From Scores to Steps: Diagnosing and Improving LLM Performance in Evidence-Based Medical Calculations

Benlu Wang[*] [1], **Iris Xia**[*] [1], **Yifan Zhang** [2,3], **Junda Wang** [2,4],
Feiyun Ouyang [2,3], **Shuo Han** [3], **Arman Cohan** [1], **Hong Yu**[†] [2,3,4], **Zonghai Yao**[†] [2,4]
[1]Department of Computer Science, Yale University, CT, USA
[2]Center for Healthcare Organization and Implementation Research, VA Bedford Health Care
[3]Miner School of Computer and Information Sciences, UMass Lowell, MA, USA
[4]Manning College of Information and Computer Sciences, UMass Amherst, MA, USA
benlu.wang@yale.edu, iris.xia@yale.edu, zonghaiyao@umass.edu

Abstract

Large language models (LLMs) have demonstrated promising performance on medical benchmarks; however, their ability to perform medical calculations, a crucial aspect of clinical decision-making, remains underexplored and poorly evaluated. Existing benchmarks often assess only the final answer with a wide numerical tolerance, overlooking systematic reasoning failures and potentially causing serious clinical misjudgments. In this work, we revisit medical calculation evaluation with a stronger focus on clinical trustworthiness. First, we clean and restructure the MedCalc-Bench dataset and propose a new step-by-step evaluation pipeline that independently assesses formula selection, entity extraction, and arithmetic computation. Under this granular framework, the accuracy of GPT-4o drops from 62.7% to 43.6%, revealing errors masked by prior evaluations. Second, we introduce an automatic error analysis framework that generates structured attribution for each failure mode. Human evaluation confirms its alignment with expert judgment, enabling scalable and explainable diagnostics. Finally, we propose a modular agentic pipeline, MedRaC, that combines retrieval-augmented generation and Python-based code execution. Without any fine-tuning, MedRaC improves the accuracy of different LLMs from 16.35% up to 53.19%. Our work highlights the limitations of current benchmark practices and proposes a more clinically faithful methodology. By enabling transparent and transferable reasoning evaluation, we move closer to making LLM-based systems trustworthy for real-world medical applications.

1 Introduction

Clinical calculation matters, but benchmarks miss the point. While large language models (LLMs) are increasingly used in clinical settings (Achiam et al., 2023; Goodman et al., 2023; Decker et al., 2023; Ayers et al., 2023; Thirunavukarasu et al., 2023a; Singhal et al., 2023; Sun et al., 2024; Yao and Yu, 2025) for question answering (Jin et al., 2021), medical documentation summarization (Shaib et al., 2023), and even decision support (Thirunavukarasu et al., 2023b; Tu et al., 2025), many of these applications hinge on the model's ability to perform reliable medical calculations (Goodell et al., 2025). Such tasks, like computing glomerular filtration rate or cardiovascular risk, require high numerical accuracy, correct formula use, and context-aware data extraction (Cockcroft and Gault, 1976; Initiative, 2010; Gage et al., 2001). Yet existing benchmarks for evaluating LLMs in this domain fall short of these requirements.

MedCalc-Bench (Khandekar et al., 2024) recently introduced a collection of real-world medical calculation tasks, drawn from widely used calculators that surveys show were regularly used by over 80% of healthcare professionals today (MDC, 2025). However, its current evaluation protocol only checks whether the final answer falls within a ±5% tolerance. This overlooks critical failures in intermediate steps, such as selecting the wrong formula, misreading patient attributes, or miscalculating values, creating an illusion of high performance while masking real risks. Additionally, we observed corrupted data points that hindered the analysis of model performance.

We address this problem by first cleaning errors in the original benchmark and proposing a three-part framework for more faithful evaluation and performance enhancement:

- A step-by-step evaluation pipeline that assesses each reasoning stage: formula selection, entity extraction, and numerical computation. [1]

[*]Equal contribution, alphabetical order
[†]Co-corresponding authors

[1]Our code and data are released here: https://github.com/Super-Billy/EMNLP-2025-MedRaC with Apache-2.0 license.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 10809–10833
November 4-9, 2025

- An LLM-based automatic error analysis framework that attributes mistakes to specific steps and generates structured explanations, validated against human experts.
- A training-free, agentic enhancement method that decomposes medical calculation into distinct stages and leverages retrieval-augmented grounding with executable code generation to reduce hallucinations and boost accuracy.

While many recent benchmarks report steady gains in accuracy, it remains unclear how much these improvements translate into safer or more deployable systems in high-stakes domains (Kung et al., 2023; Jin et al., 2024; Yang et al., 2025). By rethinking how we evaluate and support LLMs in medical calculation, an essential, tool-heavy aspect of real clinical practice, we offer a more transferable and trustworthy pathway from NLP progress to clinical impact.

2 Background and Related Work

Limitations of Final-Answer Medical Benchmarks Early benchmarks, such as MedQA (Jin et al., 2021), PubMedQA (Jin et al., 2019), and MedMCQA (Pal et al., 2022), primarily focus on factual recall and multiple-choice question answering. However, these benchmarks do not test models' ability to perform quantitative or step-by-step reasoning. MedIQ introduces a question-asking dataset that encourages models to seek missing information, though this often degrades performance (Li et al., 2024b). MedCalc-Bench (Khandekar et al., 2024) improves upon this by introducing real-world medical calculation tasks. It draws from 55 widely used MDCalc calculators and includes 1047 patient vignettes covering scenarios such as glomerular filtration rate (GFR) estimation and body mass index (BMI) calculation. These tasks require selecting the correct formula, extracting clinical variables, and performing numerical computations. Despite its innovation, MedCalc-Bench only evaluates the final numeric answer, allowing a ±5% margin of error. This can obscure errors such as formula misapplication, omission of key patient factors, or hallucinated arithmetic. As our reanalysis reveals, many answers marked as "correct" under the original metric contain faulty reasoning chains, thereby limiting their clinical reliability. We extend MedCalc-Bench by introducing a step-wise evaluation pipeline that inspects each reasoning component—formula, extraction, computation, and answer formatting—independently, revealing deeper reasoning failures that would otherwise go undetected.

Evaluating Intermediate Reasoning with LLM-as-Judge Step-wise evaluation has gained traction in general NLP tasks (Lightman et al., 2024; Chen et al., 2022b; Lee and Hockenmaier, 2025; Shen et al., 2025), with LLMs increasingly used as automated judges (Li et al., 2024a; Gu et al., 2024). Studies show that models like GPT-4 (Achiam et al., 2023; Liu et al., 2023; Fu et al., 2024) and critique-tuned variants (Ke et al., 2023) can approximate human judgment in summarization (Chen et al., 2022a), dialogue (Zheng et al., 2023; Zhang et al., 2024), and translation (Kocmi and Federmann, 2023). In the medical domain, LLM-as-judge has been applied to clinical conversations (Tu et al., 2025; Arora et al., 2025; Wang et al., 2023), medical documentation (Croxford et al., 2025; Chung et al., 2025; Brake and Schaaf, 2024), exam question answering & generation (Yao et al., 2024a,b), and medical reasoning (Jeong et al., 2024; Tran et al., 2024). Inspired by these works, we introduce the first step-wise LLM-as-Judge framework for clinical calculation tasks.

Retrieval-Augmented and Execution-Based Enhancements In clinical NLP, hallucinations are a key concern, particularly in high-stakes applications. Retrieval-augmented generation methods (Nori et al., 2023; Xiong et al., 2024, 2025; Wang et al., 2024) address this by grounding generation in trusted sources. Visual RAG approaches further improve reliability in imaging tasks (Chu et al., 2025). Surveys confirm that RAG systematically reduces fabrication (Zhang and Zhang, 2025; Amugongo et al., 2025). Program-aided reasoning and broader tool-use approaches have been extensively studied in recent work, highlighting the value of integrating external tools into LLM workflows (Mialon et al., 2023; Gao et al., 2023). Parallel to retrieval, execution-based techniques like Self-Consistency (Wang et al., 2022) and Self-Refine (Madaan et al., 2023) offer tools for reducing arithmetic and logical errors. These methods are often applied in math and symbolic reasoning, but have not been widely tested in clinical computations. We unify both strategies into a plug-and-play agentic pipeline tailored to medical calculations. By combining formula retrieval with Python code execution, our method corrects both hallucination-driven and computation-driven errors—without re-

quiring any model fine-tuning.

3 Methods

3.1 Step-wise Evaluation

As shown in Figure 1, medical calculations typically involve multiple sequential steps, such as retrieving relevant medical knowledge and identifying the appropriate formula. We propose a structured evaluation pipeline for medical calculation tasks that decomposes the reasoning process into four sequential, individually validated steps:

Formula selection. The candidate response must employ the *correct* medical calculation formula, as defined among the 55 calculators in MedCalc-Bench, and specify it fully, including appropriate units, boundary conditions, and any relevant constraints. We constructed a reference formula library corresponding to these 55 calculators, against which each model-proposed formula is evaluated. We use an evaluator to compare the predicted formula to its canonical counterpart in this library and assign a binary correctness score.

Value extraction. We ask the evaluator to extract every numerical and categorical variable from both the clinical vignette and the model's response. These extracted variables are then compared against the gold-standard answers provided in the dataset's JSON annotations. Using a closed-book LLM evaluator, we compute the alignment and assign a binary correctness score: full agreement is required to pass, while any mismatch, such as a missing, hallucinated, or incorrectly labeled variable, results in failure.

Mathematical calculation. The evaluator verifies whether each arithmetic step is valid, based on the extracted formula and values. UUnlike MedCalc-Bench, which allows a 5% margin of error, we adopt a stricter criterion, following the tolerance defined on the original calculators' website, MDCalc. Specifically, the allowed numerical tolerance depends on the number of decimal places in the LLM's answer, capped at two decimal places. For example, an answer of 10.65 is evaluated with a ± 0.005 tolerance, while answers with more than two decimals (e.g., 10.6512) are rounded and assessed with the same ± 0.005 threshold. A binary correctness score is then assigned.

Final Answer. We evaluate whether the model's final prediction is equivalent to the ground-truth answer in the dataset, allowing for valid unit conversions.

