MLDERIVE: BENCHMARKING MACHINE LEARNING REASONING WITH EVIDENCE-GUIDED EVALUATION

Anonymous authors

ABSTRACT

Solving machine learning (ML) problems requires large language models (LLMs) to apply domain knowledge through multi-step reasoning under problem-specific assumptions and constraints. Yet, how these models perform across diverse ML tasks and what specific reasoning errors drive their failures remain insufficiently characterized. We introduce MLDerive, a benchmark comprising 1,292 human-reviewed problems with reference solutions across various task types and ML knowledge areas. MLDerive links curated source problems to knowledge-aligned variants, enabling comparisons within source–variant families. To reliably assess model outputs, we propose an evidence-guided evaluator that combines tool verification, dependency-aware target checking, and semantic adjudication to determine whether solution claims jointly establish the requested conclusions, while retaining traceable evidence. Evaluation of six LLMs reveals task-level differences: all models achieve lower pass rates on optimization and algorithmic derivation than on proof tasks. Furthermore, qualitative analysis identifies three recurring reasoning failures: method misapplication, inferential insufficiency, and calculation-explanation discrepancies. Ultimately, MLDerive integrates quantitative performance, family coverage, and qualitative error analysis to provide a comprehensive framework for assessing ML reasoning.

1 INTRODUCTION

Large language models (LLMs) are becoming part of how scientific knowledge is accessed, understood, and applied. In machine learning (ML), their potential role extends from explaining concepts and assisting with implementation to analyzing models and deriving methods. These activities require coherent arguments that connect domain knowledge to the target problem. Evaluating whether models can construct such solutions is therefore important for understanding the support they can provide in ML research and education.

Existing benchmarks assess different components relevant to ML problem solving. General mathematical and scientific problem-solving benchmarks examine capabilities such as symbolic manipulation, calculation, and multi-step reasoning (Hendrycks et al., 2021; Wang et al., 2024; Ma et al., 2026). ML-focused evaluations cover domain knowledge (Pricope, 2025; Islam et al., 2026), engineering performance (Chan et al., 2025), open-ended examinations (Drori et al., 2023), and formal proof completion in ML theory (Yang et al., 2026). Together, these efforts provide complementary perspectives on ML capabilities and a foundation for assessing reasoning in complete solutions. Yet the ability to integrate ML knowledge and mathematical reasoning into complete, natural-language solutions remains less systematically evaluated.

Studying this capability requires both suitable problems and informative judgments. Broad coverage is needed to reveal differences across task types and knowledge areas. Linked problems enable assessment of whether success on a source problem is accompanied by success on all its variants. Assessing the resulting solutions also requires attention to how an argument is constructed, as emphasized by work on process-error detection and fine-grained proof evaluation (Zheng et al., 2025; Ma et al., 2026). ML solutions may involve backpropagation in neural networks, EM updates in mixture models, and feature-map construction in kernel methods, combining symbolic manipulation, gradient computation, probabilistic inference, and optimization. Verifying these different types of claims requires appropriate mathematical and computational tools, with explicit assumptions and scope for each check. However, correct intermediate computations do not by themselves establish

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the validity of a complete solution. For example, correctly computing posterior responsibilities in a mixture model does not establish that the subsequent EM parameter update satisfies the required parameter constraints or solves the M-step optimization problem. Evaluation must examine the dependencies among these results and the conditions under which they hold to determine whether they jointly support the requested conclusions.

We introduce MLDerive, a benchmark consisting of 1,292 human-reviewed problems with reference solutions. It spans probabilistic modeling, statistical inference, optimization, neural networks, and their foundations, with tasks including formula computation, analytical derivation, proof, optimization, and algorithmic derivation. The four datasets—DL-Source, PRML-Source, DL-Variants, and PRML-Variants—combine curated source problems with variants built around related knowledge. The variants broaden the range of problem settings by changing the objects, conditions, or requested outputs while retaining links to their sources. This organization supports both broad capability profiles and more detailed comparisons within related problem families.

To assess model solutions, we develop an evidence-guided evaluator that organizes key claims and their dependencies into a solution graph. Tool verification provides local evidence for dependency-aware target checking, while semantic adjudication examines unresolved steps and disputed evidence using the problem context and the supported portions of the argument. The final assessment retains traceable evidence for its judgment, including any target-relevant errors or unresolved requirements.

