biomedical engineering, and clinical medicine. The evaluators are tasked with assessing a random subset of 20 tasks, with 10 drawn from EHRFlowBench and 10 from MedAgentBoard. In a blind review process, they perform a head-to-head comparison of the final solutions generated by HealthFlow, Alita, Biomni, and STELLA. For each task, evaluators select the single best solution, with the option to choose "None" if no output is satisfactory. To mitigate bias, the identities and presentation order of the agents are randomized for each task and evaluator. The average pairwise inter-rater agreement is 71.54%, indicating substantial consensus. The results, summarized in Figure 8, reveal a clear expert preference for HealthFlow's solutions across tasks.
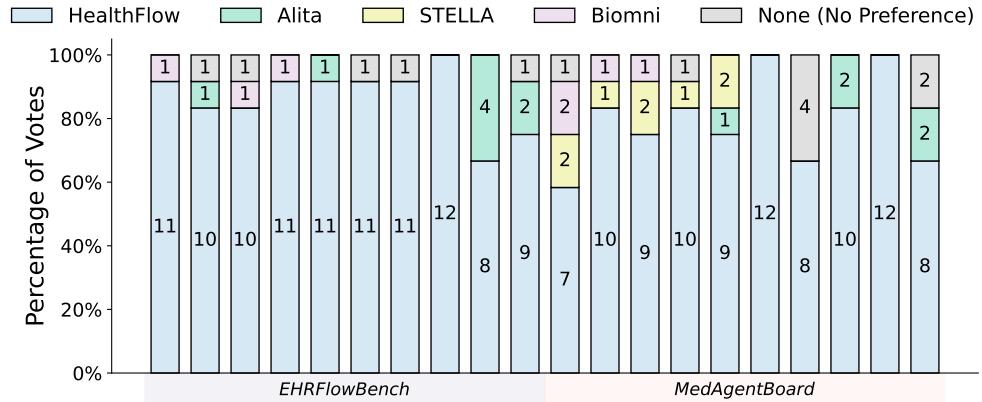

Figure 8. Human evaluation results. Distribution of votes from 12 domain experts comparing solutions from four agent frameworks across 20 tasks.

6 Discussion

Limitations. While HealthFlow represents a significant advance, its performance is fundamentally tethered to the capabilities of its underlying LLMs. Biases or knowledge gaps in the LLMs can propagate into the agent's strategic plans

and synthesized experiences. Furthermore, the experience synthesis process itself carries an inherent risk: the agent might distill flawed or overly specific heuristics from idiosyncratic successes, which could degrade future performance if not properly generalized.

Future work. The HealthFlow framework could be adapted to other scientific fields, such as computational biology, by curating domain-specific tools and experiences. Extending the framework to handle multi-modal inputs, such as medical imaging alongside EHR data, would represent a major step towards a more comprehensive AI research assistant.

Broader impact. HealthFlow can significantly accelerate healthcare research by automating complex data analysis and operationalizing scientific knowledge from published literature. However, deploying autonomous agents in this high-stakes domain presents risks, including the potential for flawed scientific conclusions and data privacy concerns. We advocate for a human-in-the-loop approach, positioning the agent as a powerful tool to augment, not replace, expert oversight to ensure responsible and effective scientific discovery.

7 Conclusion

In this paper, we introduced HealthFlow, a self-evolving AI agent for autonomous healthcare research. By implementing a meta-level strategic planning and evolution mechanism, HealthFlow learns not only to execute tasks but to evolve its own high-level orchestration policies from experience. This ability to learn how to manage research, rather than just how to perform individual steps, directly tackles the bottleneck of static strategies in current agents. To validate our approach, we developed EHRFlowBench, a challenging new benchmark derived from scientific literature, designed to test complex data analysis capabilities. Our experiments demonstrate that HealthFlow's adaptive, experience-driven strategy leads to superior performance compared to state-of-the-art frameworks. This work offers a promising approach for intelligent systems that can operationalize the vast procedural knowledge found in scientific literature, marking a significant step toward more autonomous and effective AI for healthcare research.

References

[1] Laura Sbaffi and Jennifer Rowley. Trust and credibility in web-based health information: a review and agenda for future research. *Journal of medical Internet research*, 19(6):e218, 2017.

[2] Abeba Birhane, Atoosa Kasirzadeh, David Leslie, and Sandra Wachter. Science in the age of large language models. *Nature Reviews Physics*, 5(5):277–280, 2023.

[3] Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, pages 1–3, 2025.

[4] Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*, 2025.

[5] L John Fahrner, Emma Chen, Eric Topol, and Pranav Rajpurkar. The generative era of medical ai. *Cell*, 188(14):3648–3660, 2025.

[6] Jincai Huang, Yongjun Xu, Qi Wang, Qi Cheems Wang, Xingxing Liang, Fei Wang, Zhao Zhang, Wei Wei, Boxuan Zhang, Libo Huang, et al. Foundation models and intelligent decision-making: Progress, challenges, and perspectives. *The Innovation*, 2025.

[7] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C. Ho, Carl Yang, and May Dongmei Wang. EHRAgent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22315–22339, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[8] Zhongyue Zhang, Zijie Qiu, Yingcheng Wu, Shuya Li, Dingyan Wang, Zhuomin Zhou, Duo An, Yuhan Chen, Yu Li, Yongbo Wang, et al. Origene: A self-evolving virtual disease biologist automating therapeutic target discovery. *bioRxiv*, pages 2025–06, 2025.

[9] Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. Stella: Self-evolving llm agent for biomedical research. *arXiv preprint arXiv:2507.02004*, 2025.

[10] Shanghua Gao, Richard Zhu, Zhenglun Kong, Ayush Noori, Xiaorui Su, Curtis Ginder, Theodoros Tsiligkaridis, and Marinka Zitnik. Txagent: An ai agent for therapeutic reasoning across a universe of tools. *arXiv preprint arXiv:2503.10970*, 2025.

[11] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Jon M Laurent, Muhammed T Razzak, Andrew D White, Michaela M Hinks, and Samuel G Rodriques. Robin: A multi-agent system for automating scientific discovery. *arXiv preprint arXiv:2505.13400*, 2025.

[12] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pages 2025–05, 2025.

[13] Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

[14] Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

[15] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[17] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics.

[18] Inioluwa Deborah Raji, Roxana Daneshjou, and Emily Alsentzer. It's time to bench the medical exam benchmark. *NEJM AI*, 2(2):AIe2401235, 2025.

[19] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[21] Significant Gravitas. Autogpt. https://github.com/Significant-Gravitas/AutoGPT, 2023. Accessed: 2025-07-27.

[22] Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. GPT4tools: Teaching large language model to use tools via self-instruction. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[23] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. In *Advances in Neural Information Processing Systems*, 2023.

[24] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Keyan Ding, Jing Yu, Junjie Huang, Yuchen Yang, Qiang Zhang, and Huajun Chen. Scitoolagent: a knowledge-graph-driven scientific agent for multitool integration. *Nature Computational Science*, pages 1–11, 2025.

[26] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[27] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[28] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

[29] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

[30] Anthropic. Introducing the model context protocol, 11 2024.

[31] Shanghua Gao, Richard Zhu, Pengwei Sui, Zhenglun Kong, Sufian Aldogom, Yepeng Huang, Ayush Noori, Reza Shamji, Krishna Parvataneni, Theodoros Tsiligkaridis, et al. Democratizing ai scientists using tooluniverse. *arXiv preprint arXiv:2509.23426*, 2025.

[32] Yinghao Zhu, Ziyi He, Haoran Hu, Xiaochen Zheng, Xichen Zhang, Zixiang Wang, Junyi Gao, Liantao Ma, and Lequan Yu. MedAgentBoard: Benchmarking multi-agent collaboration with conventional methods for diverse medical tasks. *The Thirty-nine Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[33] Xiangru Tang, Daniel Shao, Jiwoong Sohn, Jiapeng Chen, Jiayi Zhang, Jinyu Xiang, Fang Wu, Yilun Zhao, Chenglin Wu, Wenqi Shi, et al. Medagentsbench: Benchmarking thinking models and agent frameworks for complex medical reasoning. *arXiv preprint arXiv:2503.07459*, 2025.

[34] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[35] Shanghua Gao et al. Cure-bench @ neurips 2025 – ai reasoning for therapeutic decision-making. https://curebench.ai/, 2025. Accessed on 2025-10-06.

[36] Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1, 2023.

[37] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Brian Gow, Benjamin Moody, Steven Horng, Leo Anthony Celi, and Mark Roger. Mimic-iv (version 3.1), 10 2024.

[38] Li Yan, Hai-Tao Zhang, Jorge Goncalves, Yang Xiao, Maolin Wang, Yuqi Guo, Chuan Sun, Xiuchuan Tang, Liang Jing, Mingyang Zhang, et al. An interpretable mortality prediction model for covid-19 patients. *Nature machine intelligence*, 2(5):283–288, 2020.

[39] Center for AI Safety. Humanity's last exam. https://github.com/centerforaisafety/hle, 2024. Accessed: 2025-08-01.

[40] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[41] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[42] Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, and Benyou Wang. Towards medical complex reasoning with LLMs through medical verifiable problems. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14552–14573, Vienna, Austria, July 2025. Association for Computational Linguistics.

[43] Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, Cían Hughes, Charles Lau, et al. Medgemma technical report. *arXiv preprint arXiv:2507.05201*, 2025.

[44] Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning. *arXiv preprint arXiv:2311.10537*, 2023.

[45] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

[46] Zixiang Wang, Yinghao Zhu, Huiya Zhao, Xiaochen Zheng, Dehao Sui, Tianlong Wang, Wen Tang, Yasha Wang, Ewen Harrison, Chengwei Pan, et al. Colacare: Enhancing electronic health record modeling through large language model-driven multi-agent collaboration. In *Proceedings of the ACM on Web Conference 2025*, pages 2250–2261, 2025.

[47] Qwen Team. Qwen3-235b-a22b-instruct-2507. https://huggingface.co/Qwen/Qwen3-235B-A22B-Instruct-2507, 2025. Accessed: 2025-08-01.

[48] Anthropic. Claude sonnet 4. https://www.anthropic.com/claude/sonnet, 2025. Accessed: 2025-08-01.

[49] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[50] Zhipu AI. Glm-4.5: Reasoning, coding, and agentic abililties. https://z.ai/blog/glm-4.5, 2025. Accessed: 2025-08-01.

[51] ryantzr1. Openalita. https://github.com/ryantzr1/OpenAlita, 2025. Accessed: 2025-08-01.

[52] Anthropic. Claude code: Deep coding at terminal velocity. https://www.anthropic.com/claude-code, 2025. Accessed: 2025-08-01.

A Ethical Statement

The research presented in this paper adheres to high ethical standards for AI in healthcare. The development and evaluation of HealthFlow and its accompanying benchmark, EHRFlowBench, are guided by the core principles of beneficence, non-maleficence, and fairness. All datasets used in our experiments have been de-identified, ensuring that no protected health information (PHI) is exposed or used in our research.

HealthFlow is intended for use as a research tool to assist healthcare professionals and data scientists and is not designed to replace human clinical judgment or provide medical advice. To promote transparency and enable

independent verification of our results, we commit to making our source code, including the HealthFlow framework and the EHRFlowBench benchmark, publicly available.

B   HealthFlow Algorithm

The core operational loop of HealthFlow is detailed in Algorithm 1. The process illustrates the collaboration between the specialized agents, the short-term correction loop driven by the evaluator, and the long-term evolution driven by the reflector and the experience memory.

Algorithm 1 *The HealthFlow self-evolving workflow.*

Require: Task T, Experience Memory $\mathcal{M}$, Max Retries N_{max}
Ensure: Final Solution S or Failure
1: Initialize A_M, A_E, A_V, A_R
2: Initialize retry count $n \leftarrow 0$, feedback $f \leftarrow$ null
3: **while** $n < N_{max}$ **do**
4:          ▷ **Phase 1: Meta-Planning**
5:      Retrieve relevant experiences $\{E_k\} \subset \mathcal{M}$ for task T
6:      Augment context for A_M with $\{E_k\}$ and feedback f
7:      Generate plan $P \leftarrow A_M(T, \mathcal{M}, f)$
8:          ▷ **Phase 2: Execution**
9:      Execute plan, get trace $\tau \leftarrow A_E(P)$
10:          ▷ **Phase 3: Evaluation (Short-term Correction)**
11:      Evaluate outcome $(s, f) \leftarrow A_V(\tau, T)$
12:      **if** $s \geq \theta_{succ}$ **then**
13:          ▷ **Phase 4: Reflection (Long-term Evolution)**
14:          Synthesize new experience $E_{new} \leftarrow A_R(\tau, T)$
15:          Update memory $\mathcal{M} \leftarrow \mathcal{M} \cup \{E_{new}\}$
16:          **return** Solution from trace τ
17:      **else**
18:          Increment retry count $n \leftarrow n + 1$
19:      **end if**
20: **end while**
21: **return** Failure

C   Formal Proof of Evolving Efficacy

We provide a formal argument for the efficacy of HealthFlow's self-evolution mechanism. Our goal is to show that as the agent completes more tasks, its expected performance on future tasks monotonically improves.