To ensure that the model focuses solely on evaluating the correctness of the mathematical computation, we provide only the LLM-generated answer as input, excluding the ground-truth answer or any reference formulas, to avoid potential bias or leakage that could influence judgment.

Let $\mathcal{S}_i$ denote the result of the ith step in the calculation process, each step is dependent on the previous steps: $\mathcal{S}_i = f(\mathcal{S}_{i-1}, \ldots, \mathcal{S}_1)$. Define a validation function $\mathcal{V}(\cdot) \in \{\texttt{True}, \texttt{False}\}$. We propose that a step $\mathcal{S}_i$ can only possibly be correct if and only if the immediately preceding step is correct, that is,

$$\Diamond \mathcal{V}(\mathcal{S}_i) \iff \mathcal{V}(\mathcal{S}_{i-1}).$$

Our evaluation metric ensures correctness by verifying the validity of each step in the MedCalc Bench dataset. Specifically, we evaluate the following steps sequentially: formula correctness $\mathcal{V}(\mathrm{F})$, extraction correctness $\mathcal{V}(\mathrm{E})$, calculation correctness $\mathcal{V}(\mathrm{C})$, and final answer correctness $\mathcal{V}(\mathrm{A})$. We define the correctness of the calculation task for one case $\kappa \in \{\texttt{True}, \texttt{False}\}$, as the conjunction of validity across all individual steps:

$$\kappa = \mathcal{V}(\mathrm{F}) \wedge \mathcal{V}(\mathrm{E}) \wedge \mathcal{V}(\mathrm{C}) \wedge \mathcal{V}(\mathrm{A})$$

Further, we define the *Conditional Correctness* of each step $\mathcal{S}_i$ as the probability that the step is correct given that all preceding steps are correct:

$$\mathrm{CC}_i = \mathbb{P}\left(\mathcal{V}(\mathcal{S}_i) \mid \mathcal{V}(\mathcal{S}_1) \wedge \ldots \wedge \mathcal{V}(\mathcal{S}_{i-1})\right).$$

We also define the *First Error Attribution Rate* of step $\mathcal{S}_i$ as the proportion of examples in which $\mathcal{S}_i$ is the first step to fail, i.e., all previous steps are correct but $\mathcal{S}_i$ is incorrect:

$$\mathrm{FE}_i = \mathbb{P}(\mathcal{V}(\mathcal{S}_1) \wedge \ldots \wedge \mathcal{V}(\mathcal{S}_{i-1}) \wedge \neg \mathcal{V}(\mathcal{S}_i) \mid \neg \kappa).$$

This decomposition enables fine-grained error diagnosis and quantitative comparison across models and methods. Figure 1 illustrates the whole pipeline.

3.2 LLM-aided Evaluation and Structured Error Attribution

Building upon the step-wise evaluation pipeline, we design an LLM-aided judge to assess correctness at each stage. Given an input–output pair from the LLM Test Taker and a ground-truth reference

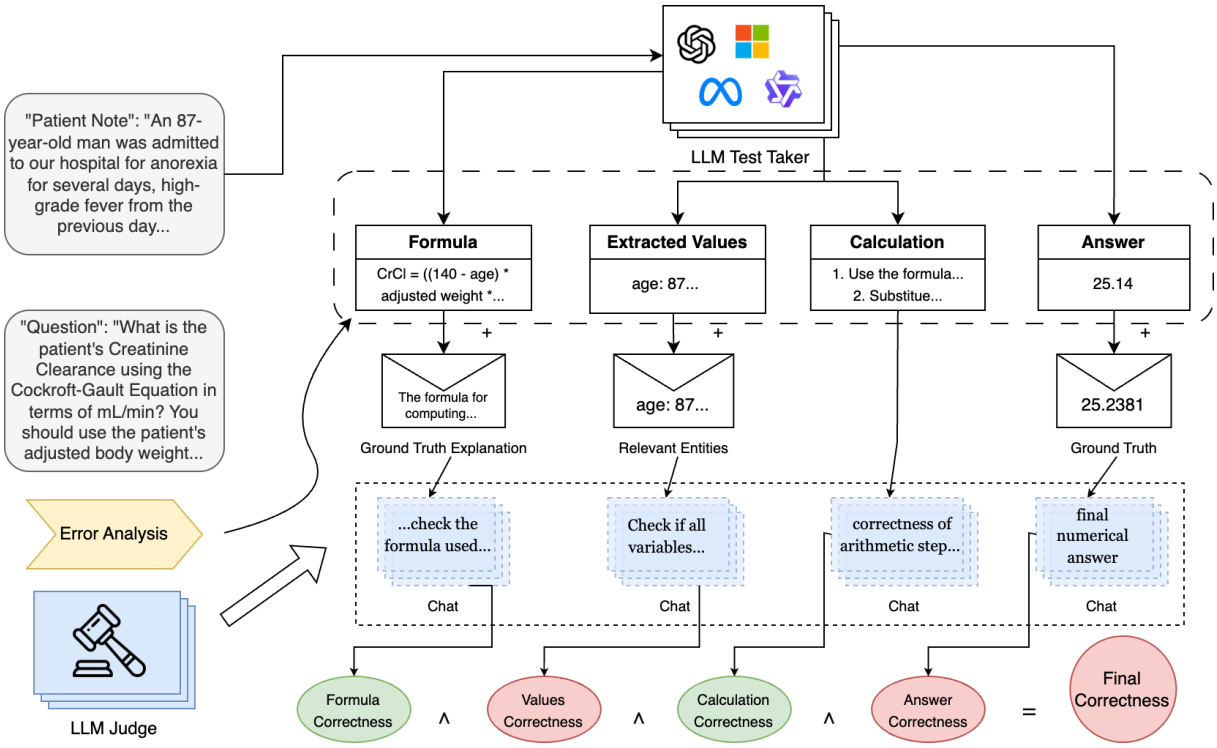

Figure 1: Step-wise LLM-aided Evaluation Pipeline. Each reasoning stage is checked by an LLM-Judge against a reference explanation to determine its correctness.

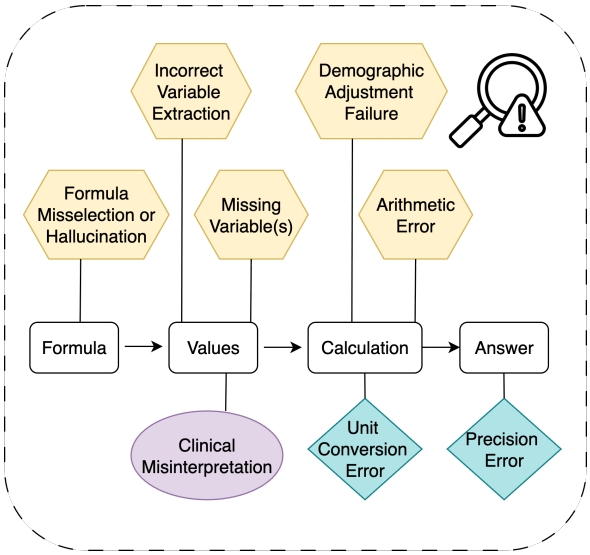

Figure 2: Categorization of Reasoning Errors in Clinical Calculation Tasks. Incorrect outputs can stem from diverse sources of failure across reasoning stages.

(e.g., extracted variable, formula used, computed value), we prompt a high-performance LLM Judge to determine semantic alignment and provide binary correctness feedback.

To further analyze the failure patterns behind incorrect answers, we define a taxonomy of common medical calculation errors, visualized in Figure 2. Each failure is assigned to one or more of the following categories:

Formula Misselection or Hallucination: The answer chooses a formula that does not fit the clinical scenario *or* distorts the correct formula by inventing, omitting, or misplacing terms, coefficients, or operators (e.g. using Cockcroft–Gault instead of CKD-EPI for an AKI patient).

Incorrect Variable Extraction: A wrong value, unit, or time-point is pulled from the note (e.g. yesterday's creatinine, or treating $\mu\text{mol}\,\text{L}^{-1}$ as $\text{mg}\,\text{dL}^{-1}$).

Clinical Misinterpretation (Rule-based): Numbers are captured correctly, but their clinical meaning is misjudged—wrong severity, threshold, or presence/absence decision (e.g. calling "trace ascites" "no ascites").

Missing Variable(s): One or more required inputs (weight, race, age group, etc.) are absent, yet the calculation proceeds, rendering the result unreliable.

Demographic Adjustment Failure: A mandatory sex, race, BSA, pregnancy, or age multiplier is skipped or applied to the wrong group (e.g. omitting the 0.85 female factor).

Unit Conversion Error (Equation-based): A value is used without the necessary unit change,

or with an incorrect factor/direction, before substitution into the formula (e.g. using 134 $\mu\text{mol}\,\text{L}^{-1}$ as 134 $\text{mg}\,\text{dL}^{-1}$).

Arithmetic Error: Pure math is wrong despite correct formula and inputs, basic addition, order of operations, exponentiation, or duplication/omission of terms.

Rounding / Precision Error (Equation-based): The final number is outside the allowed tolerance solely because of over- or under-rounding (rule of 1–2 d.p.: ±0.05 for one decimal place, ±0.005 for two).

These error types enable structured analysis of model behavior and inform targeted interventions in later modules.

3.3 MedRaC: Multi-Agent Enhancement with Formula-RAG and Code

Guided by the diagnostic insights from the step-wise evaluation and error analysis, we propose MedRaC, a modular agentic pipeline (Figure 3) to improve LLM performance on medical calculation tasks without any additional training. MedRaC combines Formula RAG, which embeds and indexes MDCalc formulas and task-specific descriptions so that relevant formulas can be retrieved and injected into the prompt before reasoning begins, thereby addressing formula selection errors and mitigating hallucination, and Python Code Execution, where the LLM is instructed to generate Python code representing the equation and this code is executed to produce the final result, eliminating arithmetic and rounding errors. MedRaC is designed to be plug-and-play, requiring no model fine-tuning and allowing it to be layered on top of existing LLM inference APIs. Each component targets a specific error type identified in our earlier analysis, enabling explainable and modular improvements.