Our evaluation of six LLMs on MLDerive reveals granular performance and coverage disparities that aggregate pass rates obscure. Across all models, optimization and algorithmic derivation tasks consistently yield lower success rates compared to proof tasks. Furthermore, high overall performance often masks incomplete variant coverage within successfully solved source families, with gaps being most pronounced in algorithmic derivations. Qualitative analysis corroborates this, showing that formally correct intermediate expressions frequently conceal unjustified inferences or inconsistent interpretations. Together, these findings underscore the necessity of evaluating task-specific robustness and family-level consistency.

Our contributions can be summarized as:

- **A structured benchmark for ML reasoning.** We construct four human-reviewed datasets spanning diverse ML knowledge areas and reasoning tasks, with source–variant links, providing a unified benchmark for evaluating natural-language ML solutions.
- **An evidence-guided evaluation framework.** We combine tool verification, dependency-aware target checking, and semantic adjudication to assess whether solution steps support the requested targets and to retain traceable evidence.
- **An empirical study of LLM-based ML reasoning.** We evaluate six LLMs across knowledge areas, reasoning tasks, and related problem formulations, revealing performance differences and recurring failures.

2 Related Work

Evaluating machine learning capabilities. Existing evaluations of ML capabilities span domain knowledge, engineering performance, and problem solving. HardML and FLAME assess knowledge and concept application through multiple-choice questions (Pricope, 2025; Islam et al., 2026), while MLE-bench evaluates engineering performance through end-to-end model development (Chan et al., 2025). Drori et al. (2023) study open-ended ML exam solving with human grading and automatic checks, while FormalML evaluates completion of ML proof subgoals in Lean (Yang et al., 2026). These efforts cover important aspects of ML competence, but provide limited coverage of complete natural-language solutions in which problem conditions, intermediate derivations, and requested conclusions must be jointly satisfied.

Scientific problem-solving benchmarks. Scientific problem-solving benchmarks provide established settings for studying multi-step reasoning across technical domains. MATH evaluates competition-level mathematical problem solving with step-by-step reference solutions (Hendrycks et al., 2021), while SciBench extends scientific problem solving to college-level mathematics, physics, and chemistry (Wang et al., 2024). GSM-Symbolic further introduces template-based variants to

examine the sensitivity of reasoning to changes in problem formulation (Mirzadeh et al., 2025). These benchmarks provide broad coverage of mathematical and scientific problem solving and enable analysis of reasoning under related problem formulations. However, ML reasoning introduces a distinct domain setting in which mathematical reasoning must remain grounded in learning-specific concepts, assumptions, objectives, and algorithmic procedures. This domain-specific grounding is less directly addressed by general mathematical and scientific problem-solving benchmarks.

Process-level solution evaluation. Recent work has moved beyond final-answer evaluation toward assessing correctness within the reasoning process. ProcessBench focuses on first-error localization in reasoning traces (Zheng et al., 2025), while ProofBench and ProofGrader support fine-grained assessment of natural-language proofs through expert ratings and automated scoring, respectively (Ma et al., 2026). Pseudo-Formalization decomposes proofs into self-contained blocks for structured verification (Barkallah et al., 2026), and MATH-VF combines formalization with computer algebra and SMT-based checking to verify mathematical statements and provide corrective feedback (Zhou & Zhang, 2025). These approaches establish complementary mechanisms for examining local correctness and reasoning structure, but are primarily developed for general mathematical reasoning tasks. MLDerive evaluates natural-language ML solutions by verifying claims with appropriate computational tools, integrating evidence along claim dependencies, and applying evidence-guided semantic adjudication to determine whether the complete solution establishes the requested conclusions under the given problem conditions.

3 MLDERIVE DATASETS

3.1 BENCHMARK SCOPE

MLDerive evaluates the ability of LLMs to integrate domain knowledge with multi-step reasoning. Formally, given a machine learning problem q, a model generates a natural-language solution s interleaved with necessary mathematical expressions. The evaluation rigorously assesses not only the correctness of the final results, but also the logical validity and completeness of the supporting derivations under the stated assumptions, naturally accommodating alternative valid solution paths.

To support this rigorous evaluation, MLDerive spans a broad spectrum of ML knowledge areas and reasoning requirements. Crucially, the benchmark organizes problems into source–variant families, enabling fine-grained comparisons across related formulations with altered conditions or targets. Furthermore, all problems feature self-contained statements, explicit objective targets, and comprehensive reference solutions, establishing a rigorous foundation for verifying LLM reasoning on ML problem-solving tasks.