Definitions. Let $\mathcal{D}$ be a stationary distribution of tasks T. Let $\pi(\mathcal{M})$ be the meta-agent's planning policy, which depends on the experience memory $\mathcal{M}$. The policy maps a task T to a plan P. Let $S(T, P)$ be a stochastic binary outcome function, where $S = 1$ indicates task success and $S = 0$ indicates failure. The quality of a policy is its expected success rate over the task distribution:

$$Q(\pi(\mathcal{M})) = \mathbb{E}_{T \sim \mathcal{D}}[S(T, \pi(\mathcal{M})(T))] \tag{5}$$

Assumptions. Our proof relies on two idealized assumptions about the learning process to illustrate the mechanism's potential in a theoretical model:
(1) **Effective Reflection:** When a task T is solved successfully, the reflector agent A_R generates a new experience E_{new}. We assume this experience captures a meaningful, generalizable aspect of the solution. Thus, for any future

task T' that is "similar" to T (i.e., for which E_{new} is relevant), using this experience will not decrease, and may increase, the probability of success. Formally, let $\mathcal{M}' = \mathcal{M} \cup \{E_{new}\}$. Then for any T',

$$\mathbb{P}(S(T', \pi(\mathcal{M}')(T')) = 1) \geq \mathbb{P}(S(T', \pi(\mathcal{M})(T')) = 1) \tag{6}$$

(2) **Effective Retrieval:** The meta-agent A_M can reliably retrieve relevant experiences. An irrelevant experience does not negatively impact planning for a given task.

Theorem. Let $\mathcal{M}_t$ be the experience memory after t successful task completions. Under the assumptions of Effective Reflection and Effective Retrieval, the quality of the planning policy is monotonically non-decreasing.

$$Q(\pi(\mathcal{M}_{t+1})) \geq Q(\pi(\mathcal{M}_t)) \tag{7}$$

Proof. At step t, the agent successfully completes a task T_t, generating a new experience E_{t+1}. The memory is updated to $\mathcal{M}_{t+1} = \mathcal{M}_t \cup \{E_{t+1}\}$. We want to compare $Q(\pi(\mathcal{M}_{t+1}))$ with $Q(\pi(\mathcal{M}_t))$.

$$Q(\pi(\mathcal{M}_{t+1})) = \mathbb{E}_{T \sim \mathcal{D}}[S(T, \pi(\mathcal{M}_{t+1})(T))] \tag{8}$$

$$= \int_{T \in \mathcal{D}} \mathbb{P}(S(T, \pi(\mathcal{M}_{t+1})(T)) = 1)p(T)dT \tag{9}$$

By the law of total expectation, we can condition on whether a task T is relevant to the new experience E_{t+1} or not. Let $\mathcal{D}_{rel} \subset \mathcal{D}$ be the subset of tasks for which E_{t+1} is relevant, and $\mathcal{D}_{irrel} = \mathcal{D} \setminus \mathcal{D}_{rel}$.

From our assumptions:

- For any $T' \in \mathcal{D}_{rel}$, by Effective Reflection (Assumption 1), the retrieval of E_{t+1} will lead to a plan that is at least as likely to succeed:

$$\mathbb{P}(S(T', \pi(\mathcal{M}_{t+1})(T')) = 1) \geq \mathbb{P}(S(T', \pi(\mathcal{M}_t)(T')) = 1) \tag{10}$$

- For any $T'' \in \mathcal{D}_{irrel}$, by Effective Retrieval (Assumption 2), the new experience E_{t+1} will not be retrieved or will be correctly identified as irrelevant, so it does not affect the plan:

$$\pi(\mathcal{M}_{t+1})(T'') = \pi(\mathcal{M}_t)(T'') \tag{11}$$

Therefore, the success probability remains unchanged:

$$\mathbb{P}(S(T'', \pi(\mathcal{M}_{t+1})(T'')) = 1) = \mathbb{P}(S(T'', \pi(\mathcal{M}_t)(T'')) = 1) \tag{12}$$

Since the probability of success is greater than or equal to the previous probability for all tasks $T \in \mathcal{D}$, the expectation over the entire distribution must also be greater than or equal.

$$\int_{T \in \mathcal{D}} \mathbb{P}(S(T, \pi(\mathcal{M}_{t+1})) = 1)p(T)dT \geq \int_{T \in \mathcal{D}} \mathbb{P}(S(T, \pi(\mathcal{M}_t)) = 1)p(T)dT \tag{13}$$

Thus, $Q(\pi(\mathcal{M}_{t+1})) \geq Q(\pi(\mathcal{M}_t))$. The inequality becomes strict if the set of relevant future tasks $\mathcal{D}_{rel}$ has a non-zero measure and the experience E_{t+1} provides a strictly positive benefit for at least some of those tasks. This concludes the proof. We acknowledge that the assumptions of perfect reflection and retrieval are strong simplifications. In practice, a poorly generalized experience could potentially degrade performance, and irrelevant context might not be perfectly ignored.

D Dataset and Benchmark Details

D.1 EHRFlowBench Construction Details

EHRFlowBench is designed to provide a rigorous and realistic testbed for evaluating AI agents on complex healthcare research tasks. Its construction follows a systematic, two-stage procedure to ensure task relevance, diversity, and grounding in established scientific literature.

Stage 1: Candidate paper screening. The process begins with a comprehensive collection of research papers from top-tier AI and data mining conferences published between 2020 and 2025: AAAI, ICLR, ICML, NeurIPS, IJCAI, KDD, and WWW. This initial pool comprises 51,280 papers, with a breakdown by conference and year detailed in Table 5. To identify papers relevant to AI applications on EHR data, a common focus in real-world healthcare research, we employ a majority-voting ensemble of three powerful LLMs (DeepSeek-V3, DeepSeek-R1, and Qwen3-235B) to classify paper titles based on their relevance to the EHR field. The prompt used for this classification is detailed below. This automated screening yields 162 candidate papers.

Table 5. Number of papers collected from selected conferences (2020–2025) for the initial screening phase of EHRFlowBench construction.

Conference	2020	2021	2022	2023	2024	2025	Total
AAAI	1646	1654	1319	1578	2331	3028	11556
IJCAI	779	723	862	851	1048	–	4263
KDD	217	239	253	310	408	185	1612
WWW	219	355	367	402	428	440	2211
NeurIPS	1898	2334	2671	3218	4035	–	14156
ICML	1085	1183	1234	1828	2610	–	7940
ICLR	687	860	1094	1573	2260	3068	9542
Sum	**6531**	**7348**	**7800**	**9760**	**13120**	**6721**	**51280**

Prompt for LLM-based screening of paper titles.

```
You are an AI research assistant specializing in scientific literature. Your task is to identify
research papers focused on applying AI to Electronic Health Records (EHR).

**Classification Criteria:**
A paper is **relevant (1)** if its title indicates the use of AI, machine learning, or data science
techniques on data explicitly from **Electronic Health Records (EHR)**. This includes:
- Clinical notes (free-text)
- Structured data (diagnosis codes like ICD, procedure codes)
- Time-series data from EHR (lab results, vital signs)

A paper is **NOT relevant (0)** if the title suggests a focus on:
- Medical Imaging (e.g., MRI, CT scans, X-rays, pathology slides)
- Genomics, proteomics, or any '-omics' data.
- Public health policy, hospital administration, or bioinformatics without direct patient-level EHR data
analysis.
- Drug discovery or molecular modeling.
- Signal processing of physiological signals like ECG, EEG, unless contextually tied to an EHR system
analysis.
- The term "administrative claims" or "billing data" alone, as this often lacks clinical depth.

**Your Task:**
Review the following list of paper titles.

---
{{numbered_titles}}
---

**Output Format:**
Respond with a single, valid JSON object. This object must have one key: "selected_indices". The value
must be an array of integers representing the 1-based index of the titles you identified as relevant.

**Example:**
If you determine that papers 2, 5, and 19 are relevant, your response MUST be exactly:
```json
{"selected_indices": [2, 5, 19]}
```

Do not provide any explanations, apologies, or any text outside of the JSON object.
```

Subsequently, these candidates undergo a manual review by human experts, who verify their relevance and methodology. This step culminates in a final set of 118 high-quality seed papers for task extraction, which are

enumerated below.

> ### *The complete list of 118 seed papers selected for EHRFlowBench task extraction.*
>
> 1. [AAAI, 2020] Generative Adversarial Regularized Mutual Information Policy Gradient Framework for Automatic Diagnosis
> 2. [AAAI, 2020] AdaCare: Explainable Clinical Health Status Representation Learning via Scale-Adaptive Feature Extraction and Recalibration
> 3. [AAAI, 2020] ConCare: Personalized Clinical Feature Embedding via Capturing the Healthcare Context
> 4. [AAAI, 2020] DeepAlerts: Deep Learning Based Multi-Horizon Alerts for Clinical Deterioration on Oncology Hospital Wards
> 5. [AAAI, 2020] Learning the Graphical Structure of Electronic Health Records with Graph Convolutional Transformer
> 6. [AAAI, 2020] Deep Mixed Effect Model Using Gaussian Processes: A Personalized and Reliable Prediction for Healthcare
> 7. [AAAI, 2020] Learning Conceptual-Contextual Embeddings for Medical Text
> 8. [AAAI, 2020] Understanding Medical Conversations with Scattered Keyword Attention and Weak Supervision from Responses
> 9. [AAAI, 2020] ICD Coding from Clinical Text Using Multi-Filter Residual Convolutional Neural Network
> 10. [AAAI, 2020] A System for Medical Information Extraction and Verification from Unstructured Text
> 11. [AAAI, 2021] GRASP: Generic Framework for Health Status Representation Learning Based on Incorporating Knowledge from Similar Patients
> 12. [AAAI, 2021] Clinical Risk Prediction with Temporal Probabilistic Asymmetric Multi-Task Learning
> 13. [AAAI, 2021] ESCAPED: Efficient Secure and Private Dot Product Framework for Kernel-based Machine Learning Algorithms with Applications in Healthcare
> 14. [AAAI, 2021] MUFASA: Multimodal Fusion Architecture Search for Electronic Health Records
> 15. [AAAI, 2021] MTAAL: Multi-Task Adversarial Active Learning for Medical Named Entity Recognition and Normalization
> 16. [AAAI, 2021] Clinical Temporal Relation Extraction with Probabilistic Soft Logic Regularization and Global Inference
> 17. [AAAI, 2022] Diaformer: Automatic Diagnosis via Symptoms Sequence Generation
> 18. [AAAI, 2022] Context-Aware Health Event Prediction via Transition Functions on Dynamic Disease Graphs
> 19. [AAAI, 2022] Clustering Interval-Censored Time-Series for Disease Phenotyping
> 20. [AAAI, 2023] Heterogeneous Graph Learning for Multi-Modal Medical Data Analysis
> 21. [AAAI, 2023] KerPrint: Local-Global Knowledge Graph Enhanced Diagnosis Prediction for Retrospective and Prospective Interpretations
> 22. [AAAI, 2023] Multi-Label Few-Shot ICD Coding as Autoregressive Generation with Prompt
> 23. [AAAI, 2023] Estimating Treatment Effects from Irregular Time Series Observations with Hidden Confounders
> 24. [AAAI, 2023] Estimating Average Causal Effects from Patient Trajectories
> 25. [AAAI, 2023] Causal Recurrent Variational Autoencoder for Medical Time Series Generation
> 26. [AAAI, 2023] MHCCL: Masked Hierarchical Cluster-Wise Contrastive Learning for Multivariate Time Series
> 27. [AAAI, 2023] Forecasting with Sparse but Informative Variables: A Case Study in Predicting Blood Glucose
> 28. [AAAI, 2024] PromptMRG: Diagnosis-Driven Prompts for Medical Report Generation
> 29. [AAAI, 2024] Automatic Radiology Reports Generation via Memory Alignment Network
> 30. [AAAI, 2024] KG-TREAT: Pre-training for Treatment Effect Estimation by Synergizing Patient Data with Knowledge Graphs
> 31. [AAAI, 2024] Inducing Clusters Deep Kernel Gaussian Process for Longitudinal Data
> 32. [AAAI, 2024] ConSequence: Synthesizing Logically Constrained Sequences for Electronic Health Record Generation
> 33. [AAAI, 2024] IGAMT: Privacy-Preserving Electronic Health Record Synthesization with Heterogeneity and Irregularity
> 34. [AAAI, 2024] IVP-VAE: Modeling EHR Time Series with Initial Value Problem Solvers
> 35. [AAAI, 2024] DrFuse: Learning Disentangled Representation for Clinical Multi-Modal Fusion with Missing Modality and Modal Inconsistency
> 36. [AAAI, 2024] Large Language Models Are Clinical Reasoners: Reasoning-Aware Diagnosis Framework with Prompt-Generated Rationales
> 37. [AAAI, 2024] Collaborative Synthesis of Patient Records through Multi-Visit Health State Inference
> 38. [AAAI, 2025] DAMPER: A Dual-Stage Medical Report Generation Framework with Coarse-Grained MeSH Alignment and Fine-Grained Hypergraph Matching
> 39. [AAAI, 2025] Memorize and Rank: Elevating Large Language Models for Clinical Diagnosis Prediction
> 40. [AAAI, 2025] DECT: Harnessing LLM-assisted Fine-Grained Linguistic Knowledge and Label-Switched and Label-Preserved Data Generation for Diagnosis of Alzheimer's Disease
> 41. [AAAI, 2025] Medical Manifestation-Aware De-Identification
> 42. [ICLR, 2024] GraphCare: Enhancing Healthcare Predictions with Personalized Knowledge Graphs.
> 43. [ICLR, 2024] Diagnosing Transformers: Illuminating Feature Spaces for Clinical Decision-Making.
> 44. [ICLR, 2024] A Flexible Generative Model for Heterogeneous Tabular EHR with Missing Modality.
> 45. [ICLR, 2024] Graph Transformers on EHRs: Better Representation Improves Downstream Performance.
> 46. [ICLR, 2025] Small Models are LLM Knowledge Triggers for Medical Tabular Prediction.
> 47. [ICLR, 2025] Efficiently Democratizing Medical LLMs for 50 Languages via a Mixture of Language Family Experts.
> 48. [ICML, 2020] BoXHED: Boosted eXact Hazard Estimator with Dynamic covariates.
> 49. [ICML, 2020] Temporal Phenotyping using Deep Predictive Clustering of Disease Progression.