4 Experiments

We conduct all experiments on MedCalc-Bench, a benchmark comprising 1,048 physician-curated clinical calculation cases. Because the original release contains several obsolete or internally inconsistent records, we manually reviewed the data and had a board-certified clinician re-audit every questionable item. After filtering out 108 faulty or deprecated entries, we retained 940 valid cases for evaluation. A detailed list of the removed items, along with the rationale for each exclusion, is provided in the Appendix A.

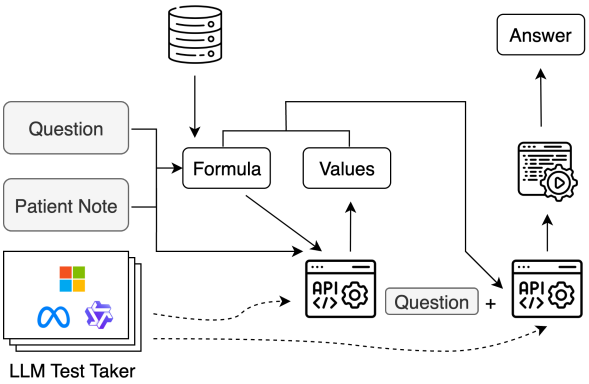

Figure 3: MedRaC Pipeline

Our primary metric is the Step-wise LLM Evaluation proposed in Section 3.1, which separately grades formula selection, entity extraction, and arithmetic computation.

Following the benchmark guidelines, we treat zero-shot Chain-of-Thought (CoT) prompting as the main baseline. In addition to the "direct" setting, where models output only the final numerical answer, we evaluate four reasoning-oriented variants. In CoT, the model produces a detailed chain of thought along with the final answer. One-shot uses the same output format but augments the prompt with a single worked example based on the same calculator as the test case. MedPrompt implements the k-nearest-neighbor retrieval component of MedPrompt with k=3 (Nori et al., 2024), without option-ordering heuristics since calculation tasks lack a multiple-choice structure. Finally, Self-Refine asks the model to critique its own response and revise the solution if an error is detected, terminating early when no error is reported, with at most five refinement rounds.

4.1 Evaluation Results

Table 1 summarizes performance across a diverse suite of closed- and open-source LLMs of varying sizes. For the direct setting, we score only the final answer, whereas all reasoning-based variants are assessed with the automatic step-wise rubric described above.

Our MedRaC method outperforms One-shot prompting across most settings. For equation-based questions, the improvement is substantial regardless of model size, confirming the benefit of external formula retrieval and modular reasoning. For *rule-based questions*, the performance gains are more nuanced. Smaller models (e.g.,

Model	Direct		CoT		One-shot		Self-Refine		Medprompt		MedRaC	
	Rule	Calc	Rule	Calc	Rule	Calc	Rule	Calc	Rule	Calc	Rule	Calc
Phi-4-mini	16.52	2.16	5.01	2.16	12.09	16.47	2.06	6.16	3.24	10.32	**35.10**	**68.39**
LLaMA3.2-3B	12.39	0.83	1.47	3.99	**14.75**	**18.47**	0.88	2.83	2.95	16.31	-[2]	-
Qwen3-4B	23.30	26.79	9.73	25.79	**49.85**	59.57	9.44	26.29	10.32	45.92	45.72	**68.72**
Qwen3-8B	29.20	42.26	16.52	38.10	**58.70**	62.90	19.76	40.10	15.93	53.74	46.61	**74.54**
LLaMA3.1-8B	19.17	2.50	6.78	8.32	25.07	20.97	5.90	6.82	11.21	28.45	**31.27**	**70.22**
Qwen3-14B	39.53	46.59	26.55	43.43	**60.77**	67.05	27.14	47.25	8.55	22.30	50.44	**78.37**
GPT-4o-mini	22.42	7.99	23.89	34.61	**52.80**	49.58	26.55	33.94	11.80	42.43	50.44	**72.71**
GPT-4o	24.48	13.98	43.07	43.93	**62.24**	54.24	44.84	44.93	25.96	56.91	51.03	**64.39**

Table 1: Performance comparison across models and prompting strategies using LLM-aided automatic evaluation. Accuracy is reported under both rule-based and calculation-based metrics.

Phi-4-mini, LLaMA series) benefit significantly from our method over One-shot, whereas stronger models (e.g., Qwen-3, GPT-3.5) show marginal improvements or even slightly worse results. We hypothesize two reasons for this pattern: (1) Larger models possess richer internal medical knowledge and are less reliant on external formulas, reducing the added value of MedRaC for rule-based cases. (2) The One-shot examples include not only scoring rules but also a worked-out example mapping patient notes to scores, which involves clinical reasoning. Stronger models are more capable of extracting and generalizing such implicit knowledge, enabling better transfer to new inputs.

4.2 Validation of LLM-aided Evaluation

We validate our evaluation pipeline from two perspectives: its ability to more effectively identify reasoning errors, and its high agreement with expert human annotations, demonstrating both improved diagnostic capability and alignment with clinical judgment.

Improved Detection of Reasoning Failures. The original MedCalc-Bench evaluates only the final numeric answer and allows a wide tolerance margin, often obscuring hallucinations or logical errors in intermediate steps. In contrast, our pipeline evaluates each stage, formula selection, variable extraction, arithmetic computation, and final answer formatting independently. This granular evaluation enables the detection of clinically significant errors that would be overlooked under final-answer-only metrics. Appendix G presents a case study where an LLM generated the correct final value but introduced multiple hallucinations during intermediate reasoning; our system successfully identified these inconsistencies.

Alignment with Expert Annotations. To evaluate the reliability of our step-wise evaluation pipeline, we compare its outputs against human annotations. We randomly sampled 46 clinical calculation questions across five calculators, spanning both rule-based and equation-based tasks. Each step in our pipeline was independently annotated for correctness by both expert and non-expert evaluators.

We assessed the alignment between our evaluation pipeline and human judgments by computing pairwise agreement scores. Specifically, we measured agreement among all human annotator pairs, as well as between our error analysis pipeline and expert annotators. Agreement is defined as simple percent agreement:

$$\text{Agreement}(a, b) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[a_i = b_i],$$

where a_i and b_i are binary correctness labels from two sources (e.g., expert and pipeline), and $\mathbb{1}[\cdot]$ is the indicator function.

Our results in Table 2 show that LLM-based error analysis achieves higher agreement with expert annotators than non-experts, and outperforms all human annotator pairs except on the extraction task. We attribute this to the LLM's careful, step-by-step consistency in evaluating responses. These findings support the validity of our evaluation pipeline in better reflecting expert clinical judgment.

Agreement Type	Formula	Extraction	Calculation	Answer
Expert–Expert	84.8%	84.8%	89.1%	95.7%
Expert–Non-Expert	72.3%	78.1%	66.6%	91.3%
LLM–Expert	90.2%	78.3%	88.1%	97.8%
All Pairs (Overall)	77.2%	81.9%	75.7%	92.5%

Table 2: Agreement scores (%) across evaluation stages.

Error Type	LLM	Non-Expert
Arithmetic	73.9%	95.7%
Clinical Misinterpretation	76.1%	87.0%
Formula	73.4%	89.1%
Variable Extraction	75.0%	76.1%
Missing Variables	90.2%	100.0%
Precision Errors	79.4%	93.5%

Table 3: Average agreement (%) between expert–LLM and expert–non-expert across error types.

4.3 Error Type Experiments

We compare the error type annotations produced by our LLM-based pipeline with those from human evaluators, as detailed in Appendix C, using the same experimental setup. Each annotator was asked to label all applicable error types in LLM-generated answers, and agreement was computed using Jaccard similarity:

$$\text{Agreement}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A and B are the sets of error types identified by two annotators.

Table 3 shows the average agreement between the LLM Judge and experts, as well as between experts and non-experts. While LLM–expert agreement is not consistently higher than human–human agreement, we observe that the LLM is reasonably aligned with expert decisions, particularly on well-defined tasks such as variable extraction and missing inputs.

These results reflect the inherent difficulty of multi-label error attribution: humans often converge on the most salient error, while LLMs evaluate each category independently and systematically. Although imperfect, the LLM's error analysis is structured, reproducible, and offers a valuable reference point for reviewing model failures.

Error-type comparison. We evaluated differences in output error types between Zero-Shot and MedRaC across four models, as detailed in Appendix E. Figure 4, using the Llama3.1-8B-Instruct model as an example, illustrates that the proposed MedRaC pipeline substantially reduces the main error types relative to the Zero-Shot CoT baseline. Almost all error categories showed decreases; among them, the steepest drops were in Formula Misselection/Hallucination (-587, -77.5%), Arithmetic Error (-352, -82.6%), and Demographic Adjustment Failure (-105, -70.9%). These decreases can be understood to stem from using grounded formulas and precise programmatic calculation. A slight increase in the low-frequency Rounding/Precision Error category likely reflects our stricter evaluation tolerance rather than a true decline in numerical performance. We also provide the error analysis of other methods evaluated with LLaMA3.1-8B in Table 4. Oneshot reduces many errors because curated examples guide step-by-step reasoning, though it cannot fix arithmetic mistakes since examples do not improve raw computation. Self-Refine performs better in math-heavy categories by iteratively correcting outputs, directly addressing numerical slips. In contrast, MedPrompt often underperforms Oneshot because noisy retrieved examples dilute key signals.

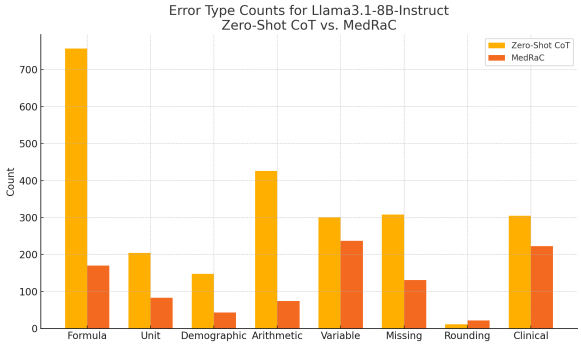

Figure 4: Error Type Counts for Llama3.1-8B-Instruct

Attribution of gains. Formula retrieval provides the model with the exact equation and relevant demographic terms, which reduces hallucinations and incorrect formulations, thereby lowering formula-related and adjustment errors. Code execution delegates arithmetic operations to Python, preventing mistakes such as incorrect operation order or unit miscalculations and yielding roughly an 83% reduction in arithmetic errors along with fewer unit conversion issues. An ablation study demonstrating the individual contributions of these two techniques will be presented in Section F.