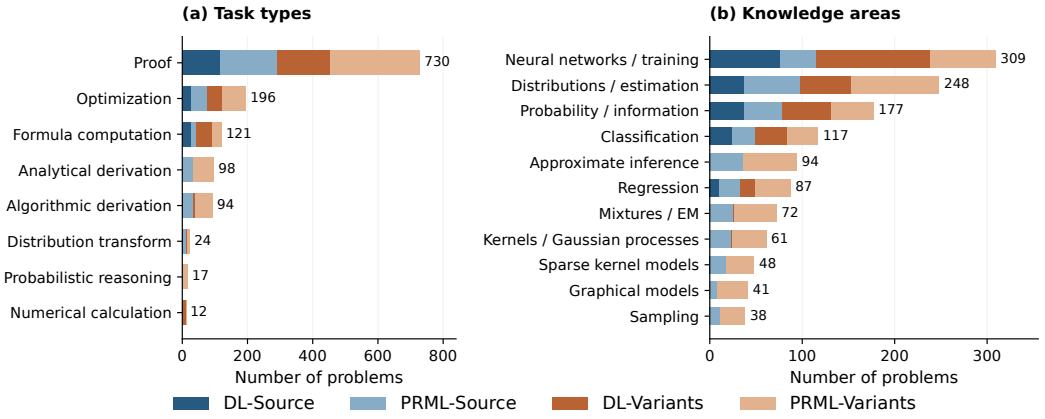

Figure 1: MLDerive dataset composition by (a) task types and (b) knowledge areas.

3.2 Sources and Coverage

MLDerive comprises 1,292 problems across four datasets: DL-Source, PRML-Source, DL-Variants, and PRML-Variants. DL-Source draws from Chapters 2–10 of *Deep Learning: Foundations and Concepts* (Bishop & Bishop, 2024); PRML-Source draws from Chapters 1–11 of *Pattern Recognition and Machine Learning* (Bishop, 2006). DL-Variants and PRML-Variants contain knowledge-aligned variants of exercises from the corresponding sources. Figure 1 summarizes the dataset distribution of MLDerive. The datasets span multiple task types across a broad range of ML knowledge areas, including distributional modeling, regression and classification, neural networks, graphical models, and approximate inference.

3.3 Construction and Review

Figure 2 illustrates the construction and review pipeline for the source problems and knowledge-aligned variants. We collect 198 DL exercises and 339 PRML exercises together with available solutions. DL solutions come from the authors' *Solutions to the Exercises*, version 1.0, distributed through the book website.[1] PRML solutions are obtained from the third-party solution manual maintained by Zhengqi Gao.[2]

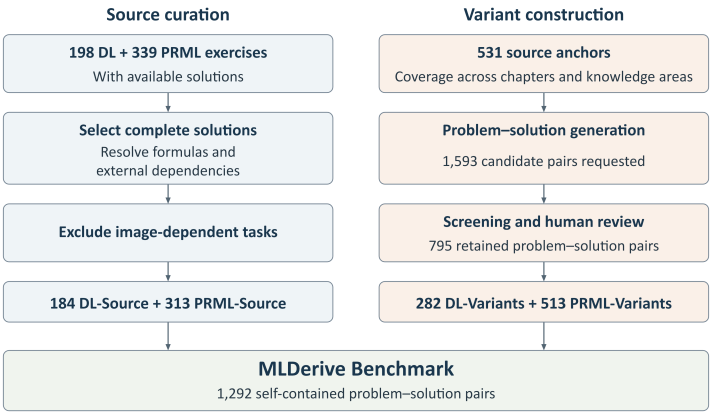

Figure 2: MLDerive dataset construction pipeline.

Self-contained source construction. We exclude exercises that require visual input or graphical output. For the remaining exercises, we resolve numbered equation references using the source textbook and incorporate the referenced formulas into the corresponding problem statements or reference solutions. Additionally, we replace cross-references to other sections or exercises with the necessary definitions and supporting arguments. We standardize mathematical notation, correct transcription errors, and explicitly specify variable roles, dimensions, domains, and task requirements. Finally, we jointly review and revise each problem statement and its reference solution to ensure consistency and eliminate dependencies on textbook-specific context.

Knowledge-aligned variant generation. We use the selected 531 anchor problems for variant generation. For each anchor, `GPT-5.6-Luna` extracts a structured specification of the underlying concepts, reasoning operations, mathematical objects, assumptions, target conclusions, and solution outline. The prompt requires modifications along at least two dimensions, such as representation, inference direction, mathematical object family, intermediate objectives, or proof structure.

Quality control and dataset curation. We screen generated candidates for structural validity, self-containment, alignment with target specifications, and diversity. `GPT-5.6-Luna` re-solves and evaluates each candidate against predefined acceptance thresholds, with at most one revision cycle permitted per candidate. This process yields 795 accepted candidates from 1,593 generation requests.