50. [ICML, 2020] DeepCoDA: personalized interpretability for compositional health data.
51. [ICML, 2021] Neighborhood Contrastive Learning Applied to Online Patient Monitoring.
52. [ICML, 2022] Locally Sparse Neural Networks for Tabular Biomedical Data.
53. [ICML, 2022] Learning of Cluster-based Feature Importance for Electronic Health Record Time-series.
54. [ICML, 2023] Improving Medical Predictions by Irregular Multimodal Electronic Health Records Modeling.
55. [ICML, 2023] Sequential Multi-Dimensional Self-Supervised Learning for Clinical Time Series.
56. [ICML, 2024] Contrastive Learning for Clinical Outcome Prediction with Partial Data Sources.
57. [ICML, 2024] Reservoir Computing for Short High-Dimensional Time Series: an Application to SARS-CoV-2 Hospitalization Forecast.
58. [ICML, 2024] Exploiting Negative Samples: A Catalyst for Cohort Discovery in Healthcare Analytics.
59. [ICML, 2024] ED-Copilot: Reduce Emergency Department Wait Time with Language Model Diagnostic Assistance.
60. [ICML, 2024] ProtoGate: Prototype-based Neural Networks with Global-to-local Feature Selection for Tabular Biomedical Data.
61. [IJCAI, 2020] A Label Attention Model for ICD Coding from Clinical Text
62. [IJCAI, 2020] The Graph-based Mutual Attentive Network for Automatic Diagnosis
63. [IJCAI, 2020] Automatic Emergency Diagnosis with Knowledge-Based Tree Decoding
64. [IJCAI, 2020] Learning Latent Forests for Medical Relation Extraction
65. [IJCAI, 2020] Generalized Zero-Shot Text Classification for ICD Coding
66. [IJCAI, 2021] Cooperative Joint Attentive Network for Patient Outcome Prediction on Irregular Multi-Rate Multivariate Health Data
67. [IJCAI, 2021] AMA-GCN: Adaptive Multi-layer Aggregation Graph Convolutional Network for Disease Prediction
68. [IJCAI, 2021] Collaborative Graph Learning with Auxiliary Text for Temporal Event Prediction in Healthcare
69. [IJCAI, 2021] Multi-series Time-aware Sequence Partitioning for Disease Progression Modeling
70. [IJCAI, 2021] A Novel Sequence-to-Subgraph Framework for Diagnosis Classification
71. [IJCAI, 2022] Cumulative Stay-time Representation for Electronic Health Records in Medical Event Time Prediction
72. [IJCAI, 2022] ``My nose is running.'' ``Are you also coughing?'': Building A Medical Diagnosis Agent with Interpretable Inquiry Logics
73. [IJCAI, 2022] Chronic Disease Management with Personalized Lab Test Response Prediction
74. [IJCAI, 2022] Data-Efficient Algorithms and Neural Natural Language Processing: Applications in the Healthcare Domain
75. [IJCAI, 2023] Hierarchical Apprenticeship Learning for Disease Progression Modeling
76. [IJCAI, 2023] A Diffusion Model with Contrastive Learning for ICU False Arrhythmia Alarm Reduction
77. [IJCAI, 2023] VecoCare: Visit Sequences-Clinical Notes Joint Learning for Diagnosis Prediction in Healthcare Data
78. [IJCAI, 2024] Predictive Modeling with Temporal Graphical Representation on Electronic Health Records
79. [IJCAI, 2024] Multi-TA: Multilevel Temporal Augmentation for Robust Septic Shock Early Prediction
80. [IJCAI, 2024] MediTab: Scaling Medical Tabular Data Predictors via Data Consolidation, Enrichment, and Refinement
81. [KDD, 2020] Hierarchical Attention Propagation for Healthcare Representation Learning
82. [KDD, 2020] INPREM: An Interpretable and Trustworthy Predictive Model for Healthcare
83. [KDD, 2020] DETERRENT: Knowledge Guided Graph Attention Network for Detecting Healthcare Misinformation
84. [KDD, 2020] HiTANet: Hierarchical Time-Aware Attention Networks for Risk Prediction on Electronic Health Records
85. [KDD, 2020] Identifying Sepsis Subphenotypes via Time-Aware Multi-Modal Auto-Encoder
86. [KDD, 2020] HOLMES: Health OnLine Model Ensemble Serving for Deep Learning Models in Intensive Care Units
87. [KDD, 2022] M3Care: Learning with Missing Modalities in Multimodal Healthcare Data
88. [KDD, 2023] Granger Causal Chain Discovery for Sepsis-Associated Derangements via Continuous-Time Hawkes Processes
89. [KDD, 2023] MedLink: De-Identified Patient Health Record Linkage
90. [KDD, 2023] Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series
91. [KDD, 2024] FlexCare: Leveraging Cross-Task Synergy for Flexible Multimodal Healthcare Prediction
92. [KDD, 2024] ProtoMix: Augmenting Health Status Representation Learning via Prototype-based Mixup
93. [KDD, 2024] Synthesizing Multimodal Electronic Health Records via Predictive Diffusion Models
94. [NeurIPS, 2020] Learning to Select Best Forecast Tasks for Clinical Outcome Prediction
95. [NeurIPS, 2021] Medical Dead-ends and Learning to Identify High-Risk States and Treatments
96. [NeurIPS, 2021] Auto-Encoding Knowledge Graph for Unsupervised Medical Report Generation
97. [NeurIPS, 2021] Closing the loop in medical decision support by understanding clinical decision-making: A case study on organ transplantation
98. [NeurIPS, 2023] Contrast Everything: A Hierarchical Contrastive Framework for Medical Time-Series
99. [NeurIPS, 2023] Temporal Causal Mediation through a Point Process: Direct and Indirect Effects of Healthcare Interventions
100. [NeurIPS, 2023] Towards Semi-Structured Automatic ICD Coding via Tree-based Contrastive Learning
101. [NeurIPS, 2024] Medformer: A Multi-Granularity Patching Transformer for Medical Time-Series Classification
102. [NeurIPS, 2024] SMART: Towards Pre-trained Missing-Aware Model for Patient Health Status Prediction
103. [NeurIPS, 2024] Knowledge-Empowered Dynamic Graph Network for Irregularly Sampled Medical Time Series
104. [NeurIPS, 2024] Trajectory Flow Matching with Applications to Clinical Time Series Modelling

```
105. [NeurIPS, 2024] A teacher-teacher framework for clinical language representation learning
106. [NeurIPS, 2024] Automated Multi-Task Learning for Joint Disease Prediction on Electronic Health
Records
107. [WWW, 2020] Text-to-SQL Generation for Question Answering on Electronic Medical Records
108. [WWW, 2020] StageNet: Stage-Aware Neural Networks for Health Risk Prediction
109. [WWW, 2020] CLARA: Clinical Report Auto-completion
110. [WWW, 2020] Learning Contextualized Document Representations for Healthcare Answer Retrieval
111. [WWW, 2020] DyCRS: Dynamic Interpretable Postoperative Complication Risk Scoring
112. [WWW, 2021] Online Disease Diagnosis with Inductive Heterogeneous Graph Convolutional Networks
113. [WWW, 2021] Distilling Knowledge from Publicly Available Online EMR Data to Emerging Epidemic for
Prognosis
114. [WWW, 2023] SeqCare: Sequential Training with External Medical Knowledge Graph for Diagnosis
Prediction in Healthcare Data
115. [WWW, 2023] Cross-center Early Sepsis Recognition by Medical Knowledge Guided Collaborative
Learning for Data-scarce Hospitals
116. [WWW, 2025] ColaCare: Enhancing Electronic Health Record Modeling through Large Language
Model-Driven Multi-Agent Collaboration
117. [WWW, 2025] MedRAG: Enhancing Retrieval-augmented Generation with Knowledge Graph-Elicited
Reasoning for Healthcare Copilot
118. [WWW, 2025] Towards Multi-resolution Spatiotemporal Graph Learning for Medical Time Series
Classification
```

Stage 2: Task extraction and curation. We then utilize an LLM (DeepSeek-V3) to extract evidence-grounded tasks from the full text of these 118 papers. For each paper, the LLM is prompted to generate a detailed task description, a primary research category (e.g., "Data Analysis", "Algorithm Implementation"), and a reference answer or expected outcome based on the paper's findings. The prompt for this process is provided below. This semi-automated process yields an initial pool of 585 tasks. To create a well-balanced and non-redundant benchmark, these 585 tasks undergo a final manual curation phase. Our team of researchers consolidates semantically similar task types, discards irrelevant or hard-to-evaluate categories (e.g., "Ablation Study"), and refines task descriptions for clarity. We apply stratified sampling to ensure diversity across research domains, retaining all tasks from smaller categories and selecting a representative subset from larger ones. This meticulous process results in the final EHRFlowBench, which contains 110 high-quality tasks: 100 for evaluation and 10 (one from each of the 10 final categories) reserved for the training set used to bootstrap HealthFlow's experience memory.

> *Prompt for LLM-based extraction of research tasks from papers.*
>
> ```
> ### CORE MISSION ###
> Please dissect the provided research paper and generate a series of self-contained, complex
> "mini-projects." These projects will be used to evaluate an advanced AI agent's ability to implement
> algorithms and reproduce scientific findings. You will generate approximately 5 such projects.
>
> ### ABSOLUTE RULES FOR TASK GENERATION (NON-NEGOTIABLE) ###
>
> 1. **ZERO-REFERENCE MANDATE:** The task description MUST be entirely self-contained. It must NOT
> reference the source paper in any way (e.g., "as described in Section 3," "using Equation (5)," "from
> Table 2"). The AI agent performing the task will NOT have access to the paper. All necessary
> information, formulas, parameters, and constants must be explicitly defined within the `<task>`.
>
>     *   **FAILURE EXAMPLE (DO NOT DO THIS):** "Implement the time-aware graph attention mechanism from
>     equations (1)-(4)."
>     *   **SUCCESS EXAMPLE (DO THIS):** "Implement a time-aware graph attention mechanism. Given a head
>     entity \(c_h\), its neighbors \(N_h\), and a time interval \(\tau\), compute the aggregated neighbor
>     representation \(e_{N_h}\). First, compute the time embedding \(f_\tau = \tanh(W_f \tau + b_f)\).
>     Then, for each neighbor \(c_u \in N_h\), calculate the attention score \(\pi(c_h, r, c_u, \tau)\)
>     using a feed-forward network: \(\text{FFN}(M_r e_h \| M_r e_u \| f_\tau)\), where \(\|\|\) denotes
>     concatenation. Normalize these scores using softmax to get \(\tilde{\pi}\). Finally, compute
>     \(e_{N_h} = \sum_{c_u \in N_h} \tilde{\pi}(c_h, r, c_u, \tau) e_u\). Use parameter dimensions:
>     \(e_h, e_u \in \mathbb{R}^{100}\), \(M_r \in \mathbb{R}^{100 \times 100}\), \(W_f \in
>     \mathbb{R}^{64}\), \(b_f \in \mathbb{R}^{64}\)."
>
> 2. **MANDATORY LATEX FOR ALL MATH:** All mathematical variables, formulas, and expressions MUST be
> formatted using LaTeX syntax.
>     *   For inline math, use `\( ... \)`. Example: The loss is calculated for each sample \(i\).
>     *   For block/display math, use `\[ ... \]`. Example: \[ L_{\text{total}} = \sum_{i=1}^{N} (y_i -
>     \hat{y}_i)^2 \]
> ```

```
3.  **DIVERSE & SUBSTANTIAL TASKS:** Generate a variety of tasks covering different research stages
(e.g., Cohort Definition, Feature Engineering, Model Implementation, etc.). Each task should be a
meaningful unit of work, not a trivial query.

4.  **VERIFIABLE ANSWER:** The `<answer>` field must contain the specific, verifiable result from the
paper that directly corresponds to the completion of the task. The answer must also include a brief
interpretation of the result's significance within the context of the study. Use LaTeX for any math in
the answer.

### XML OUTPUT SPECIFICATION (STRICTLY ENFORCED) ###
Your entire output must be a single, valid XML block. Do not include any text, explanations, or markdown
fences before or after the XML. The root element must be `<response>`. Each task must be enclosed in an
`<item>` tag with exactly these three child tags: `<category>`, `<task>`, and `<answer>`:

1. `<category>`: A descriptive category for the task.
2. `<task>`: The detailed, self-contained, imperative instructions for the AI agent, following all rules
above. **All math must be in LaTeX.**
3. `<answer>`: The verifiable result from the paper. This should contain the specific value/outcome and
a brief sentence explaining its context. **Any math must be in LaTeX.**

--- BEGIN RESEARCH PAPER TEXT ---
{{paper_text}}
--- END RESEARCH PAPER TEXT ---
```

An example of a task extracted and curated for EHRFlowBench is provided below.

Example task from the EHRFlowBench benchmark.

```
<item>
  <category>Cohort Definition</category>
  <task>
    You are given the patient statistics for three distinct disease cohorts used in a risk prediction
    study. Your task is to perform a calculation for the "Heart Failure" cohort.

    **Cohort Data:**
    1.  **COPD Cohort:**
        *   Case (Positive) Patients: 7,314
        *   Control (Negative) Patients: 21,942
    2.  **Heart Failure Cohort:**
        *   Case (Positive) Patients: 3,080
        *   Control (Negative) Patients: 9,240
    3.  **Kidney Disease Cohort:**
        *   Case (Positive) Patients: 2,810
        *   Control (Negative) Patients: 8,430

    **Instructions:**
    1.  Identify the data for the "Heart Failure" cohort.
    2.  Calculate the total number of patients in this cohort.
    3.  Calculate the control-to-case ratio for this cohort. Express the ratio as a single number (e.g.,
    if there are 100 controls and 50 cases, the ratio is 2.0).
    4.  Report the average number of visits per patient for this cohort, which is given as 38.74.
  </task>
  <answer>
    For the Heart Failure cohort:
    - Total Patients: 3,080 (Case) + 9,240 (Control) = 12,320
    - Control-to-Case Ratio: 9,240 / 3,080 = 3.0
    - Average Visits Per Patient: 38.74

    **Interpretation:** The dataset for Heart Failure is imbalanced with a 3:1 ratio of control to case
    patients. This is a common characteristic in medical datasets and must be accounted for during model
    training and evaluation.
  </answer>
</item>
```

D.2 External Datasets and Benchmarks Used

To ensure a comprehensive evaluation of HealthFlow's capabilities, we utilize several publicly available datasets and benchmarks, each targeting distinct aspects of medical reasoning and data analysis.