Residual challenges. Error types that depend on nuanced clinical understanding, such as Incorrect Variable Extraction (-64, -21.3%) and Clinical Misinterpretation (-82, -26.9%), show relatively limited improvement. These cases often require background medical knowledge or familiarity with clinical reasoning, which current LLMs lack. Even when a correct formula is available, models may

[2]The model fails to output executable code

Method	Formula Error	Missing Variables	Missing/ Misused Demographic Coeff.	Unit Conversion Error	Arithmetic Errors	Rounding/ Precision	Incorrect Variable Extraction	Clinical Mis-interpretation
CoT	757	308	148	204	426	11	301	305
Oneshot	295	152	44	82	455	5	194	241
Self-Refine	818	716	29	46	101	5	370	118
Medprompt	477	271	60	97	430	54	307	295
MedRaC	170	131	43	83	74	22	237	223
code-only	776	430	93	181	91	17	416	318
rag-only	211	142	44	107	318	35	213	238

Table 4: Error counts by method and error type.

struggle if the necessary medical context is implicit or not explicitly encoded. This indicates that upstream information extraction remains a bottleneck.

5 Ablation Studies

To analyze the contribution of each MedRaC component, we conduct controlled ablations.

Formula RAG We compare MedRaC with and without retrieval, keeping the rest of the pipeline fixed. In the no-retrieval variant, the LLM is prompted to infer relevant background information before extracting the value. As shown in Table 5, accuracy drops sharply from 64.68% to 25.64% without retrieval. The formula stage becomes the primary failure point, with its First Error Attribution Rate (FE) rising to 71.96% and its Conditional Correctness (CC) falling to 7.34%. These results suggest that retrieval is crucial for selecting accurate formulas and for subsequent reasoning.

Components	MedRaC	MedRaC w/o RAG
Acc % ↑	64.68	25.64
Formula FE % ↓	20.78	71.96
Formula CC % ↑	92.66	46.49

Table 5: Comparing accuracies w/ and w/o RAG

Code We compare variants with and without code execution. In place of code generation, the model is asked to produce chain-of-thought reasoning to calculate the requested value. Since DeepSeek-v3 struggles to accurately assess Python code, we rely on reasoning models as judges; results for DeepSeek-v3 are shown here, with the rest reported in Appendix F. Across different judge models, the inclusion of the Code component consistently reduces error rates.

Memory Scaling We expand the formula bank from 55 to 785 formulas and evaluate retrieval performance using OpenAI's embedding models. The smaller set corresponds exactly to the 55 calculators in MedCalc-Bench, while the larger set includes nearly all formulas from the MDCalc website's evidence sections.[3]

Components	MedRaC	MedRaC w/o Code
Acc % ↑	64.68	53.09
Calc FE % ↓	3.23	31.88
Calc CC % ↑	97.82	76.52

Table 6: Comparing accuracies w/ and w/o Code

Retrieval is considered successful if any of the top-k retrieved formulas match the ground-truth formula for a given question. As shown in Table 7:

- All three embedding models achieve 100% top-2 accuracy in both the 55- and 785-formula settings.
- Even with a 14× increase in formula count, top-1 accuracy remains high. For instance, text-embedding-ada-002 maintains 100%, while text-embedding-3-large and text-embedding-3-small still achieve over 96%.

These results suggest that retrieval-augmented methods are especially well-suited for medical calculation tasks, not because of any inherent superiority of RAG itself, but due to the unique nature of medical formulas. These formulas are highly structured, semantically distinct, and domain-specific, which allows embedding-based retrieval to remain robust even as the size of the knowledge base increases. This makes our method practically scalable to a much broader range of clinical calculators beyond those in MedCalc-Bench.

[3]Formulas were obtained via web scraping. Ensure proper licensing and data usage compliance.

6 Discussion and Conclusion

Medical calculations are not just a numeric task. They represent structured, high-stakes reasoning in clinical workflows. Our study reveals that existing evaluation metrics, which focus solely on final answer accuracy, often fail to capture critical reasoning failures such as formula misuse, variable misinterpretation, or arithmetic errors. These oversights may result in overly optimistic assessments of model safety and applicability.

We introduce a stepwise evaluation framework and a structured error taxonomy that enable more transparent, diagnostic, and actionable feedback on model behavior. Furthermore, our MedRaC pipeline improves performance without additional training by augmenting model reasoning with explicit retrieval and executable code. Through controlled ablations and human-aligned validation, we show that each component directly mitigates failure modes in clinical computation.

Importantly, our findings point to a broader methodological shift: as language models are deployed in safety-critical domains, evaluating intermediate reasoning and domain-grounded correctness becomes essential. This work advocates for domain-aware, explanation-oriented evaluation practices that bridge the gap between model development and real-world deployment. By prioritizing interpretability and modular error analysis over end-task scores, we take a step toward safer, more trustworthy AI systems that serve beyond NLP's traditional boundaries.

7 Limitations

While our step-wise evaluation framework and MedRaC pipeline provide more granular insight into LLM reasoning in medical calculations, several limitations remain. First, our benchmark currently focuses on structured, single-turn tasks involving well-defined formulas. This setup may not accurately capture the ambiguity, context switching, or exception handling that are common in real-world clinical reasoning.

Second, although our dataset covers 55 diverse calculators and our RAG component scales to hundreds more, all experiments were conducted in English, using curated clinical notes. The generalizability of our results to multilingual settings, noisy EHR data, or patient-facing dialogue remains to be studied.

Third, the correctness judgments at each reasoning step rely on LLM-as-Judge evaluation. While we validated this against expert annotations, LLM-based evaluation may still be prone to error propagation, especially for subtle clinical misinterpretations.

Finally, while MedRaC enhances factual reliability through modular design, it assumes access to accurate formula banks and structured variables. Future work should explore more open-ended clinical reasoning and integrate real-time human oversight in deployment settings.

8 Ethics Statement

This study uses only publicly available and anonymized clinical data, including physician-written vignettes and structured case reports from sources such as PubMed Central. No identifiable patient data were accessed or used.

The MedCalc-Bench dataset and the associated MedRaC framework are designed solely for evaluating and improving LLM capabilities in medical calculations under controlled conditions. They are not intended for diagnostic use or direct clinical deployment. All outputs must be reviewed and interpreted by licensed healthcare professionals.

To validate our evaluation pipeline, we engaged two medical experts in the United States to annotate reasoning steps and provide structured feedback on LLM outputs. They were compensated for their time at a rate of $40 per hour, following academic ethical standards.

Finally, given the high-stakes nature of clinical MedCalc-Bench disqualifies a model for any real-world medical calculation task, while passing should be considered a necessary but not sufficient condition for use. We caution against deploying LLMs for clinical decision-making without rigorous benchmarking, error attribution, and domain-specific oversight. Our work is intended to contribute to the responsible development of AI for healthcare, rather than replacing expert judgment.

Acknowledgments

This material is the result of work supported with resources and the use of facilities at the Center for Healthcare Organization and Implementation Research, VA Bedford Health Care.

References

2025. About us - mdcalc. https://www.mdcalc.com/about-us. Accessed: 2025-09-19.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Lameck Mbangula Amugongo, Pietro Mascheroni, Steven Brooks, Stefan Doering, and Jan Seidel. 2025. Retrieval augmented generation for large language models in healthcare: A systematic review. *PLOS Digital Health*, 4(6):e0000877.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. 2025. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*.

John W Ayers, Adam Poliak, Mark Dredze, Eric C Leas, Zechariah Zhu, Jessica B Kelley, Dennis J Faix, Aaron M Goodman, Christopher A Longhurst, Michael Hogarth, et al. 2023. Comparing physician and artificial intelligence chatbot responses to patient questions posted to a public social media forum. *JAMA internal medicine*, 183(6):589–596.

Nathan Brake and Thomas Schaaf. 2024. Comparing two model designs for clinical note generation; is an llm a useful evaluator of consistency? *arXiv preprint arXiv:2404.06503*.

Hong Chen, Duc Vo, Hiroya Takamura, Yusuke Miyao, and Hideki Nakayama. 2022a. Storyer: Automatic story evaluation via ranking, rating and reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1739–1753. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yun-Wei Chu, Kai Zhang, Christopher Malon, and Martin Renqiang Min. 2025. Reducing hallucinations of medical multimodal large language models with visual retrieval-augmented generation.

Philip Chung, Akshay Swaminathan, Alex J Goodell, Yeasul Kim, S Momsen Reincke, Lichy Han, Ben Deverett, Mohammad Amin Sadeghi, Abdel-Badih Ariss, Marc Ghanem, et al. 2025. Verifact: Verifying facts in llm-generated clinical text with electronic health records. *arXiv preprint arXiv:2501.16672*.

Donald W Cockcroft and Henry Gault. 1976. Prediction of creatinine clearance from serum creatinine. *Nephron*, 16(1):31–41.

Emma Croxford, Yanjun Gao, Elliot First, Nicholas Pellegrino, Miranda Schnier, John Caskey, Madeline Oguss, Graham Wills, Guanhua Chen, Dmitriy Dligach, et al. 2025. Automating evaluation of ai text generation in healthcare with a large language model (llm)-as-a-judge. *medRxiv*, pages 2025–04.

Hannah Decker, Karen Trang, Joel Ramirez, Alexis Colley, Logan Pierce, Melissa Coleman, Tasce Bongiovanni, Genevieve B Melton, and Elizabeth Wick. 2023. Large language model- based chatbot vs surgeon-generated informed consent documentation for common procedures. *JAMA network open*, 6(10):e2336997–e2336997.

Jinlan Fu, See Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2024. Gptscore: Evaluate as you desire. pages 6556–6576.