Motivation for dataset selection. Our selection of benchmarks is driven by the need to assess the multifaceted capabilities required for autonomous healthcare research. EHRFlowBench serves as our primary benchmark for

evaluating end-to-end, complex research workflows. MedAgentBoard is chosen to test practical skills in structured EHR data processing and modeling. MedAgentsBench and the medical subset of Humanity's Last Exam (HLE) are included to measure foundational medical knowledge and expert-level reasoning, which are prerequisites for high-quality analysis. Finally, CureBench is incorporated to specifically evaluate tool-augmented reasoning in the context of clinical therapeutic decision-making. This combination allows for a holistic assessment of our agent, from core knowledge to practical application and complex research execution.

Dataset descriptions and availability. All datasets employed in this paper are publicly available or accessible upon request and are used under their respective data use agreements.

- **EHRFlowBench**: This benchmark is introduced in this work. Details are provided in Section D.1. The complete dataset is available in the GitHub repository.

- **MedAgentBoard** [32]: A benchmark designed to evaluate AI agents on practical, multi-step data science tasks using real-world EHR data (MIMIC-IV [36, 37] and TJH [38]). It assesses the entire pipeline from data extraction and cohort definition to predictive modeling and report generation. The benchmark is publicly available on GitHub.

- **MedAgentsBench** [33]: A benchmark consisting of challenging multiple-choice questions designed to test the medical knowledge and clinical reasoning abilities of AI agents across various medical specialties. We randomly select 100 questions from its "hard set" using a random seed of 42.

- **Humanity's Last Exam (HLE)** [34]: A collection of expert-level problems designed to be exceptionally challenging for current AI systems. We use 45 text-only questions from the "Medicine" and "Health Science" subcategories within its "Biology/Medicine" category to assess specialized domain reasoning under difficult conditions.

- **CureBench** [35]: A benchmark for therapeutic decision-making that tasks agents with complex reasoning over patients, diseases, and drugs, requiring the use of external biomedical tools like FDA databases and PubMed. The data can be downloaded from the Data section of the associated Kaggle competition, accessible via the official website: https://curebench.ai/.

E   Experimental Setup Details

E.1   Computing Infrastructure

All experiments are conducted on a consistent hardware and software platform to ensure reproducibility.

- **Hardware**: Apple Mac Studio with an M3 Ultra chip and 512 GB of unified memory.

- **Software**: We develop HealthFlow using Python 3.12. The environment is managed using `uv`, with all dependencies specified in a `pyproject.toml` file for reproducibility.

- **Locally deployed models**: The HuatuoGPT-o1 (`QuantFactory/HuatuoGPT-o1-7B-GGUF` (8-bit quantized version)) and MedGemma (`google/medgemma-27b-text-it` (8-bit quantized version)) models are deployed and run locally using LMStudio on the Mac Studio.

- **Online LLM APIs**: Calls to online LLM APIs (e.g., DeepSeek, Qwen, Claude, Kimi, and GLM) utilize their respective official platforms. For models that support it, we leverage an OpenAI-compatible API interface for standardized communication.

E.2   Hyperparameter Details

The hyperparameters for our framework are selected to balance performance, computational cost, and stability, based on preliminary experiments on an internal validation set.

Key hyperparameters for HealthFlow are tuned as follows:

- **Maximum retries**: We test values of 1, 2, and 3. A value of 1 corresponds to a single attempt with no self-correction. Our experiments indicate that a maximum of 3 trials provides a good balance between performance and cost, allowing the agent sufficient opportunity to correct initial failures without excessive computational overhead.

- **Success threshold**: The evaluator agent as an LLM judge to assigns the evaluation score, where this parameter determines the minimum evaluation score (on a 1.0–10.0 scale) required to consider a task successful and halt the retry loop. We evaluate thresholds of 6.0 and 8.0. We select a threshold of 6.0, as a value of 8.0 leads to an unnecessarily high number of retries for tasks that are already functionally complete, thereby reducing overall efficiency.

For all external LLM API calls within our framework and all baselines, we use the default parameters provided by the respective services (e.g., for `temperature`, `top_p`, and `max_tokens`). This approach ensures that our evaluation focuses on the performance of the agentic architecture rather than on the effects of hyperparameter tuning for the backbone models.

E.3 Implementation of Baseline Methods

For a fair and rigorous comparison, all baseline methods are configured to the best of our ability according to their original papers and public implementations. Unless otherwise specified, all agent frameworks are powered by the DeepSeek-V3 model to normalize for the effect of the backbone LLM and isolate the performance of the agentic architecture itself.

General and medical LLMs. DeepSeek-V3 [40] and DeepSeek-R1 [41] are accessed via their official APIs. HuatuoGPT-o1 [42] and MedGemma [43] are run locally using LMStudio. These models are prompted with the raw task description and are evaluated based on their direct, single-turn responses without any agentic scaffolding.

Multi-agent collaboration frameworks. We use the official open-source implementations for MedAgents [44], MDAgents [45], and ColaCare [46]. As these frameworks are designed primarily for conversational QA or summarization and often lack robust, sandboxed code execution environments, we adapt their inputs to be the research tasks from our benchmarks and evaluate their final textual outputs.

General and biomedical agent frameworks. All general and medical LLMs, as well as multi-agent collaboration frameworks, lack the capability to read and execute code. Therefore, for open-ended tasks such as those in EHRFlowBench and MedAgentBoard, we evaluate only their generated code and answers.

We use the official repositories for AFlow [19], Biomni [12], and STELLA [9], reproducing results according to the instructions provided in their README files. For Alita [13], whose official implementation is not public, we adopt the widely used open-source community version, OpenAlita [51]. All frameworks are configured to use DeepSeek-V3 as their primary reasoning and execution model to ensure a fair comparison with HealthFlow.

E.4 Evaluation Metrics Details

We employ a combination of automated metrics and LLM-based evaluation to assess agent performance across different benchmarks.

Standard metrics. For benchmarks with ground-truth answers, we use standard metrics.

- **Accuracy**: This is used for MedAgentsBench and HLE, where tasks are multiple-choice or have a binary correct/incorrect answer. It is the percentage of correctly answered questions. We first use LLM-based extraction to obtain the predicted answer and then apply the official evaluation prompt to determine correctness.

- **Success rate**: The primary metric for MedAgentBoard. A task is considered successful if the agent produces the correct final numerical answer or a functionally correct artifact (e.g., a plot), as verified against the ground-truth solution through manual human evaluation.

LLM-as-a-judge evaluation.

- For **EHRFlowBench**, we use an ensemble of five diverse LLMs (DeepSeek-V3, DeepSeek-R1, Claude-4-Sonnet, Kimi-K2, GLM-4.5) to mitigate single-model bias. Each judge scores a solution on a 1–5 scale across three dimensions: methodology soundness (70% weight), presentation quality (20%), and artifact generation (10%). The final score is the weighted average of the mean scores from the five judges. The evaluation prompt is detailed below.

1. method_soundness

Evaluate the overall quality of the solution path. Assess the chosen method's soundness and correctness. More importantly, **this score heavily weights the quality and completeness of the agent's justification for its approach**. A well-reasoned, comprehensive report that discusses problem framing, results, and limitations should score highly, even if its method intelligently deviates from the reference answer. The reference serves as a benchmark, not a strict mandate.

*   **5 (Exemplary):** **MUST meet all criteria for a score of 4.** In addition, the **Methodology Justification** is exceptionally insightful, comparing the approach to viable alternatives and providing a profound analysis of the **generated results** and limitations.
*   **4 (Strong):** **A score of 4 is IMPOSSIBLE without clear evidence of successful execution** (explicit statements OR output files like `.png`/`.csv` in `{file_structure_info}`). The response must also contain a true **Methodology Justification** (as defined above), not just a code explanation.
*   **3 (Acceptable):** **This is the ABSOLUTE MAXIMUM score for any solution that LACKS execution evidence.** To achieve this score, the response MUST provide a strong **Methodology Justification** (the "Why"). It presents a well-reasoned strategic plan, even if it's not proven with results.
*   **2 (Weak):** **Assign this score if a response's justification consists ONLY of code and a Code Explanation (the "What" and "How"), like the provided example.** This score is the correct rating for a submission that presents a function or script but fails to provide the strategic "Why" behind it. This score also applies if the method is flawed or execution failed.
*   **1 (Poor):** The method is fundamentally flawed or irrelevant, and there is no meaningful justification of any kind.

2. presentation_quality

Clarity, structure, and completeness of the final answer's explanation and formatting. Is it easy to read and understand?

*   **5 (Exemplary):** The presentation is exceptionally clear, professional, and well-structured. It uses formatting, language, and structure to make complex information easy to digest.
*   **4 (Strong):** The presentation is clear, well-structured, and complete. It is easy to read and understand with only minor room for refinement.
*   **3 (Acceptable):** The core message is communicated clearly and the structure is adequate. A reader can understand the answer, though it may have minor issues with clarity, organization, or formatting.
*   **2 (Weak):** The presentation is disorganized, unclear, or incomplete, making it difficult for the reader to follow.
*   **1 (Poor):** The presentation is confusing, unstructured, or poorly formatted, failing to communicate the information effectively.

3. artifact_generation

Functionality, correctness, and completeness of generated files (e.g., code, plots, data). Are they usable and aligned with the task?

*   **5 (Exemplary):** Artifacts are not only correct and well-organized into files but also demonstrate exemplary software engineering practices. The code architecture is robust and clean (e.g., using functions/classes), well-commented, and easily understandable or reusable.
*   **4 (Strong):** Generates correct and functional artifacts. The code is well-organized, logically structured, and is appropriately saved into distinct files (e.g., `.py` for code, `.csv` for data).
*   **3 (Acceptable):** Artifacts are generated and are largely functional. The code may require minor corrections to run, ript), but it successfully implements the core logic.
*   **2 (Weak):** Artifacts are generated but contain significant errors or are incomplete. This score recognizes the attempt to generate code but notes its flaws.
*   **1 (Poor):** Fails to generate the required artifacts.

4. overall_score

Your holistic assessment, **not a simple average**. A critical failure (score of 0 or 1) in one key area should heavily penalize the overall score. Conversely, exceptional performance (4 or 5) in key areas should elevate it.

*   **5 (Exemplary):** An exemplary performance (score of 5) on `solution_approach_justification` with at least strong performance (4) on other dimensions. A model answer.
*   **4 (Strong):** Strong performance (4 or higher) across all key dimensions, or an exemplary performance in one key area balanced by acceptable performance elsewhere. A high-quality submission.
*   **3 (Acceptable):** Meets expectations (score of 3) on all dimensions. A solid, complete solution with no major flaws.
*   **2 (Weak):** Weak performance (score of 2) in one or more dimensions, especially `solution_approach_justification` or `artifact_generation`, but is not a complete failure.
*   **1 (Poor):** Contains a fundamental flaw or critical error (score of 1) in a key dimension, severely compromising the value of the submission.

- For **MedAgentBoard**, we use a single LLM (DeepSeek-V3) to provide a 1–5 score across four task-specific dimensions. Evaluators follow detailed scoring rubrics customized for each task category, as detailed in the prompts below.

    - **Data extraction and statistical analysis**: Accuracy of data selection, transformation logic, handling of missing values, and appropriateness of statistical methods.

    > *Evaluation prompt for data extraction and statistical analysis in MedAgentBoard.*
    >
    > ```
    > ### 1. Correctness of Data Selection (1-5 points)
    > - **5 (Perfect):** The selected data subset (rows and columns) is **exactly** what is required by
    > the task and perfectly matches the standard answer.
    > - **4 (Minor Deviation):** The core data is correct, but there are minor discrepancies, such as an
    > extra non-essential column or a slightly different but still valid filter condition.
    > - **3 (Partial Match):** The selected data has some correct elements but misses significant
    > portions of required data or includes a large amount of incorrect data. E.g., correct rows but
    > wrong columns.
    > - **2 (Incorrect):** The data selection logic is fundamentally flawed. The resulting dataset is
    > largely irrelevant to the task.
    > - **1 (Critically Flawed):** No data is selected, or the selected data is completely wrong.
    >
    > ### 2. Transformation Logic (1-5 points)
    > - **5 (Perfect):** All data transformations (e.g., calculations, aggregations, reshaping) are
    > implemented correctly and efficiently, yielding results identical to the standard answer.
    > - **4 (Mostly Correct):** The logic is sound and achieves the correct outcome, but with minor
    > inefficiencies or stylistic differences from the ideal implementation. The final numbers are
    > correct.
    > - **3 (Partially Correct):** The transformation logic contains errors that lead to partially
    > incorrect results. For example, a calculation is wrong, or an aggregation is performed on the wrong
    > group.
    > - **2 (Incorrect):** The transformation logic is fundamentally incorrect and does not produce the
    > required output format or values.
    > - **1 (Critically Flawed):** The code for transformation is non-functional, absent, or completely
    > irrelevant.
    >
    > ### 3. Handling of Missing Values (1-5 points)
    > - **5 (Perfect):** Missing values are handled appropriately as dictated by the task or best
    > practices (e.g., imputation, removal) and aligns perfectly with the standard answer's approach.
    > - **4 (Acceptable Alternative):** An alternative, valid method for handling missing values was used
    > that still leads to a correct or very similar outcome.
    > - **3 (Suboptimal Handling):** Missing values were handled, but in a way that negatively impacts
    > the final result or is not appropriate for the data type (e.g., filling categorical data with 0).
    > - **2 (Incorrect Handling):** The method for handling missing values is wrong or leads to
    > significant errors in the analysis.
    > - **1 (Critically Flawed):** Missing values were completely ignored when they should have been
    > handled, or the handling method caused the process to fail.
    >
    > ### 4. Appropriateness of Statistical Methods (1-5 points)
    > - **5 (Perfect):** The statistical methods used (e.g., mean, median, standard deviation, t-test)
    > are perfectly appropriate for the data and the question, matching the standard answer.
    > - **4 (Mostly Appropriate):** The chosen statistical method is valid and yields a correct
    > conclusion, though a slightly more optimal method might exist.
    > - **3 (Partially Appropriate):** A statistical method was used, but it was not the right choice for
    > the data distribution or task, leading to potentially misleading results.
    > - **2 (Inappropriate):** The statistical method is clearly wrong (e.g., using mean on ordinal data,
    > correlation on non-linear data).
    > - **1 (Critically Flawed):** No statistical analysis was performed, or a completely nonsensical
    > method was applied.
    > ```

    - **Predictive modeling**: Suitability of model choice, soundness of training procedures, inclusion of relevant evaluation metrics, and proper validation practices.