Brian F Gage, Amy D Waterman, William Shannon, Michael Boechler, Michael W Rich, and Martha J Radford. 2001. Validation of clinical classification schemes for predicting stroke: results from the national registry of atrial fibrillation. *Jama*, 285(22):2864–2870.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Alex J Goodell, Simon N Chu, Dara Rouholiman, and Larry F Chu. 2025. Large language model agents can use tools to perform clinical calculations. *npj Digital Medicine*, 8(1):163.

Rachel S Goodman, J Randall Patrinely, Cosby A Stone Jr, Eli Zimmerman, Rebecca R Donald, Sam S Chang, Sean T Berkowitz, Avni P Finn, Eiman Jahangir, Elizabeth A Scoville, et al. 2023. Accuracy and reliability of chatbot responses to physician questions. *JAMA network open*, 6(10):e2336483.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *The Innovation*.

Collaborative Initiative. 2010. 2010 rheumatoid arthritis classification criteria. *Arthritis & Rheumatism*, 62(9):2569–2581.

Minbyul Jeong, Jiwoong Sohn, Mujeen Sung, and Jaewoo Kang. 2024. Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. *Bioinformatics*, 40(Supplement_1):i119–i129.

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

Qiao Jin, Fangyuan Chen, Yiliang Zhou, Ziyang Xu, Justin M Cheung, Robert Chen, Ronald M Summers, Justin F Rousseau, Peiyun Ni, Marc J Landsman, et al. 2024. Hidden flaws behind expert-level accuracy of multimodal gpt-4 vision in medicine. *NPJ Digital Medicine*, 7(1):190.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *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.

Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, et al. 2023. Critiquellm: Scaling llm-as-critic for effective and explainable evaluation of large language model generation. corr, abs/2311.18702. detection for generative large language models. pages 9004–9017.

Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad Safranek, Abid Anwar, Andrew Zhang, et al. 2024. Medcalc-bench: Evaluating large language models for medical calculations. *Advances in Neural Information Processing Systems*, 37:84730–84745.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality. *arXiv preprint arXiv:2302.14520*.

Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. 2023. Performance of chatgpt on usmle: potential for ai-assisted medical education using large language models. *PLoS digital health*, 2(2):e0000198.

Jinu Lee and Julia Hockenmaier. 2025. Evaluating step-by-step reasoning traces: A survey. *arXiv preprint arXiv:2502.12289*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan Ilgen, Emma Pierson, Pang Wei W Koh, and Yulia Tsvetkov. 2024b. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37:28858–28888.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*. Survey Certification.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, et al. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*.

Harsha Nori, Naoto Usuyama, Nicholas King, Scott Mayer McKinney, Xavier Fernandes, Sheng Zhang, and Eric Horvitz. 2024. From medprompt to o1: Exploration of run-time strategies for medical challenge problems and beyond. *arXiv preprint arXiv:2411.03590*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Chantal Shaib, Millicent Li, Sebastian Joseph, Iain Marshall, Junyi Jessy Li, and Byron C Wallace. 2023. Summarizing, simplifying, and synthesizing medical evidence using gpt-3 (with varying success). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1387–1407.

Chengyu Shen, Zhen Hao Wong, Runming He, Hao Liang, Meiyi Qiang, Zimo Meng, Zhengyang Zhao, Bohan Zeng, Zhengzhou Zhu, Bin Cui, et al. 2025. Let's verify math questions step by step. *arXiv preprint arXiv:2505.13903*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Mengxuan Sun, Ehud Reiter, Anne E Kiltie, George Ramsay, Lisa Duncan, Peter Murchie, and Rosalind Adam. 2024. Effectiveness of chatgpt in explaining complex medical reports to patients. *arXiv preprint arXiv:2406.15963*.

Arun James Thirunavukarasu, Refaat Hassan, Shathar Mahmood, Rohan Sanghera, Kara Barzangi, Mohanned El Mukashfi, and Sachin Shah. 2023a. Trialling a large language model (chatgpt) in general practice with the applied knowledge test: observational study demonstrating opportunities and limitations in primary care. *JMIR Medical Education*, 9(1):e46599.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023b. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Hieu Tran, Zonghai Yao, Junda Wang, Yifan Zhang, Zhichao Yang, and Hong Yu. 2024. Rare: Retrieval-augmented reasoning enhancement for large language models. *arXiv preprint arXiv:2412.02830*.

Tao Tu, Mike Schaekermann, Anil Palepu, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Yong Cheng, et al. 2025. Towards conversational diagnostic artificial intelligence. *Nature*, pages 1–9.

Junda Wang, Zhichao Yang, Zonghai Yao, and Hong Yu. 2024. Jmlr: Joint medical llm and retrieval training for enhancing reasoning and professional question answering capability. *arXiv preprint arXiv:2402.17887*.

Junda Wang, Zonghai Yao, Zhichao Yang, Huixue Zhou, Rumeng Li, Xun Wang, Yucheng Xu, and Hong Yu. 2023. Notechat: a dataset of synthetic doctor-patient conversations conditioned on clinical notes. *arXiv preprint arXiv:2310.15959*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6233–6251.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. 2025. Improving retrieval-augmented generation in medicine with iterative follow-up questions. In *Pacific Symposium on Biocomputing (PSB) 2025*, pages 199–214. World Scientific.

Zhichao Yang, Zonghai Yao, Mahbuba Tasmin, Parth Vashisht, Won Seok Jang, Feiyun Ouyang, Beining Wang, David McManus, Dan Berlowitz, and Hong Yu. 2025. Unveiling gpt-4v's hidden challenges behind high accuracy on usmle questions: Observational study. *Journal of Medical Internet Research*, 27:e65146.

Zonghai Yao, Aditya Parashar, Huixue Zhou, Won Seok Jang, Feiyun Ouyang, Zhichao Yang, and Hong Yu. 2024a. Mcqg-srefine: Multiple choice question generation and evaluation with iterative self-critique, correction, and comparison feedback. *arXiv preprint arXiv:2410.13191*.

Zonghai Yao and Hong Yu. 2025. A survey on llm-based multi-agent ai hospital.

Zonghai Yao, Zihao Zhang, Chaolong Tang, Xingyu Bian, Youxia Zhao, Zhichao Yang, Junda Wang, Huixue Zhou, Won Seok Jang, Feiyun Ouyang, et al. 2024b. Medqa-cs: Benchmarking large language models clinical skills using an ai-sce framework. *arXiv preprint arXiv:2410.01553*.

Chen Zhang, Luis Fernando D'Haro, Yiming Chen, Malu Zhang, and Haizhou Li. 2024. A comprehensive analysis of the effectiveness of large language models as automatic dialogue evaluators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19515–19524.

Wan Zhang and Jing Zhang. 2025. Hallucination mitigation for retrieval-augmented large language models: A review. *Mathematics*, 13(5):856.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Removed Data List and Reasons

From the original 1 048 examples, we excluded 108 records (10.3%) after a double-blinded review by two expert clinicians. Retaining these flawed items would have distorted both step-wise and end-to-end accuracy, so they were discarded before any model-level analysis.

Calculator ID 13 — *Estimated Due Date* (Equation-Based).
Rows removed: all.
The gestational-age equation in the benchmark spreadsheet was mis-typed, yielding uniformly incorrect targets; therefore, every instance was removed.

Calculator ID 28 — *APACHE II Score* (Rule-Based).
Rows removed: all.
The ground-truth rule assigned +4 points when the alveolar–arterial gradient exceeded 349 mmHg, but authoritative guidelines award +3 points from 350–499 mmHg and reserve +4 points for ≥ 500 mmHg.

Calculator ID 3 — *FeverPAIN Score for Strep Pharyngitis* (Rule-Based).
Rows removed: 451 only.
Wrong ground truth answer.

Calculator ID N/A
Rows removed (45 total):
`472, 803, 473, 946, 940, 804, 764, 936, 761, 798, 930, 738, 934, 938, 789, 469, 792, 948, 944, 937, 781, 941, 507, 478, 801, 945, 931, 477, 806, 929, 763, 794, 471, 932, 481, 942, 947, 943, 805, 486, 939, 768, 810, 933, 935, 468`
For these cases with negative answers, the data incorrectly specifies the Lower Limit and Upper Limit in reversed order—for example, Lower Limit = –4 and Upper Limit = –5—due to improper handling of negative values. As a result, the original benchmark evaluation method always yields incorrect results.

Calculator ID 11 — *QTc Bazett Calculator* (Equation-Based).
Rows removed: all.
Prompt instructions required answers in *seconds*, whereas the ground-truth column stored QTc in *milliseconds*, causing a systematic unit mismatch. Standard QT/QTc formulae—including Bazett—are defined in seconds.

Calculator ID 36 — *Caprini Score (2005)* (Rule-Based).
Rows removed: all.
The benchmark granted +1 point to every female patient. The validated rule adds this point only for pregnancy or other specific obstetric conditions; indiscriminate gender scoring overestimates risk.

B Inference Environment

All runs are inference-only—no training is performed. We use two NVIDIA RTX A6000 GPUs for local execution of open-source models. For GPT models, we use the default settings. For all other open-source models, we set the temperature to 0.6, top-p to 0.95, and repetition penalty to 1.0. The LLM Evaluation pipeline is conducted using DeepSeek-chat as the evaluator, and Error Types are checked by DeepSeek-reasoner.

Model	#F	Top-1 (%)	Top-2 (%)
ada-002	55	100.00	100.00
	785	100.00	100.00
3-large	55	98.18	100.00
	785	96.36	100.00
3-small	55	94.55	100.00
	785	98.18	100.00

Table 7: Top-k retrieval accuracy on formula sets of size 55 and 785 using OpenAI embedding models. "#F" denotes number of formulas.

C Human Annotation Details

Our human annotation study was conducted in a single session with four annotators: two medically trained experts and two students from medically related fields. Annotators were provided with the LLM-generated outputs and asked to assess their correctness. For outputs judged incorrect, they were further asked to identify the corresponding error types.