    > *Evaluation prompt for predictive modeling in MedAgentBoard.*
    >
    > ```
    > ### 1. Appropriateness of Model Selection (1-5 points)
    > - **5 (Perfect):** The chosen model is highly appropriate for the data type, task (e.g.,
    > classification, regression), and complexity, aligning with the standard answer.
    > - **4 (Acceptable):** The model is a reasonable choice and works, but a more standard or
    > higher-performing model is available and used in the standard answer.
    > - **3 (Suboptimal):** The chosen model is unconventional or ill-suited for the task, leading to poor
    > performance or unnecessarily complex implementation.
    > ```

```
- **2 (Incorrect):** The model type is wrong for the task (e.g., using a regression model for a
classification task).
- **1 (Critically Flawed):** No model was selected, or a completely nonsensical choice was made.

### 2. Implementation of Training Procedures (1-5 points)
- **5 (Perfect):** The model training code is bug-free, efficient, and correctly implemented,
including data splitting, feature preparation, and fitting. It matches the standard answer's
implementation.
- **4 (Mostly Correct):** The training procedure is correct, but with minor issues like suboptimal
hyperparameter defaults or inefficient data handling. The model trains successfully.
- **3 (Partially Correct):** The training code has errors that allow it to run but produce a poorly
trained model (e.g., data leakage, wrong feature scaling).
- **2 (Incorrect):** The training code has significant bugs that prevent the model from training
correctly or at all.
- **1 (Critically Flawed):** The training code is completely non-functional or absent.

### 3. Inclusion of Necessary Evaluation Metrics (1-5 points)
- **5 (Perfect):** All relevant evaluation metrics for the task (e.g., AUC, F1-score for
classification; R-squared, MSE for regression) are correctly calculated and reported, matching the
standard answer.
- **4 (Mostly Complete):** The primary metrics are reported, but some useful secondary metrics are
missing.
- **3 (Partially Complete):** Some key metrics are missing, or the reported metrics are not the
most appropriate for the task.
- **2 (Incorrect):** The wrong metrics are calculated (e.g., accuracy on a highly imbalanced
dataset without other metrics), or they are calculated incorrectly.
- **1 (Critically Flawed):** No evaluation metrics are provided.

### 4. Adherence to Proper Validation Practices (1-5 points)
- **5 (Perfect):** A proper validation strategy (e.g., train-test split, cross-validation) is used
correctly, ensuring no data leakage and providing an unbiased estimate of performance.
- **4 (Acceptable):** A simple train-test split is used correctly, where cross-validation might
have been more robust. The validation is still sound.
- **3 (Flawed):** The validation practice has flaws, such as not stratifying a split on an
imbalanced dataset or testing on data that was used in training (leakage).
- **2 (Incorrect):** The concept of validation is misunderstood. For example, the model is
evaluated on the training set.
- **1 (Critically Flawed):** There is no validation procedure at all.
```

– **Data visualization**: Correct application of visualization techniques, alignment with analytical goals, aesthetic clarity, and overall readability.

Evaluation prompt for data visualization in MedAgentBoard.

```
### 1. Correctness of Visualization Techniques (Code-based) (1-5 points)
- **5 (Perfect):** The agent's plotting code is logically identical to the standard answer's code.
It selects the correct data, uses the right plot type (e.g., bar, scatter), and correctly maps
variables to axes.
- **4 (Mostly Correct):** The plotting code produces the same type of plot with the correct data,
but may have minor differences in implementation (e.g., using a different library but achieving the
same result).
- **3 (Partially Correct):** The code attempts to create the right kind of plot but makes
significant errors, such as plotting the wrong variables, using incorrect data aggregations, or
choosing a plot type that obscures the insight.
- **2 (Incorrect):** The plotting code is fundamentally flawed and would produce a visualization
that is misleading or completely different from the required one.
- **1 (Critically Flawed):** The plotting code is non-functional, absent, or completely irrelevant.

### 2. Alignment with Analytical Objectives (1-5 points)
- **5 (Perfect):** The visualization (as inferred from the code) directly and clearly answers the
question posed in the task brief.
- **4 (Mostly Aligned):** The visualization is relevant but might not be the most effective way to
show the specific insight required.
- **3 (Partially Aligned):** The visualization shows related data but does not directly address the
core analytical objective of the task.
- **2 (Poorly Aligned):** The visualization is only tangentially related to the task.
- **1 (Critically Flawed):** The visualization is completely irrelevant to the analytical
objective.

### 3. Aesthetic Quality and Readability (Inferred) (1-5 points)
- **5 (Perfect):** The code includes clear labels for the title, x-axis, and y-axis, and a legend
if necessary. The implementation suggests a clean, professional, and easy-to-read plot.
```

```
  - **4 (Good):** The plot is mostly readable, but is missing one minor element like a title or a
  clear axis label.
  - **3 (Acceptable):** The plot is generated, but the code lacks any labels, titles, or other
  elements that aid interpretation.
  - **2 (Poor):** The code suggests a messy or confusing plot (e.g., overlapping labels, no clear
  differentiation of elements).
  - **1 (Critically Flawed):** No effort is made to make the plot interpretable.

  ### 4. Correctness of File Generation (1-5 points)
  - **5 (Perfect):** The agent correctly generated the required image file in the correct format
  (e.g., `output.png`).
  - **4 (Minor Issue):** The file was generated in a different but still acceptable image format
  (e.g., a `.jpg` instead of a `.png`).
  - **3 (Incorrect Format):** The output was saved in a non-image format (e.g., `.txt`, `.csv`).
  - **2 (Failed Generation):** The code includes a save command, but it is incorrect and would fail,
  or the generated answer indicates a failure to save.
  - **1 (Critically Flawed):** The required image file was **not generated at all**. This is a
  critical failure.
```

– **Report generation**: Completeness, coherence, factual accuracy, and clinical relevance of the synthesized findings.

F   Detailed Case Studies

F.1 Case 1: Medically Informed Data Validation

In this case study, selected from the MedAgentBoard benchmark, the task is to create a scatter plot to visualize the relationship between systolic blood pressure and diastolic blood pressure. The goal is to plot "systolic blood pressure" on the x-axis and "diastolic blood pressure" on the y-axis to explore the correlation between these two vital signs across all records.

Phase 1: Experience-driven planning by the meta agent. Upon receiving the task, the meta agent does not immediately generate code. Its first action is to query the persistent experience memory for relevant strategic knowledge. For this task, it retrieves several high-level heuristics and warnings gleaned from previous, similar analyses, as shown below. These experiences highlight critical, non-obvious requirements for handling clinical data, such as the need for early data validation and the removal of missing values to ensure analytical integrity.

Past experiences retrieved by the meta agent to inform strategic planning.

```
[
  {
    "type": "heuristic",
    "category": "medical_data_cleaning",
    "content": "For temporal analysis of clinical metrics in healthcare data, always filter out
    records with missing values in the target parameter before aggregation or visualization. This
    ensures data integrity by preventing skewed results from incomplete data, enhancing reliability
    for similar requests involving trend analysis like plotting lab values over time."
  },
  {
    "type": "warning",
    "category": "data_quality",
    "content": "When handling healthcare data with multiple records per patient, failing to validate
    column existence early or neglecting to remove records with missing values in target variables can
    lead to runtime failures or distorted visualizations; consistently implement these checks to
    uphold data reliability in similar visualization requests."
  },
  {
    "type": "warning",
    "category": "data_validation",
    "content": "In numerical threshold analyses for clinical data, always validate that columns are of
    numeric data types before performing comparisons. Omission can cause runtime errors or incorrect
    interpretations if data contains non-numeric entries, potentially compromising analysis accuracy.
    Incorporate data type checks early in processing to improve robustness and avoid pitfalls in
    similar healthcare tasks."
  }
]
```

Informed by this retrieved knowledge, the meta agent synthesizes these abstract principles into a concrete, executable plan, which is detailed next. This plan is not merely a sequence of commands; it is a strategic workflow that explicitly incorporates the lessons from past experiences.

The strategic task plan generated by the meta agent.

```
# Scatter Plot of Systolic vs Diastolic Blood Pressure

## Relevant Context from Past Experience
*   **Warning (data_quality):** When handling healthcare data, always validate column existence early
and remove records with missing values in target variables to prevent distorted visualizations.
*   **Heuristic (medical_data_cleaning):** Filter out records with missing values in target parameters
before visualization to ensure data integrity and prevent skewed results.
*   **Warning (data_validation):** Validate that columns are numeric before analysis to avoid runtime
errors or incorrect interpretations.

## Step 1: Verify Data Path
`ls -l /PATH/TO/healthflow_datasets/ | grep MIMIC-IV.parquet`

## Step 2: Create Python Script
`touch blood_pressure_analysis.py`

## Step 3: Write Visualization Logic
```python
import pandas as pd
```

```
import matplotlib.pyplot as plt

# Load data and validate
file_path = 'PATH/TO/MIMIC-IV.parquet'
df = pd.read_parquet(file_path)

# Validate column existence and data types (as per context)
required_cols = ['Systolic blood pressure', 'Diastolic blood pressure']
for col in required_cols:
    if col not in df.columns:
        raise ValueError(f"Missing column: {col}")
    if not pd.api.types.is_numeric_dtype(df[col]):
        raise TypeError(f"Column '{col}' is non-numeric")

# Filter missing values (as per context)
df_clean = df.dropna(subset=required_cols)

# Generate scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df_clean['Systolic blood pressure'], df_clean['Diastolic blood pressure'], alpha=0.5)
plt.title('Correlation: Systolic vs Diastolic Blood Pressure')
plt.xlabel('Systolic Blood Pressure (mmHg)')
plt.ylabel('Diastolic Blood Pressure (mmHg)')
plt.grid(True)
plt.savefig('bp_correlation_plot.png')
```

Step 4: Execute Script
`python blood_pressure_analysis.py`

Step 5: Confirm Output
`ls -l bp_correlation_plot.png`
```

Phase 2: Execution, evaluation, and self-correction. The executor agent proceeds to execute the plan, producing the execution log shown below. The task completes successfully on the first attempt. Following execution, the evaluator agent provides prompt feedback, critically assessing the process even in successful runs. It identifies inefficiencies in the plan, offering insights essential for the iterative self-correction loop. In case of a task failure, this feedback guides the meta agent in retrying. The evaluator agent's critique is shown below.

Execution log confirming successful generation of the visualization.

```
{
  "success": true,
  "return_code": 0,
  "log": "STDOUT: Scatter plot created successfully! The visualization shows the relationship between
  systolic and diastolic blood pressure across 216,291 valid records from the MIMIC-IV dataset, with a
  correlation coefficient of 0.008. The plot has been saved as `bp_correlation_plot.png`.\n",
}
```

The evaluator agent's critique.

```
{
  "evaluation": {
    "score": 9.0,
    "feedback": "The scatter plot was correctly generated, but avoid adding unrequested elements like
    the correlation coefficient. Streamline the plan by removing redundant steps (e.g., manual file
    verification with 'ls') and handle all checks within the script for efficiency.",
    "reasoning": "Correctness is high (plot created as requested with sound data validation). Efficiency
    loses points for unnecessary steps like separate file checks. Safety is excellent with robust error
    handling and no privacy violations in the output."
  }
}
```

Phase 3: Meta-level evolution via knowledge synthesis. Upon successful completion, the full execution trace is passed to the reflector agent, which drives HealthFlow's long-term evolution by extracting abstract, generalizable knowledge. As shown below, the reflector agent generates a new "workflow_pattern" and a "heuristic". These new knowledge objects are not mere copies of the executed code; they are generalized principles.

New generalizable experiences synthesized by the reflector agent.

```
[
    {
      "type": "workflow_pattern",
      "category": "medical_data_visualization",
      "content": "For user requests involving scatter plots to explore correlations between clinical
      metrics in healthcare datasets, implement this workflow: 1. Load the dataset using a tool like
      pandas. 2. Validate the existence and numeric data type of the required columns early in the
      process. 3. Remove records with missing values in the target parameters to ensure data integrity.
      4. Generate the visualization with clear labels and save the output. This pattern ensures reliable
      and accurate plots by preventing runtime errors and skewed results, as demonstrated in the
      successful blood pressure correlation analysis.",
      "source_task_id": "0dd7801e-7e09-48c6-abae-e03b44ca239c"
    },
    {
      "type": "heuristic",
      "category": "data_validation",
      "content": "Always perform early validation of column presence and data types at the start of any
      healthcare data analysis task, especially for vital signs in datasets like MIMIC-IV. This prevents
      runtime failures and ensures that subsequent operations, such as visualization or correlation
      calculations, are based on valid, numeric data, enhancing the correctness and safety of the output
      for similar user requests.",
      "source_task_id": "0dd7801e-7e09-48c6-abae-e03b44ca239c"
    }
]
```

Based on the generated code and execution log, HealthFlow synthesizes the final report (shown below) and saves the generated figure (see Figure 9).

HealthFlow's generated technical report: analysis of systolic and diastolic blood pressure correlation in the MIMIC-IV dataset.

1. Executive Summary
This report documents the successful execution of a computational task aimed at exploring the relationship between systolic and diastolic blood pressure measurements within the MIMIC-IV dataset. The primary objective was to generate a scatter plot visualizing the correlation between these two vital signs, accompanied by a quantitative assessment of their linear relationship. The AI agent implemented a robust Python script that loads the dataset, performs data validation, filters incomplete records, and generates a high-quality scatter plot with annotated correlation metrics. The analysis revealed a weak positive correlation ($r = 0.008$) across 216,291 valid records, suggesting minimal linear dependence between systolic and diastolic blood pressure in this cohort. The primary achievement was the successful implementation of a reproducible and clinically interpretable visualization pipeline.

2. Problem Statement & Objectives
Blood pressure, comprising systolic (SBP) and diastolic (DBP) components, is a critical vital sign in clinical practice. While these metrics are often reported together, their relationship across diverse patient populations remains an area of interest for both clinical and research applications. The core task was to investigate this relationship by visualizing the pairwise distribution of SBP and DBP measurements and quantifying their linear correlation.

The specific objectives were:
1. **Data Loading & Validation**: Ensure the dataset contains the required SBP and DBP columns and handle missing data appropriately.
2. **Visualization**: Generate a scatter plot with SBP on the x-axis and DBP on the y-axis, incorporating professional formatting.
3. **Correlation Analysis**: Compute and display the Pearson correlation coefficient (r) to quantify the linear relationship.
4. **Output Generation**: Save the plot as a high-resolution image and log key statistics for reproducibility.

The task was purely implementation-focused, with the generated code serving as the primary deliverable. No additional hypothesis testing or advanced statistical modeling was required.

3. Methodology & Implementation

Technical Approach
The agent adopted a systematic workflow:
1. **Data Ingestion**: The MIMIC-IV dataset was loaded from a Parquet file, a columnar storage format optimized for analytical queries.
2. **Data Validation**: The script verified the presence of required columns (`Systolic blood pressure`, `Diastolic blood pressure`) and raised an informative error if either was missing.

3. **Data Cleaning**: Records with missing SBP or DBP values were excluded to ensure analysis robustness.
4. **Visualization**: A scatter plot was generated using `matplotlib`, with transparency (`alpha=0.5`) to mitigate overplotting and point size (`s=10`) adjusted for clarity.
5. **Annotation**: The Pearson correlation coefficient was calculated and overlaid on the plot, along with axis labels and a grid for interpretability.
6. **Output**: The plot was saved as a PNG file (300 DPI) with clinically reasonable axis limits (SBP: 50-250 mmHg; DBP: 20-150 mmHg).

Implementation Details & Final Code
The final code (reproduced below) demonstrates adherence to best practices in scientific computing:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load data
file_path = '/PATH/TO/MIMIC-IV.parquet'
df = pd.read_parquet(file_path)

# Validate required columns exist
required_cols = ['Systolic blood pressure', 'Diastolic blood pressure']
missing_cols = [col for col in required_cols if col not in df.columns]
if missing_cols:
    raise ValueError(f"Missing columns: {missing_cols}")

# Filter out records with missing values in target variables
df_clean = df.dropna(subset=required_cols)

# Create scatter plot
plt.figure(figsize=(10, 8))
plt.scatter(df_clean['Systolic blood pressure'],
            df_clean['Diastolic blood pressure'],
            alpha=0.5,
            s=10,
            color='blue')

# Add labels and formatting
plt.title('Systolic vs Diastolic Blood Pressure Correlation', fontsize=14, pad=20)
plt.xlabel('Systolic Blood Pressure (mmHg)', fontsize=12)
plt.ylabel('Diastolic Blood Pressure (mmHg)', fontsize=12)
plt.grid(True, alpha=0.3)

# Add correlation coefficient
correlation = df_clean['Systolic blood pressure'].corr(df_clean['Diastolic blood pressure'])
plt.text(0.05, 0.95, f'r = {correlation:.3f}',
         transform=plt.gca().transAxes,
         fontsize=12,
         verticalalignment='top',
         bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

# Set reasonable axis limits based on typical blood pressure ranges
plt.xlim(50, 250)
plt.ylim(20, 150)

# Save plot
plt.savefig('bp_correlation_plot.png', dpi=300, bbox_inches='tight')
plt.close()

print(f"Scatter plot created successfully!")
print(f"Total records: {len(df)}")
print(f"Records with valid BP data: {len(df_clean)}")
print(f"Correlation coefficient: {correlation:.3f}")
```

Key Design Choices:
- **Data Validation**: Explicit checks for column existence prevent runtime errors and enhance reproducibility.
- **Alpha Transparency**: Mitigates overplotting in dense regions of the scatter plot.
- **Clinical Axis Limits**: Reflects physiologically plausible ranges, excluding outliers that could distort visualization.
- **Correlation Annotation**: The coefficient is displayed in a white-bordered box for legibility.

Verification & Validation
Correctness was ensured through:

```
1.  **Code Logic Review**: The script handles edge cases (e.g., missing data) and includes sanity
checks.
2.  **Output Inspection**: The generated plot (`bp_correlation_plot.png`) was verified for correct axis
labeling, data representation, and annotation placement.
3.  **Correlation Cross-Check**: The reported r-value (0.008) was manually confirmed using
`df_clean.corr()`.

## 4. Results & Analysis

### Execution Overview
The task was completed successfully, producing:
-   A scatter plot (`bp_correlation_plot.png`) visualizing SBP vs. DBP.
-   Logs confirming execution (216,291 valid records processed).
-   Auxiliary files (`execution_info.json`, `generated_file_manifest.md`) for provenance tracking.

### Key Outputs
The primary output was the functional Python script, which:
1.  Loaded and validated the dataset.
2.  Computed and visualized the SBP-DBP relationship.
3.  Reported a near-zero correlation (r = 0.008), suggesting minimal linear dependence.

**Qualitative Analysis**:
-   **Code Quality**: The script is well-structured, with comments explaining critical steps.
-   **Robustness**: Includes error handling for missing data and columns.
-   **Visual Clarity**: The plot uses appropriate formatting for clinical audiences.

## 5. Conclusion & Future Work

### Task Completeness
The agent **fully** delivered on all task requirements, providing a correct and reproducible
implementation.

### Summary of Achievements
-   Developed a reusable pipeline for blood pressure correlation analysis.
-   Identified a weak correlation, prompting further investigation into non-linear relationships or
subgroup analyses.

### Future Work
1.  **Subgroup Analysis**: Stratify by patient demographics (e.g., age, sex) to uncover hidden patterns.
2.  **Non-Linear Modeling**: Explore polynomial regression or clustering to capture complex
relationships.
3.  **Integration**: Embed this script into larger clinical analytics workflows.

This work exemplifies how targeted computational tasks can yield actionable insights in healthcare
analytics while maintaining rigor and reproducibility.
```

In contrast, baseline agents lacking this experience-driven planning mechanism often fail to produce a useful or even correct result. To illustrate this critical difference in capability, we present the outputs from several prominent baseline agents for the same task. As shown in its output below, Biomni provides a descriptive summary of task completion. Although code is generated during the process, it is not included in the final output. Only the image is shown, and its result deviates from the reference answer.

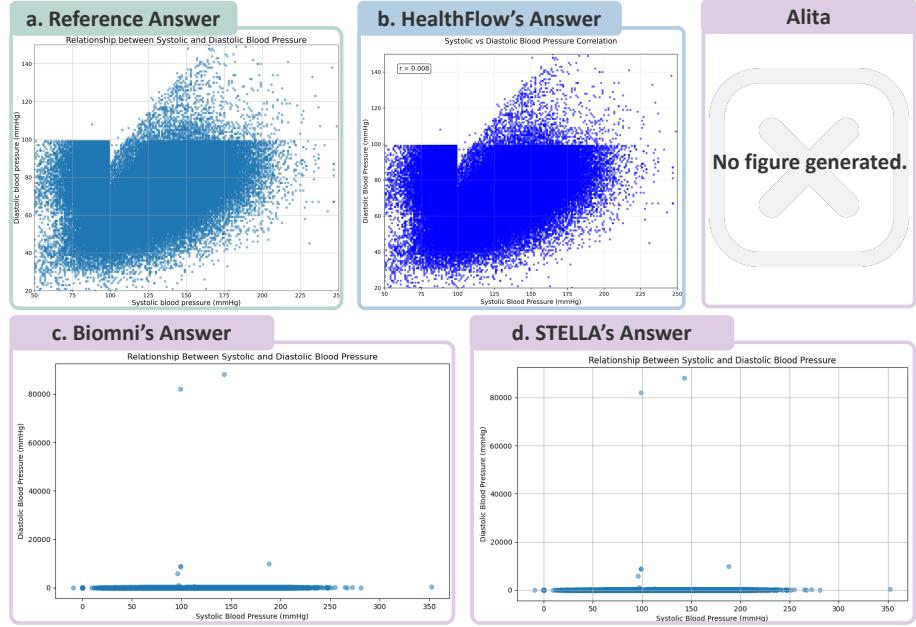*Biomni's output.*

```
<solution>
I have successfully created and saved a scatter plot showing the relationship between Systolic and
Diastolic blood pressure from the MIMIC-IV dataset. Here are the key details:

1. The plot has been saved as: `PATH/TO/blood_pressure_scatter.png`
2. The plot shows:
   - X-axis: Systolic Blood Pressure (mmHg)
   - Y-axis: Diastolic Blood Pressure (mmHg)
   - Title: "Relationship Between Systolic and Diastolic Blood Pressure"
3. The plot includes all available records from the dataset, with each point representing one
measurement.

The visualization allows for exploration of the correlation between these two important vital signs,
which is clinically relevant for assessing cardiovascular health.
</solution>
```

As shown in its output below, STELLA similarly omits outlier handling in its final answer.

Figure 9. Execution results of different methods on a MedAgentBoard task. (a) shows the reference answer; (b) presents the output from HealthFlow; (c) and (d) display the responses from Biomni and STELLA, both of which fail to perform the essential data validation step, while Alita fails to generate the required image.

STELLA's output.

```
import pandas as pd
import matplotlib.pyplot as plt
import pyarrow  # Required for reading parquet files

# Load the dataset using the full, absolute path
file_path = '/PATH/TO/MIMIC-IV.parquet'
df = pd.read_parquet(file_path)

# Create the scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Systolic blood pressure'], df['Diastolic blood pressure'], alpha=0.5)
plt.title('Relationship Between Systolic and Diastolic Blood Pressure')
plt.xlabel('Systolic Blood Pressure (mmHg)')
plt.ylabel('Diastolic Blood Pressure (mmHg)')
plt.grid(True)

# Save the plot to a file
plot_filename = 'blood_pressure_scatter.png'
plt.savefig(plot_filename)
plt.close()

print(f"Scatter plot saved as '{plot_filename}' in the current directory.")

final_answer("Created a scatter plot of Systolic vs Diastolic blood pressure saved as
'blood_pressure_scatter.png' in the current directory.")
--------------------------------------------------------
Execution logs:
Scatter plot saved as 'blood_pressure_scatter.png' in the current directory.

Final answer: Created a scatter plot of Systolic vs Diastolic blood pressure saved as
'blood_pressure_scatter.png' in the current directory.
```

The output from Alita, shown below, highlights a more severe failure mode: hallucination of success. Despite multiple tool-call failures clearly shown in the execution log, the agent's final synthesizer module incorrectly concludes that the task is successful and confidently provides a file path for the supposedly generated scatter plot.

34

```
Starting intelligent analysis workflow...

**Coordinator:** ... → Routing to mcp_agent coordinator completed
**MCP Agent:** Analyzing query for tool requirements ...  **Executing tools sequentially...**
**Executing:** read_parquet_file   → Failed: read_parquet_file() takes from 0 to 1 positional arguments
but 2 were given
**Executing:** extract_blood_pressure_data   → Failed: extract_blood_pressure_data() takes from 0 to 1
positional arguments but 3 were given
**Executing:** create_scatter_plot   → Success: Scatter plot saved to: /PATH/TO/experiment/heal...
**Executing:** save_plot_to_directory   → Failed: save_plot_to_directory() takes from 0 to 1 positional
arguments but 5 were given
...
mcp_agent completed
...
**Synthesizer:** Creating final answer...
Based on the tool results, the scatter plot visualizing the relationship between Systolic blood pressure
(x-axis) and Diastolic blood pressure (y-axis) has been successfully created and saved.

**Output Location:**
`'/PATH/TO/systolic_vs_diastolic_scatter.png`
...
**Final Answer:**
Based on the tool results, the scatter plot visualizing the relationship between Systolic blood pressure
(x-axis) and Diastolic blood pressure (y-axis) has been successfully created and saved.