We initially selected 35 samples from code-based generations and 15 from non-code generations. During review, two samples were found to have incorrect ground truth labels. These were replaced with two additional randomly selected samples from the same calculator. However, further inspection revealed additional labeling issues. After discarding all problematic entries, the final dataset contained 46 validated samples.

C.1 Human Annotation Guidance

Human Evaluation Guidelines Evaluation Steps Thank you very much for assisting us with data annotation. The data to be evaluated is provided in a Google Sheet, where each row corresponds to a response generated by a large language model (LLM) for a medical calculation task. These responses fall into two broad categories: - Equation-based Calculation Tasks: Questions that require explicit mathematical formulae (e.g., computing BMI). - Rule-based Calculation Tasks: Questions that involve assigning points based on clinical criteria and summing them (e.g., Wells score). Step 1: Answer Validation Please read the Patient Note, and carefully read the Question and the Ground Truth Explanation. Then assess the LLM Answer.

The answer consists of four steps: formula, extracted values, calculation, and final answer. Formula and extracted values are complemented by their corresponding reasons. Please review each field carefully and verify that the steps and results are accurate. If it's not fully correct, please refer to the Possible Error Types section on the next page and select all errors that are present. Note: ONLY if ALL intermediate steps and the final result are correct is the LLM Answer considered correct. Note2: Formula correctness is assessed on both the formula field and the actual formula used during calculation. Step 2: LLM Self-Evaluation Assessment Next, review the LLM Evaluation columns. These involves another model's assessment of previous LLM's performance in terms of: - Formula correctness - Extracted values correctness - Mathematical calculation correctness

For each column, please judge the evaluation result as one of the following: Correct / Correct but explanation flawed / Incorrect We greatly appreciate your help and detailed annotations! Possible Error Types Incorrect Formula Selection The wrong medical formula is used for the given clinical scenario. Example: Using the Cockcroft-Gault equation to estimate GFR in an AKI patient instead of CKD-EPI. Internal Formula Logic or Parameter Errors (Hallucination) - Equation-based Questions: The overall formula structure is correct, but components such as terms, coefficients, or exponents are incorrect or hallucinated. Example: Writing Framingham QTc as QT + 154*1 – RR, where missing parentheses result in a miscalculation.

- Rule-based Questions: Scoring items are missing or fabricated. Example: Omitting "recent surgery" from the Wells score or adding a non-existent "family history" item. Incorrect Variable Extraction Values extracted from the patient note are incorrect in terms of number, timing, or unit. Example: Extracting "heart rate = 76 bpm" as 176 bpm, or using a lab value from a previous visit instead of the current one. Clinical Misinterpretation (Rule-based Only) Misunderstanding the clinical implications of a symptom or finding can lead to incorrect scoring. Example: "Abdomen was diffusely distended" suggests mild ascites (+2), but the model assumes no ascites and assigns +1. Missing Variables The model fails to extract the required inputs, making it impossible to complete the calculation. Example: Missing weight or race information causes incomplete or halted computation.

Unit Conversion Errors Units are not converted correctly before calculation, resulting in serious numerical errors. Example: Using 134 μmol/L creatinine in the MDRD formula without converting to mg/dL. Missing or Misused Demographic/Adjustment Coefficients Important adjustment factors, such as gender, race, BMI-based weight corrections, or pregnancy status, are omitted or misused. Example: Not applying a 0.85 coefficient for female patients in the Cockcroft-Gault equation. Arithmetic Errors - Equation-based Questions: Incorrect mathematical operations, such as order-of-operations errors or basic calculation mistakes. Example: Writing (A + B) * C as A + B * C, or calculating 3×3 as 10.

- Rule-based Questions: Correct scoring items are identified, but summed incorrectly. Example: Adding 1 + 2 + 1 and mistakenly writing 5. Rounding / Precision Errors Rounding is too aggressive or insufficient, leading to clinically significant inaccuracies. Use the number of decimal places in the LLM's answer to determine the required precision, up to a maximum of 2 decimal places. If the LLM returns 10.65, evaluate it to 2 decimal places with a tolerance of ±0.005. If it returns 10.7, use 1 decimal place with a tolerance of ±0.05. If it returns 10.6512, use two decimal places with a tolerance of ±0.05. NOTE: The final answer has already been pre-checked against the ground truth answer in the LLM Answer Eval column. You do not need to manually re-check it based on this precision rule. This error type should be marked when rounding errors or insufficient precision in intermediate or final steps cause the final answer to fall outside the tolerance range/result in "Incorrect" in the Answer Evaluation Column.

D Categorized Evaluation Results

We present categorized evaluation results in Table 8. MedRaC delivers substantial improvements in specialties that depend heavily on numerical calculations—such as Nephrology (39.7% → 90.4%), Thrombosis/Hematology (28.8% → 76.3%), Clinical Pharmacology (5.0% → 65.0%), and Endocrinology & Metabolism (65.6% → 90.2%). These gains stem from two design features: grounding formula selection in trusted medical knowledge to reduce hallucinations, and executing code to eliminate arithmetic errors. Although MedRaC does not outperform all baselines in every domain (e.g., Oneshot attains higher scores in Pulmonology and Hepatology), it achieves the most consistent and large-scale improvements in areas where computational fidelity is paramount.

By contrast, smaller gains are observed in domains such as General Practice/Family Medicine (10.0% → 40.0%) and Hepatology/Gastroenterology (16.9% → 66.2%), where success depends more on clinical judgment and contextual interpretation than on direct computation. A full comparison across models of varying parameter scales is included in the appendix.

Model scale further differentiates performance across domain types. In narrative-heavy, guideline-driven specialties (e.g., Hepatology/Gastroenterology, Infectious Disease, General Practice/Family Medicine), larger models within the same family (e.g., Qwen, LLaMA) exhibit stronger clinical recall and decision-making, reflecting the benefits of broader contextual reasoning. Conversely, in deterministic, calculation-intensive domains (e.g., Endocrinology & Metabolism, Obstetrics & Gynecology), even smaller models paired with code execution approach the performance ceiling. Beyond this point, increasing model size yields diminishing returns and may occasionally introduce over-generation or minor regressions.

E Error Type Results of other Models

Including Llama3.1-8B-Instruct, we evaluated a total of four models under both Zero-shot CoT and MedRaC: two reasoning models (GPT-4o-mini, Qwen3-4B) and two general-purpose models (Llama3.1-8B-Instruct, Qwen3-8B). The results are consistent with the trend observed for Llama3.1-8B-Instruct. In addition, we find that reasoning models show greater improvements in value extraction.

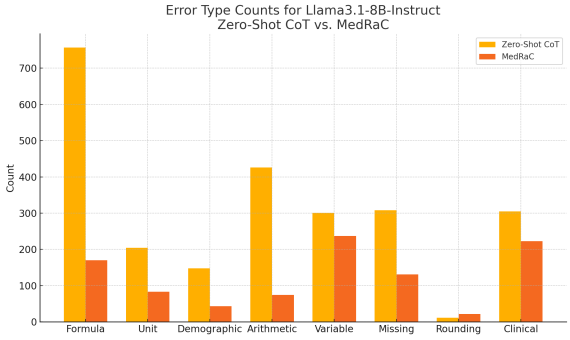

Figure 4: Error Type Counts for Llama3.1-8B-Instruct

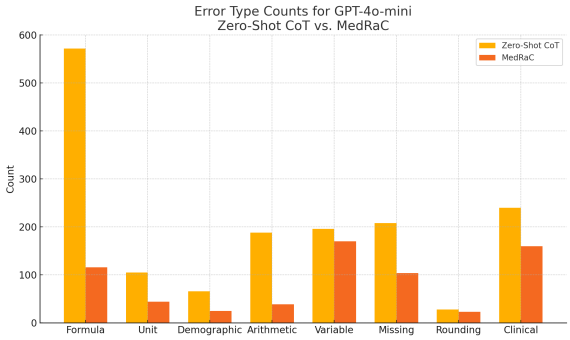

Figure 6: Error Type Counts for gpt-4o-mini

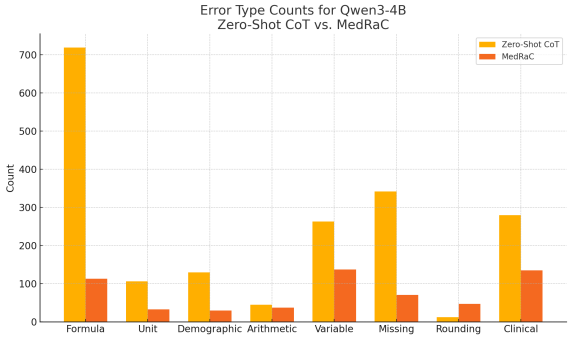

Figure 7: Error Type Counts for qwen3-4B

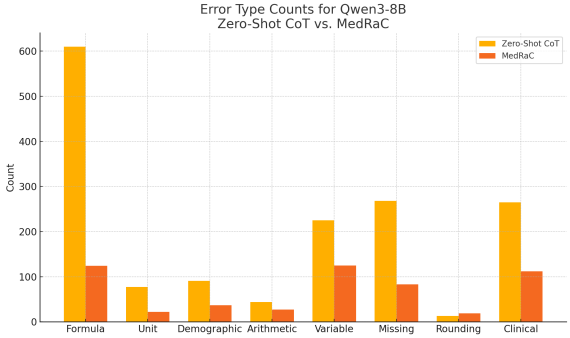

Figure 8: Error Type Counts for qwen3-8B

F Additional Ablation Results

We additionally report the evaluation of the code component by GPT-4.1 and GPT-4o-mini.