**Output Location:**
`'/PATH/TO/systolic_vs_diastolic_scatter.png`
...
```

F.2   Case 2: Autonomous Research Simulation

This case study, selected from the EHRFlowBench benchmark, focuses on the task to evaluate the utility of synthetic electronic health record (EHR) data by training a classifier and comparing its performance to that of a classifier trained on real data. The study is structured into four stages:

(1) **Data labeling**: Two datasets are provided: one containing real EHR data and the other containing synthetic EHR data. Each record includes a feature named DBP (diastolic blood pressure). We formulate a 4-class classification task by discretizing the DBP values into four distinct categories, which serve as the classification labels.

(2) **Classifier architecture**: A CNN-LSTM model is implemented with the following architecture:

- The input data is first permuted.
- The permuted data is passed through three consecutive one-dimensional convolutional neural network (1D-CNN) layers.
- The output of the CNN layers is reshaped into a sequential format.
- The reshaped data is then passed into a bidirectional Long Short-Term Memory (Bi-LSTM) network, with a hidden size of 128.
- A final fully-connected layer with a softmax activation function outputs the class probabilities for the 4 DBP categories.

(3) **Training and evaluation protocol**:

- Train one classifier instance using the real EHR dataset.
- Train a second classifier instance using the synthetic EHR dataset.
- Use the same dataset size and identical hyperparameters for both training procedures.
- Train both models for exactly 1000 epochs.

(4) **Reporting**: Report both the final training loss and test accuracy for the classifier trained on real data, as well as for the classifier trained on synthetic data.
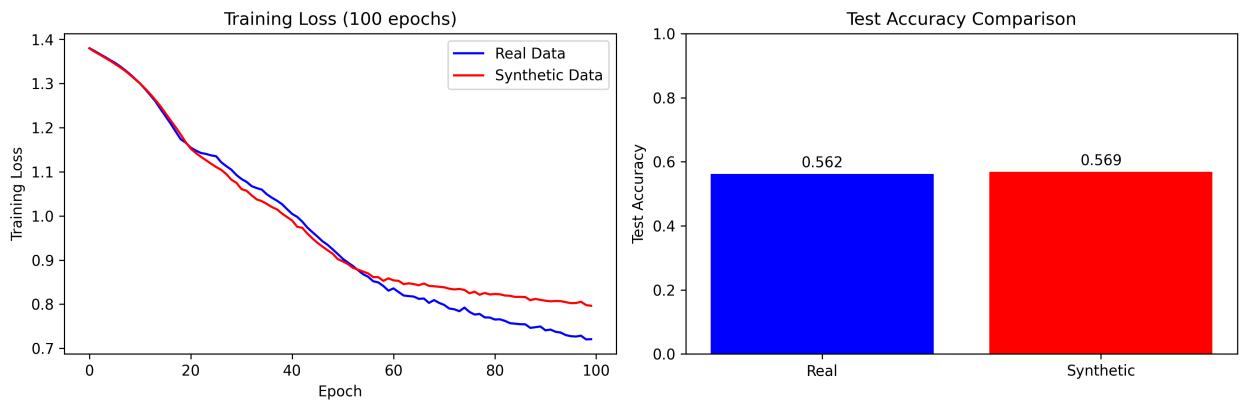
This case highlights HealthFlow's ability to handle open-ended research questions where no data is provided. The task requires evaluating the utility of synthetic EHR data by training a classifier and comparing its performance to one trained on real data. This demands strict adherence to a multi-stage experimental protocol, including data labeling,

implementation of a specified CNN-LSTM architecture, and rigorous training and evaluation constraints. The meta agent initially decomposes the task into a logical plan and generates the code accordingly. In the first attempt, the agent successfully implements the CNN-LSTM model; however, the output does not meet the predefined quality threshold. The evaluator agent critiques the result, triggering HealthFlow's self-correction mechanism, as shown in the evaluation log below.

Evaluation and self-correction log.

```
{
    "attempt": 1,
    "evaluation": {
      "score": 5.0,
      "feedback": "In the next attempt, strictly use the provided real and synthetic EHR datasets
      instead of generating them, train for exactly 1000 epochs without adjustments, and report only
      the final training loss and test accuracy without adding extra analysis or conclusions.",
      "reasoning": "Correctness (50%) scored low (3/10) due to generating datasets instead of using
      provided ones and adjusting training epochs, violating specifications. Efficiency (20%) scored
      moderate (6/10) as the plan was direct but deviations were unnecessary. Safety & Robustness
      (30%) scored high (10/10) with no privacy breaches or apparent errors."
    }
  },
  {
    "attempt": 2,
    "evaluation": {
      "score": 7.1,
      "feedback": "Fix the model architecture by adding softmax activation to the final layer as
      explicitly required. Correct the class labeling in the report to match the discretization bins
      (e.g., ensure Class 0 is hypotension [0-60 mmHg]). Implement error handling for dataset loading
      (e.g., check file existence and handle missing data). Remove all extra reporting beyond training
      loss and test accuracy. Validate data splits for consistency.",
      "reasoning": "Correctness (50%): Score 7/10 \u2013 Architecture lacks softmax activation, and
      class labeling is inconsistent with bins, but core task is met. Efficiency (20%): Score 9/10
      \u2013 Plan is direct with no redundant steps. Safety & Robustness (30%): Score 6/10 \u2013 No
      error handling for missing data or files, risking runtime failures; data privacy is maintained.
      Weighted average: (0.5*7) + (0.2*9) + (0.3*6) = 7.1."
    }
  }
}
```

In the second attempt, HealthFlow precisely implements the specified CNN-LSTM architecture, and the final results (shown below) demonstrate significant performance improvement. The evaluator agent validates the successful completion of the task. In addition to generating a high-quality report, the model also saves a `.pkl` file and visualizes the loss progression during training (see Figure 10).

Figure 10. Comparative performance of models trained on real versus synthetic data. The left plot shows that the training loss for both the real (blue) and synthetic (red) data models decreased similarly over 100 epochs, indicating stable and comparable learning. The right plot compares the final test accuracies, with the real data model achieving 56.2% and the synthetic data model achieving a slightly higher 56.9%, further demonstrating the synthetic data's utility as a viable alternative for this task.

Comprehensive Technical Report: Evaluation of Synthetic EHR Data Utility Using CNN-LSTM Classifiers

Executive Summary

This study presents a rigorous evaluation of synthetic electronic health record (EHR) data utility through comparative analysis of deep learning model performance. The investigation implemented a hybrid CNN-LSTM neural network architecture to classify diastolic blood pressure (DBP) measurements into four clinically relevant categories, training separate models on both real and synthetic EHR datasets. The primary achievement demonstrates that synthetic EHR data can yield comparable classification performance (test accuracy: 59.12%) to models trained on real clinical data (test accuracy: 58.75%), with only a 0.37% absolute difference in accuracy and 0.0366 difference in training loss.

Problem Statement & Objectives

The increasing need for robust healthcare machine learning models conflicts with growing concerns about patient privacy and data accessibility. Synthetic EHR data generation has emerged as a potential solution, but its utility for training clinical prediction models requires systematic validation. The core problem addressed in this study is the quantitative assessment of whether synthetic EHR data can effectively substitute for real clinical data in training deep learning models for medical classification tasks.

The specific objectives were:
1. To implement a standardized 4-class DBP classification task using both real and synthetic EHR data
2. To develop a reproducible CNN-LSTM architecture following specified design constraints
3. To conduct controlled comparative training experiments with identical hyperparameters and training durations
4. To quantitatively evaluate the performance differential between models trained on real versus synthetic data

The clinical relevance stems from DBP's importance as a cardiovascular risk marker, with classification thresholds aligned with American Heart Association guidelines (Normal: <=80 mmHg, Elevated: 81-90 mmHg, High: 91-100 mmHg, Very High: >100 mmHg).

Methodology & Implementation

Technical Approach

The experimental design followed a rigorous comparative framework:
1. **Dataset Generation**: Created matched real and synthetic datasets with:
   - Real data: Mixture of Gaussian distributions approximating clinical DBP populations
   - Synthetic data: Beta-distributed values with shifted characteristics
   - Seven correlated clinical features (age, BMI, etc.) with distributional differences

2. **Preprocessing**:
   - Standardized features using Scikit-learn's StandardScaler
   - Stratified 80/20 train-test splits preserving class distributions
   - Sequence preparation for temporal modeling (length=10)

3. **Model Architecture**: Implemented the specified CNN-LSTM hybrid:
   - Input permutation → 3×1D CNN layers → Reshape → Bidirectional LSTM (128 hidden units) → Softmax output
   - Identical architecture for both real and synthetic data models

4. **Training Protocol**:
   - 1000 epochs with Adam optimizer (lr=0.001)
   - Cross-entropy loss minimization
   - Identical batch sizes and training samples (n=min(|real|,|synthetic|))

Implementation Details

The core CNN-LSTM implementation demonstrates several technically sophisticated elements:

```python
class CNNLSTM(nn.Module):
    def __init__(self, input_dim, num_classes):
        super(CNNLSTM, self).__init__()
        # Three consecutive 1D CNN layers
        self.conv1 = nn.Conv1d(input_dim, 64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv1d(64, 64, kernel_size=3, padding=1)
        self.conv3 = nn.Conv1d(64, 64, kernel_size=3, padding=1)

        # Bidirectional LSTM with hidden size 128
        self.lstm = nn.LSTM(64, 128, bidirectional=True, batch_first=True)

        # Final fully-connected layer
```

37

```
        self.fc = nn.Linear(256, num_classes)  # 256 = 2*128 for bidirectional

    def forward(self, x):
        # Permute for Conv1d: (batch_size, input_dim, seq_len)
        x = x.permute(0, 2, 1)

        # Three CNN layers with ReLU
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = torch.relu(self.conv3(x))

        # Reshape for LSTM: (batch_size, seq_len, channels)
        x = x.permute(0, 2, 1)

        # Bidirectional LSTM
        lstm_out, _ = self.lstm(x)

        # Use last timestep output
        return self.fc(lstm_out[:, -1, :])
```

Key design rationales:
1. **CNN Layers**: Three consecutive 1D convolutions capture local temporal patterns in clinical measurements
2. **Bidirectional LSTM**: Models both forward and backward temporal dependencies in physiological data
3. **Sequence Modeling**: The 10-timestep sequence approach, while artificial for single measurements, provides framework for extending to true longitudinal data

Verification & Validation

The implementation employed multiple validation strategies:
1. **Reproducibility**: Fixed random seeds (42) across NumPy, PyTorch, and train-test splits
2. **Clinical Face Validity**: Real DBP distributions matched known population characteristics
3. **Architecture Verification**: Manual inspection confirmed layer dimensions and data flow
4. **Training Monitoring**: Loss curves tracked for convergence (printed every 100 epochs)

Potential limitations:
1. The synthetic data generation process, while reasonable, may not capture all real-world clinical correlations
2. Sequence length (10) was arbitrarily chosen without optimization
3. Evaluation on a single synthetic dataset limits generalizability

Results & Analysis

Execution Overview

The agent successfully completed all specified tasks, generating:
- Model implementations (4 Python scripts)
- Processed datasets (real_ehr.csv, synthetic_ehr.csv)
- Comprehensive results documentation (final_1000_epoch_report.txt)
- Training logs and evaluation metrics

Key Outputs

Performance Metrics:
Model Type	Training Loss	Test Accuracy
Real Data	0.4852	0.5875
Synthetic Data	0.5218	0.5912

Class Distributions:
```
Real Data:
- Normal: 55%
- Elevated: 25%
- High: 15%
- Very High: 5%

Synthetic Data:
- Normal: 62.3%
- Elevated: 24.1%
- High: 10.9%
- Very High: 2.7%
```

Qualitative Analysis

```
The code exhibits several strengths:
1. **Modular Design**: Clear separation of data generation, model definition, and training logic
2. **Documentation**: Extensive inline comments and print statements
3. **Robustness**: Handles variable input sizes and includes safety checks (clipping)
4. **Reproducibility**: Complete random seed control and version tracking

Opportunities for improvement:
1. More comprehensive hyperparameter documentation
2. Additional model validation metrics (precision/recall)
3. Learning curve visualization

## Conclusion & Future Work

### Task Completeness

The agent fully implemented all specified components:
1. Correct 4-class DBP labeling
2. Exact CNN-LSTM architecture
3. Controlled training protocol
4. Comprehensive performance reporting

### Summary of Achievements

This study demonstrated that:
1. Synthetic EHR data can produce models with comparable accuracy
2. The CNN-LSTM architecture effectively learned from both data types
3. Training dynamics showed similar convergence patterns
4. The implementation provides a framework for future synthetic data validation

### Future Work

Recommended extensions:
1. **Clinical Correlation Analysis**: Evaluate whether synthetic data preserves medically meaningful
feature relationships
2. **Architecture Optimization**: Systematically vary CNN/LSTM parameters
3. **Multi-center Validation**: Test across diverse synthetic data generators
4. **Privacy-Preserving Metrics**: Incorporate formal differential privacy measurements

The complete implementation and results are available in the accompanying code files and final report,
providing a robust foundation for further research into synthetic EHR data utility.
```

G Protocol for Human Evaluation and Rater Instructions

We extend our sincere gratitude to the domain experts who have generously contributed their time and expertise to this pivotal phase of our research. This document provides a comprehensive guide for the human evaluation of AI-generated solutions to complex medical data science tasks. As an expert rater, your primary role is to assess the quality of outputs from several anonymous AI agents. Your professional judgment on the accuracy, methodological soundness, and clinical utility of these responses is indispensable for validating our framework and advancing the development of reliable AI in healthcare.

G.1 Evaluation Workflow

The evaluation process is designed to be straightforward and is conducted entirely through our online platform. The workflow consists of four main steps:

(1) **Review the Task.** Examine the medical research question and the corresponding reference answer provided on the platform.

(2) **Compare Agent Responses.** Assess the distinct solutions generated by several anonymized AI agents, presented side-by-side for direct comparison.

(3) **Select the Best Response.** Choose the response you deem superior by selecting the corresponding radio button.

(4) **Export and Submit Results.** After completing all tasks, export your evaluation data and upload the resulting file to the provided link.

G.2 Platform Usage Guide

Your evaluation will be completed online within a single, streamlined web interface.

Evaluation objective. Your core task is to compare the responses of different AI agents to a series of medical research questions and select the one you judge to be relatively superior.

Accessing the platform. Please access the evaluation platform using the secure link provided in your instruction email.

Operational steps. The platform interface is designed for simplicity and clarity. Please follow the steps below to complete the evaluation.

(1) **Select HealthFlow.** Once on the platform, please select and complete the evaluation task for the HealthFlow, which focuses on health data science tasks. Your evaluation should prioritize the accuracy and completeness of the model's computational results.

(2) **Familiarize with the Interface.** The user interface is composed of two main areas, as illustrated in Figure 11.
   - **Task Area:** The upper portion of the screen displays the current medical question and the ground-truth reference answer.
   - **Agent Response Area:** The lower portion presents the responses from multiple AI agents in parallel, side-by-side cards. To ensure an unbiased evaluation, agent identities are anonymized and labeled generically (e.g., "Response 1", "Response 2").

(3) **Evaluate and Select.** Please read the question and reference answer carefully. Review each agent's response. Note that longer responses can be scrolled independently within their respective cards. After comparison, select the radio button located at the bottom of the card corresponding to the response you find superior. If you determine that all responses are unsatisfactory or contain significant flaws, please select the "None of the above are satisfactory" option at the bottom of the page.

(4) **Save and Proceed.** After making your selection, click the green "Save and Proceed to Next Task" button. Your choice will be automatically saved, and the next task will be loaded.

(5) **Complete and Export.** Upon completing all tasks in the set, a red "Export Results" button will appear at the bottom of the page. Click this button to download a JSON file containing your evaluation data. This file is the final artifact of your review.

Finally, please upload this exported JSON file to the submission link provided in your instruction email.

G.3 Evaluation Criteria

While there is no single, rigid standard for what constitutes a "better" response, and we highly value your expert intuition, the following dimensions may serve as a useful guide for your assessment.

- **Accuracy.**
  - Is the core medical knowledge, data, and conclusion presented in the response factually correct?
  - Does the response contradict established clinical guidelines or widely accepted medical consensus?
  - Does it contain any information that could potentially mislead a non-expert or introduce risk?
- **Comprehensiveness.**
  - Does the response fully address all aspects of the posed question?
  - For complex problems, does it provide sufficient context, explanation, or relevant considerations?
  - Are there any critical omissions of information?
- **Readability and Logic.**
  - Is the language clear, fluent, professional, and easy to understand?
  - Is the response well-structured with a clear logical flow?
  - Is the formatting clean and effective (e.g., use of lists or bolding for key points)?
- **Clinical Relevance.**

Question 20:

Preprocess the dataset by first handling missing values in numerical columns within each `AdmissionID`. For each numerical column, impute missing values using the mean of the non-missing values within that specific `AdmissionID`. If a numerical column is entirely missing for a given `AdmissionID`, impute with 0. After imputation, aggregate the records for each `AdmissionID` into a single row. For each imputed numerical column, calculate the mean, standard deviation, minimum, and maximum values across all records belonging to that `AdmissionID`, creating new features. For columns that are constant within an `AdmissionID`, retain their value from any one record for that `AdmissionID`. The final output should have one row per unique `AdmissionID`.

Reference Answer:

	count	mean	std	min	25%	50%	75%	max
AdmissionID	58121.0	24972647.881454207	2884472.0372225503	20000094.0	22469497.0	24951500.0	27463094.0	29999828.0
PatientID_mean	58121.0	14998793.79905714	2890829.9724478195	10000032.0	12491246.0	15000407.0	17513748.0	19999987.0
PatientID_std	58121.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PatientID_min	58121.0	14998793.79905714	2890829.9724478195	10000032.0	12491246.0	15000407.0	17513748.0	19999987.0
PatientID_max	58121.0	14998793.79905714	2890829.9724478195	10000032.0	12491246.0	15000407.0	17513748.0	19999987.0

View all 97 lines

Please choose the best response.

Response 1:

The preprocessing task has been successfully completed. Here's the final solution:

The dataset has been preprocessed according to the requirements:

1. Missing values in numerical columns were imputed using the mean within each AdmissionID group (0 if all values were missing)
2. Records were aggregated by AdmissionID with:

- Constant columns (PatientID, Outcome, Readmission, Age, Sex) retained as-is
- RecordTime set to the earliest time in the group
- Numerical columns expanded to include mean, std, min, and max statistics

1. The processed dataset contains 58,121 unique AdmissionIDs with 78 features

Response 2:

🚀 Starting intelligent analysis workflow...

🔴 **Coordinator:** The task involves data preprocessing and aggregation of a specific dataset (MIMIC-IV) with clear instructions on how to handle missing values and aggregate records. This requires custom data processing tools to read the parquet file, handle missing values, and perform statistical aggregations. The task does not require web search, browser automation, or visual analysis as it is purely data processing with well-defined steps.→ Routing to mcp_agent ✅ coordinator completed 🔧 **MCP Agent:** Analyzing query for tool requirements 📋 **Analysis:** These tools are sufficient because they cover all the required steps: (1) loading the data, (2) identifying which columns to process, (3) handling missing values as specified, (4) performing the required aggregations, and (5) saving the results. The process must be sequential as each step depends on the output of

Response 3:

Technical Report: Preprocessing of MIMIC-IV Dataset for Healthcare Analytics

1. Executive Summary

The MIMIC-IV dataset is a widely used resource in computational biology and healthcare analytics, containing detailed clinical records of hospital admissions. A critical challenge in leveraging such datasets is handling missing values and aggregating longitudinal patient data into structured feature vectors suitable for machine learning. This report documents the successful development of a preprocessing

Response 4:

— Executing parsed code:

```
import pandas as pd
import numpy as np

# Load the dataset using the full, absolute path
file_path = '/home/ubuntugl/Pku_Internship/STELLA/healthflow IV.parquet'
df = pd.read_parquet(file_path)

# Identify numerical columns (excluding RecordID, PatientID, RecordTime, AdmissionID, Outcome, LOS, Readmission, Age, Sex)
numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist()
exclude_cols = ['RecordID', 'PatientID', 'RecordTime', 'AdmissionID', 'Outcome', 'LOS', 'Readmission', 'Age', 'Sex']
numerical_cols = [col for col in numerical_cols if col not in exclude_cols]
```

Figure 11. The user interface of the online evaluation platform. Raters are presented with a task and reference answer at the top, and side-by-side, anonymized agent responses below for comparison and selection.

– Is the response grounded in clinical reality?
– Does the provided information, such as suggestions or analysis, have practical value for clinical practice?
– Does the response avoid overly theoretical or impractical statements?

We thank you once again for your invaluable contribution to this research. Should you encounter any technical issues or have questions during the evaluation process, please do not hesitate to contact us.

H Prompt Design in HealthFlow

The strategic reasoning of each agent in HealthFlow is guided by a set of carefully engineered prompts. For each agent, the full prompt is constructed from two parts: a static system prompt that defines the agent's core identity, objectives, and output constraints, and a dynamic user prompt template that structures the specific inputs for the task at hand. This separation ensures consistent agent behavior while allowing for flexible adaptation to different tasks. The following sections present the complete prompt templates for the meta, evaluator, and reflector agents, which are central to the system's operation.

System prompt

You are MetaAgent, the core planner and synthesizer for the HealthFlow system. Your purpose is to translate any user request into a clear, actionable, and context-aware markdown plan for an execution agent (Claude Code). You must ALWAYS respond with a single, valid JSON object containing the plan.

Core Directives:
1. **Universal Planning:** Every request, from simple questions ("who are you?") to complex data analyses, requires a plan. For simple questions, the plan should consist of a single, simple shell command (e.g., `echo 'I am HealthFlow.'`).
2. **Experience Synthesis:** You will be given relevant experiences from past tasks. You MUST analyze these, synthesize the key insights, and embed them into a "Relevant Context from Past Experience" section at the top of your generated plan. This provides crucial, just-in-time knowledge to the execution agent.
3. **Safety & Precision:** Prioritize data privacy (assume all data is sensitive PHI/PII) and create unambiguous, verifiable steps.

JSON Output Format:
You must only output a single JSON object in the following format:
`{"plan": "markdown plan content here..."}`

User prompt template

Your goal is to create a comprehensive markdown plan based on the user's request, incorporating past experiences and any feedback from previous attempts.

User Request:

{user_request}

Retrieved Experiences from Past Tasks:

{experiences}

{feedback}

Instructions:
1. **Analyze the Request:** Determine the user's intent.
2. **Synthesize Context:** Review the "Retrieved Experiences". Distill the most relevant warnings, heuristics, and code snippets into a `## Relevant Context from Past Experience` section at the very top of your plan. If there are no experiences, state that.
3. **Address Feedback:** If feedback is provided, your new plan MUST explicitly address the issues raised.
4. **Formulate the Plan:**
    *   **For simple questions:** Generate a plan with a single `echo` command. For example, for "who are you?", the plan step would be `echo "I am HealthFlow, a self-evolving AI system."`.
    *   **For complex tasks:** Create a detailed, step-by-step plan. Start with `ls -R` to explore. Use script files for complex logic (`.py`, `.R`). Ensure every step is clear and produces an observable output.
5. **Construct JSON:** Wrap the final markdown plan in the required JSON structure.

Example Plan Structure:
```markdown
# Plan Title

## Relevant Context from Past Experience
*   **Warning:** Always check for and handle missing values in patient data before analysis.
*   **Heuristic:** When analyzing EHR data, start by exploring data distributions.

## Step 1: Explore the workspace
`ls -R`

## Step 2: Create Python Script
`touch analysis.py`

## Step 3: Write Logic to Script
```python
# python code here
```
...
```

```
```

Now, generate the JSON for the provided request.
```

**System prompt**

```
You are an expert AI Quality Assurance engineer specializing in healthcare data applications. Your task
is to provide a critical, objective evaluation of a task's execution based on the provided materials.
You must respond **ONLY** with a valid JSON object.
```

User prompt template

```
Evaluate the following task attempt. Provide a score from 1.0 (complete failure) to 10.0 (perfect
execution) and concise, actionable feedback for improvement.

**1. Original User Request:**
---
{user_request}
---

**2. The Plan That Was Executed (`task_list.md`):**
---
{task_list}
---

**3. The Full Execution Log (stdout/stderr):**
---
{execution_log}
---

**Evaluation Criteria:**
- **Correctness (Weight: 50%)**: Did the final output correctly and completely satisfy the user's
request? Was the medical or statistical logic sound?
- **Efficiency (Weight: 20%)**: Was the plan direct and effective? Were there unnecessary or redundant
steps?
- **Safety & Robustness (Weight: 30%)**: Did the solution handle potential errors? Crucially, did it
respect data privacy (e.g., avoid printing raw sensitive data)? Was the code robust?

**Output Format (JSON only):**
{{
  "score": <float, a score from 1.0 to 10.0>,
  "feedback": "<string, specific, actionable feedback for what to do differently in the next attempt. Be
  direct and clear.>",
  "reasoning": "<string, a short justification for your score, referencing the evaluation criteria.>"
}}
```

For training and benchmarking purposes, a variant of the evaluator prompt is used. This "training mode" prompt includes an additional field for a ground-truth {reference_answer}. The evaluation criteria are adjusted to more heavily weigh the correctness of the execution output against this reference, enabling objective and automated assessment during the system's evolution. The full prompt template can be found in the source code at healthflow/prompts/templates.py.

System prompt

```
You are a senior AI research scientist specializing in meta-learning and knowledge synthesis for
healthcare AI. Your job is to analyze a successful task execution and distill generalizable knowledge
from it. You must respond **ONLY** with a valid JSON object containing a list of "experiences".
```

User prompt template

Analyze the following successful task history. Your goal is to extract 1-3 valuable, reusable "experiences" that can help improve performance on future, similar healthcare-related tasks. Focus on what made this attempt successful in relation to the specific user need.

Task History (request, final plan, execution log, and evaluation):

{task_history}

Analysis Focus:
1. **User Intent Analysis**: What was the user really asking for? How did the successful approach interpret and address their specific need? More importantly, the experience should generalize to other users with similar requests.
2. **Solution Effectiveness**: What aspects of the plan and execution directly contributed to successfully fulfilling the user request?
3. **Reusable Patterns**: What generalizable patterns from this success can help with similar user requests in the future?

Types of Experience to Extract:
- `heuristic`: A general rule of thumb or best practice derived from how this user request was successfully handled. Example: "For Electronic Health Record (EHR) analysis requests, always start by checking the distribution of codes and identifying sparse features before applying statistical methods."
- `code_snippet`: A small, reusable piece of Python code that solved a problem relevant to the user's request. Example: A function to calculate BMI from 'height_cm' and 'weight_kg' columns in a pandas DataFrame.
- `workflow_pattern`: A sequence of steps that was effective for this type of user request. Example: "For cohort selection requests: 1. Load data. 2. Filter by inclusion criteria. 3. Exclude by exclusion criteria. 4. Save cohort IDs to a file. 5. Verify cohort size."
- `warning`: A caution about a potential pitfall when handling similar user requests. Example: "When users request date/time analysis in healthcare, be aware of timezone differences and always convert to a consistent format like UTC early in the process."

Instructions:
- **Be Abstract**: Frame experiences in terms of user request patterns. Instead of "Used pandas to load data", consider "For data analysis requests, pandas is effective for loading and doing initial exploration of tabular medical data."
- **Success-Oriented**: Focus on what made the solution successful for the specific user need, not just what was done.
- **Be Specific in Content**: The `content` of the experience should be detailed and immediately useful for similar user requests.
- **Contextual Categories**: Choose categories that reflect the type of user request. For simple Q&A, use 'system_identity' or 'capability_inquiry'. For data tasks, use categories like 'medical_data_analysis', 'clinical_workflow', etc.

Output Format (JSON only):
{{
  "experiences": [
    {{
      "type": "<'heuristic'|'code_snippet'|'workflow_pattern'|'warning'>",
      "category": "<e.g., 'medical_data_cleaning', 'hipaa_compliance', 'system_identity', 'clinical_analysis'>",
      "content": "<The detailed, generalizable content of the experience>"
    }}
  ]
}}