Specialty	MedRaC	CoT	MedPrompt	Self-Refine	One-shot
Nephrology	**198 / 219 (90.41%)**	87 / 219 (39.73%)	15 / 219 (6.85%)	97 / 219 (44.29%)	169 / 219 (77.17%)
Cardiology	**166 / 236 (70.34%)**	76 / 236 (32.20%)	82 / 236 (34.75%)	103 / 236 (43.64%)	153 / 236 (64.83%)
Thrombosis/Hematology	**45 / 59 (76.27%)**	17 / 59 (28.81%)	17 / 59 (28.81%)	25 / 59 (42.37%)	44 / 59 (74.58%)
Pulmonology & Critical Care	78 / 100 (78.00%)	40 / 100 (40.00%)	33 / 100 (33.00%)	54 / 100 (54.00%)	**93 / 100 (93.00%)**
Hepatology/Gastroenterology	43 / 65 (66.15%)	11 / 65 (16.92%)	8 / 65 (12.31%)	23 / 65 (35.38%)	**50 / 65 (76.92%)**
Endocrinology & Metabolism	**110 / 122 (90.16%)**	80 / 122 (65.57%)	41 / 122 (33.61%)	91 / 122 (74.59%)	105 / 122 (86.07%)
Obstetrics & Gynecology	**38 / 40 (95.00%)**	36 / 40 (90.00%)	27 / 40 (67.50%)	36 / 40 (90.00%)	27 / 40 (67.50%)
Infectious Disease	**26 / 39 (66.67%)**	16 / 39 (41.03%)	8 / 39 (20.51%)	10 / 39 (25.64%)	19 / 39 (48.72%)
Clinical Pharmacology	**26 / 40 (65.00%)**	2 / 40 (5.00%)	22 / 40 (55.00%)	4 / 40 (10.00%)	22 / 40 (55.00%)
General Practice/Family Medicine	8 / 20 (40.00%)	2 / 20 (10.00%)	1 / 20 (5.00%)	3 / 20 (15.00%)	**9 / 20 (45.00%)**

Table 8: Correct counts and percentages across specialties (best per specialty in bold).

Evaluation	MedRaC	w/o Code
Formula Error (FE) ↓		
DeepSeek-R1 (reasoning)	3.23	31.88
GPT-4.1	2.69	7.02
GPT-4o-mini (reasoning)	0.85	16.95
Calculation Correctness (CC) ↑		
DeepSeek-R1 (reasoning)	97.82	76.52
GPT-4.1	98.97	96.38
GPT-4o-mini (reasoning)	99.53	86.95

Table 9: Ablation results of MedRaC with and without the Code component. FE = Formula Error (lower is better), CC = Calculation Correctness (higher is better).

G Case study: Hallucinations in LLM's answer and how our LLM Eval pipeline detects them

Basic record (Row 369, *Sodium Correction for Hyperglycemia* calculator)

Patient 57-year-old **male**; $\text{Na}_{\text{meas}} = 127$ mmol/L (127 mEq/L); glucose $= 527$ mg/dL

Clinical note
Diabetic foot with massive hyperglycaemia and haemodynamic instability (full narrative in dataset)

Question "What is the corrected sodium concentration (mEq/L) using the Hillier 1999 equation at admission?"

Gold-standard reasoning

Hillier's formula (**?**):

$$\text{Na}_{\text{corr}} = \text{Na}_{\text{meas}} + 0.024\,(\text{glucose} - 100).$$

$$127 + 0.024\,(527 - 100) = \mathbf{137.248}\ \text{mEq/L}.$$

Baseline tolerance (dataset,±6.8624): $[130.39,\ 144.11]$ mEq/L.

LLM original answer (excerpt)

"Corrected sodium (mEq/L) = 127 + 0.016×527 $= 127 + 8.432 = 135.432$ *mEq/L."*

Why the baseline benchmark says "Correct"

The Medcalc benchmark inspects only whether the final number lies within the broad interval above. Because $135.432 \in [130.39, 144.11]$, the response is labelled **"Correct"**, even though the equation is mis-specified.

Our stricter numeric rule

We judge the final figure to the next decimal place beyond the model's precision (max. two places). The LLM output has **three** decimals, so we round to two and require agreement within ± 0.005:

$$|\,135.43 - 137.25\,| = 1.82\ >\ 0.005,$$

hence the answer is **incorrect** despite passing the coarse range check.

Step-by-step LLM Eval verdict

Component	Result	Key comment
Formula selection	Incorrect	Used $0.016 \times$ glucose and omitted -100.
Entity extraction	Correct	Na $=127$, glucose $=527$ captured accurately.
Arithmetic steps	Correct	$0.016 \times 527 = 8.432$, addition correct.
Final answer (precision-aware)	Incorrect	$135.432 \neq 137.248$ under strict tolerance.
Overall	**Incorrect**	Hidden equation error & numeric miss flagged.

Clinical significance

A two-point sodium underestimate may appear minor, but in critically ill, haemodynamically unstable patients, such mis-corrections can drive inappropriate fluid or insulin therapy. Our granular pipeline reveals both the hallucinated coefficient and the subtle numeric shortfall, preventing a misleading "pass" and supporting *clinically defensible* deployment of LLMs.

H Prompt Templates

In this appendix, we present the exact prompt templates used in our evaluation pipeline. All prompts follow a structured format consisting of a system message and a user message. For prompts related to reasoning variants such as **Direct**, **CoT**, **Oneshot**, and **Self-Refine**, please refer to our released code.

H.1 LLM Evaluation Pipeline Prompt

Prompt for LLM Eval Pipeline

```

    def _gen_eval_prompt(self, answer, reference, name_of_step):
        # System message is the same for all steps
        system_msg = (
            "You are a medical calculation assistant. Evaluate whether each
                step is correct by comparing it to the gold-standard reference
                ."
        )

        # For calculation steps, omit the gold-standard reference entirely
        if name_of_step == "calculation":
            user_msg = (
                f"{name_of_step.capitalize()} to be evaluated:\n{answer}\n\n"
                "Note: Judge ONLY the mathematical correctness of each
                    arithmetic "
                "step (addition, subtraction, multiplication, division, powers
                    , "
                "roots, etc.). Do NOT assess whether the formula used is
                    appropriate "
                "or whether the input values were correct or reasonable. Treat
                     small rounding or "
                "decimal-precision differences as acceptable"
                'Respond in this JSON format:\n\n'
                '{"result": "Correct" or "Incorrect", "explanation": "Brief
                    justification."}'
            )
            return system_msg, user_msg

        # For all other steps, include the gold-standard reference first
        user_msg = (
            f"{name_of_step.capitalize()} to be evaluated:\n{answer}\n\n"
            f"Gold-standard reference (fully correct):\n{reference}\n\n"
            "Determine if the given part is correct according to the Gold-
                standard reference. "
            'Respond in this JSON format:\n\n'
            '{"result": "Correct" or "Incorrect", "explanation": "Brief
                justification."}'
        )

        if name_of_step == "formula":
            user_msg += (
                "\n\n"
                "Note: Judge ONLY whether the mathematical formula or scoring
                    standard invoked is appropriate. Do NOT evaluate:"
                "the specific values plugged into the formula,"
                "the correctness of any later calculations."
                "If the gold-standard reference lists multiple valid variants
                    (e.g., male vs. female, different ethnicities), the answer
                     is considered correct as long as it correctly applies ANY
                     one of those variants. If the provided formula includes
                    more detail than the gold-standard reference but the
                    overlapping portion is consistent and correct, it should
                    still be considered correct."
            )
        elif name_of_step == "extracted_values":
            user_msg += (
                "\n\n"
```

```
                "Note: Check if all variables given in the gold-standard reference are found or implied. "
                "Ignore any naming discrepancies, as long as the meaning is the same. "
                "It is ok if the answer has more variables than the gold-standard reference."
                "If the given answer has a different unit than the gold-standard answer, please do conversion first. "
                "Answers with reasonable rounding errors MUST be considered Correct."
            )
        elif name_of_step in ("answer", "final_answer"):
            user_msg += (
                "\n\n"
                "Note: You ONLY need to check whether the final numerical answer matches the provided gold-standard reference. "
                "The correctness of the intermediate steps does NOT matter. If one has a unit and the other does not, please ignore the unit. "
                "If the given answer has a different unit than the gold-standard answer, please do conversion first. "
                "Answers with rounding to the nearest integer and reasonable computational deviations MUST be considered Correct."
            )

        return system_msg, user_msg
```

H.2 LLM Judge for Error Types

H.2.1 Formula Error

Formula Error Prompts

```

def build_formula_error_prompts(
    ground_truth_formulas: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for gt, ans in zip(ground_truth_formulas, answers):
        system_message = SYS_MSG.format(error_type="Formula Error")
        user_message = (
            f"Ground-Truth Formula:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Evaluate whether the formula or scoring system used in the answer is appropriate and correctly constructed for the given clinical context.\n\n"
            "You must check for the following issues:\n"
            "- **Incorrect Formula Selection**: A completely wrong formula is used for the clinical question (e.g., using Cockcroft-Gault for AKI instead of CKD-EPI).\n"
            "- **Internal Formula Construction Errors**: The selected formula appears intended to be correct but is flawed in structure or logic. Look for:\n"
            "   Incorrect or missing coefficients or constants\n"
            "   Wrong mathematical operators (e.g., `*` instead of `^`)\n"
            "   Misused parentheses, terms in wrong places, or reversed logic\n"
            "   Hallucinated or fabricated terms in formula\n"
            "   Fabricated or omitted scoring items (e.g., omitting "recent surgery" in the Wells Score, or adding a non-existent item like "family history")\n\n"
            "**Important Notes:**\n"
            "- Do NOT evaluate variable extraction correctness here.\n"
            "- Do NOT evaluate numerical calculation or rounding accuracy.\n"
```

```
            "- If multiple formula variants exist and the answer uses any
                valid one, it is acceptable.\n"
            "- If the answer includes extra details but the core formula is
                correct, that is acceptable.\n\n"
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.2 Variable Error

Variable Error Prompts

```


def build_variable_extraction_error_prompts(
    patient_notes: List[str],
    questions: List[str],
    ground_truth_Extracted_values: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for note, q, gt, ans in zip(patient_notes, questions,
        ground_truth_Extracted_values, answers):
        system_message = SYS_MSG.format(error_type="Incorrect Variable
            Extraction Error")
        user_message = (
            f"Patient Note:\n{note}\n\n"
            f"Question:\n{q}\n\n"
            f"Ground-Truth Variable Extraction:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Determine whether the answer incorrectly extracted key
                variables from the patient note.\n\n"
            "You should look for the following possible errors:\n"
            "1. **Wrong value**: The extracted value (e.g., heart rate,
                creatinine) does not match the patient note.\n"
            "2. **Wrong unit**: The extracted unit is misinterpreted (e.g.,
                _mol/L mistaken for mg/dL).\n"
            "3. **Wrong instance**: Multiple similar values exist (e.g., lab
                values from different days), and the wrong one was selected.\n
                \n"
            "**Do NOT evaluate:**\n"
            "- Whether the formula chosen is appropriate and correct\n"
            "- Do NOT judge whether the final answer is correct, focus only on
                 the value extraction part.\n"
            "- Whether the numerical calculation is accurate\n\n"
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.3 Misinterpretation Error

Misinterpretation Error Prompts

```

def build_clinical_misinterpretation_prompts(
    patient_notes: List[str],
    questions: List[str],
    ground_truth_explanations: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
```

```
    prompts = []
    for note, q, gt, ans in zip(patient_notes, questions,
        ground_truth_explanations, answers):
        system_message = SYS_MSG.format(error_type="Clinical Misinterpretation
            Error")
        user_message = (
            f"Patient Note:\n{note}\n\n"
            f"Question:\n{q}\n\n"
            f"Scoring Rubric and corresponding result:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Evaluate whether the clinical meaning of each finding was
                interpreted correctly based on the scoring rubric and patient
                note.\n\n"
            "This error type reflects a misunderstanding of medical knowledge
                or common clinical reasoning, leading to incorrect
                interpretation of the patient's symptoms or findings.\n\n"
            "You should check for the following types of errors:\n"
            "1. **Incorrect severity classification** (e.g., mild vs. severe
                ascites)\n"
            "2. **Wrong presence/absence judgment** (e.g., assigning points
                for recent surgery when not present)\n"
            "3. **Incorrect threshold interpretation** (e.g., age >75
                incorrectly treated as <75)\n"
            "4. **Misunderstanding clinical terms or context** (e.g.,
                interpreting 'occasional alcohol use' as 'chronic alcohol
                abuse')\n\n"
            "**Important Notes:**\n"
            "- The variable values may be correctly extracted from the note,
                but the error lies in the clinical judgment or
                misclassification.\n"
            "- Do NOT evaluate the correctness of the scoring formula, numeric
                computation, or unit conversion.\n"
            "- If the clinical inference depends on subtle wording or
                ambiguity in the note, highlight that in your explanation.\n\n
                "
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.4 Missing Variable Error

Missing Variable Error Prompts

```

def build_missing_variable_prompts(
    patient_notes: List[str],
    questions: List[str],
    ground_truth_Extracted_values: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for note, q, gt, ans in zip(patient_notes, questions,
        ground_truth_Extracted_values, answers):
        system_message = SYS_MSG.format(error_type="Missing Variable
            Extraction Error")
        user_message = (
            f"Patient Note:\n{note}\n\n"
            f"Question:\n{q}\n\n"
            f"Ground-Truth Variable Extraction:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Identify whether the answer failed to extract or include
                one or more variables that are necessary to perform the
                correct calculation.\n\n"
            "You should look for cases where:\n"
```

```
            "1. A required input variable is completely missing.\n"
            "2. The model skipped over variables because they were ambiguous
                or not explicitly stated.\n"
            "3. The answer proceeds with partial information, leaving out
                fields that the formula or score requires.\n\n"
            "**Do NOT evaluate:**\n"
            "- Whether the formula used is correct\n"
            "- Whether the extracted variables are accurate\n"
            "- Whether the final calculation is numerically correct\n\n"
            "Focus only on whether the model omitted key inputs needed to
                properly execute the formula or scoring rule.\n"
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.5 Unit Error

Unit Error Prompts

```


def build_unit_conversion_error_prompts(
    patient_notes: List[str],
    questions: List[str],
    ground_truth_explanations: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for note, q, gt, ans in zip(patient_notes, questions,
        ground_truth_explanations, answers):
        system_message = SYS_MSG.format(error_type="Unit Conversion Error")
        user_message = (
            f"Patient Note:\n{note}\n\n"
            f"Question:\n{q}\n\n"
            f"Ground-Truth Explanation:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Evaluate whether any input variable was used with the wrong
                unit, or skipped unit conversion when required by the formula
                .\n\n"
            "You should look for the following types of errors:\n"
            "1. The value is used directly without converting to the expected
                unit (e.g., using creatinine 134 _mol/L directly in a formula
                that expects mg/dL).\n"
            "2. The conversion is attempted but the result is wrong (e.g.,
                wrong conversion factor or direction).\n"
            "3. The unit label is misunderstood or misinterpreted (e.g.,
                confusing mEq/L with mmol/L).\n\n"
            "**Do NOT evaluate:**\n"
            "- Whether the formula chosen is appropriate\n"
            "- Whether the correct value was extracted from the note\n"
            "- Whether the afterwards numerical computation was otherwise
                accurate\n\n"
            "Only evaluate whether the units used match those required by the
                formula, and whether any necessary conversions were done
                correctly.\n"
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.6 Demographic Error

Demographic Error Prompts

```

def build_adjustment_coefficient_error_prompts(
    patient_notes: List[str],
    questions: List[str],
    ground_truth_explanations: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for note, q, gt, ans in zip(patient_notes, questions,
        ground_truth_explanations, answers):
        system_message = SYS_MSG.format(error_type="Missing or Misused
            Demographic/Adjustment Coefficient Error')
        user_message = (
            f'Patient Note:\n{note}\n\n'
            f'Question:\n{q}\n\n'
            f'Ground-Truth Explanation:\n{gt}\n\n'
            f'Answer to be evaluated:\n{ans}\n\n'
            'Task: Evaluate whether demographic- or context-based adjustment
                coefficients were properly applied in the formula.\n\n'
            'Specifically, check for:\n'
            '1. Missing adjustment - A coefficient required by the formula is
                missing (e.g., sex multiplier is omitted)'
            '2. Incorrect coefficient used - The formula includes a
                coefficient, but it does not match the patient's
                characteristics (e.g., using male factor for a female patient)
                '
            '3. Incorrect demographic inference - The model assumes the wrong
                demographic category (e.g., classifying patient as non-Black
                when clearly stated otherwise)'
            'Common adjustment dimensions may include:'
            '- Sex (e.g., male vs. female)\n'
            '- Race/ethnicity (e.g., Black vs. non-Black)'
            '- Age thresholds\n'
            '- Pregnancy status\n'
            '- Weight class (e.g., obese vs. normal weight)'
            'Do NOT evaluate:\n'
            '- Formula structure or selection\n'
            '- Variable extraction accuracy\n'
            '- Unit conversion correctness\n'
            '- Final numerical calculation\n\n'
            'Return a STRICT JSON response: '
            '{'error_present': 'Yes' or 'No', 'explanation': ''}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.7 Arithmetic Error

Arithmetic Error Prompts

```

def build_arithmetic_error_prompts(
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for  ans in answers:
        system_message = SYS_MSG.format(error_type="Arithmetic Error")
        user_message = (
            f"Answer to be evaluated:\n{ans}\n\n"
```

```
            "Task: All variables, units, and formula structure are assumed to
                be correct.\n"
            "Your task is to verify whether the **arithmetic computation**
                itself is correct.\n\n"
            "Check for:\n"
            "1. Basic arithmetic errors (e.g., 4 + 3 = 6)\n"
            "2. Wrong order of operations (e.g., using left-to-right instead
                of proper parentheses)\n"
            "3. Errors in exponentiation, multiplication, or division\n"
            "4. Missing or duplicated numeric terms\n\n"
            "**Do NOT evaluate:**\n"
            "- Formula selection or structure\n"
            "- Variable extraction\n"
            "- Unit conversion\n"
            "- Rounding or precision formatting\n\n"
            "If the calculation process is entirely accurate, a reasonable
                margin of error is acceptable."
            "Return a STRICT JSON response: "
            '{"error_present": "Yes" or "No", "explanation": ""}.'
        )
        prompts.append((system_message, user_message))
    return prompts
```

H.2.8 Rounding Error

Rounding Prompts

```


def build_rounding_error_prompts(
    ground_truth_explanations: List[str],
    answers: List[str],
) -> List[Tuple[str, str]]:
    prompts = []
    for gt, ans in zip(ground_truth_explanations, answers):
        system_message = SYS_MSG.format(error_type="Rounding / Precision Error
            ")
        user_message = (
            f"Ground-Truth Explanation:\n{gt}\n\n"
            f"Answer to be evaluated:\n{ans}\n\n"
            "Task: Determine whether the numeric **final result** in the
                answer is imprecise due to **rounding or insufficient decimal
                precision**, "
            "even if the formula used and the overall arithmetic are mostly
                correct.\n\n"
            "This error type should be marked when rounding errors or
                insufficient precision in intermediate or final steps "
            "cause the final answer to fall outside the tolerance range.\n\n"
            "**Rules for evaluating precision:**\n"
            "- Use the number of decimal places in the LLM's final answer to
                determine the expected precision (up to a maximum of 2 decimal
                 places).\n"
            "- If the answer is ‘10.65‘ _ round to **2 decimal places**,
                tolerance _0.005\n"
            "- If the answer is ‘10.7‘  _ round to **1 decimal place**,
                tolerance _0.05\n"
            "- If the answer is ‘10.6512‘ _ still round to **2 decimal places
                **, tolerance _0.005 (overprecision beyond 2 d.p. does **not**
                 increase accuracy expectations)\n\n"
            "**DO mark as a Rounding / Precision Error if:**\n"
            "- The calculation is mostly correct but rounding was done
                incorrectly (e.g., too few decimals)\n"
            "- The result deviates from the ground truth only because the
                final answer lacks the required precision (per above rules)\n\
                n"
            "**DO NOT mark as Rounding / Precision Error if:**\n"
```

```
            "- The formula used is incorrect\n"
            "- The arithmetic calculation is wrong\n"
            "- The answer is completely off due to conceptual misunderstanding
                \n\n"
            "Return a STRICT JSON response with:\n"
            '{"error_present": "Yes" or "No", "explanation": ""}'
        )
        prompts.append((system_message, user_message))
    return prompts
```