[1] `https://www.bishopbook.com/`

[2] `https://github.com/zhengqigao/PRML-Solution-Manual`

An ML researcher reviews all problem statements and reference solutions across the four dataset components for well-posedness, mathematical correctness, and solution completeness. After final curation, DL-Variants and PRML-Variants contain 282 and 513 problems, respectively.

4 Evidence-Guided Evaluator

Given a problem and a candidate solution, MLDerive assesses whether the key claims hold and jointly establish the requested conclusions. The evaluator constructs a solution graph represented as a directed acyclic graph (DAG) of dependencies, selects tools to verify individual claims, and checks how the resulting evidence supports the problem targets. Semantic adjudication examines the remaining gaps and disputed claims in the context of the supported argument; its findings inform the final target assessment and solution-level judgment. Figure 3 details the overall generation and evaluation pipeline, while Algorithm 1 in Appendix A.1 summarizes the evaluation procedure.

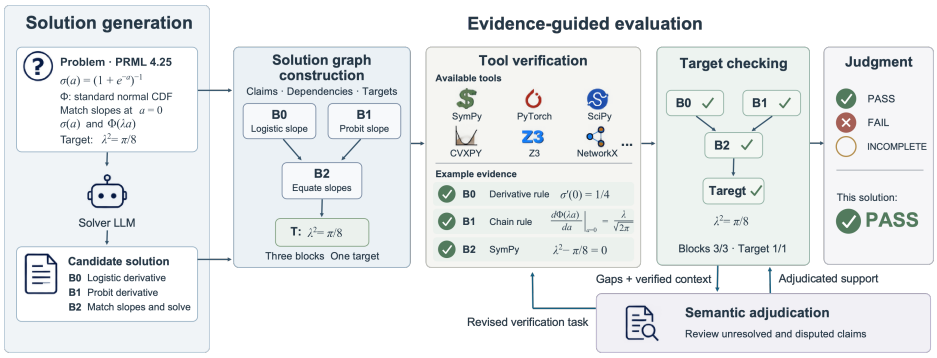

Figure 3: Overview of the MLDerive evaluation framework.

4.1 Solution Graph Construction

Assessment distinguishes the premises a solution may use, the targets T it must establish, and the constraints on numerical or symbolic results. Evaluator-side context records explicit conditions, definitions, function signatures, and applicable standard conventions. Together, these specifications determine what counts as a valid solution to the problem.

The solution graph is a DAG $G = (P \cup B, E)$ connecting admitted premises P with material claim blocks B extracted from the candidate solution. Each block retains its source text, claim, conditions, and dependencies and is associated with the requested targets T it supports. The DAG preserves both the reuse of an intermediate result across subsequent steps and the joint support of multiple premises for a conclusion. It connects each local check to the targets it supports, enabling the evaluator to trace the consequences of a gap or error through the argument. Appendix A.3 provides an example for PRML Exercise 4.25.

4.2 Tool Verification and Target Checking

Verification tasks are dynamically routed to tools based on the operations and properties involved in each claim. Symbolic computation checks algebraic relations and derivatives; automatic differentiation checks gradients; constraint solving tests implications under explicit conditions. Specialized computational checks address probability distributions, statistical quantities, optimization, and inference. For numerical or symbolic results, comparison proceeds from normalized matching to numerical or symbolic equivalence checks. Appendix A.2 provides the task–tool mapping (Table 2) and graph-construction details. Verification records $\mathcal{V}$ retain supported, refuted, or unresolved outcomes, together with the assumptions and scope of each check.

Dependency-aware target checking determines whether locally supported claim blocks form a complete argument for the requested conclusions. For each block $b \in B$, $\mathrm{dep}(b) \subseteq P \cup B$ denotes its direct dependencies. Let $S \subseteq B$ contain locally supported claim blocks, based on applicable

tool evidence in $\mathcal{V}$ and, when available, admitted semantic findings. We define node usability $U_S : P \cup B \to \{0, 1\}$ recursively as

$$U_S(p) = 1, \qquad p \in P,$$
$$U_S(b) = \mathbf{1}[b \in S] \wedge \left(\bigwedge_{u \in \mathrm{dep}(b)} U_S(u) \right), \qquad b \in B. \tag{1}$$

Here, $\mathbf{1}[\cdot]$ is the indicator function, and $\wedge$ and $\bigwedge$ denote logical conjunction, with the empty conjunction defined as true. Admitted premises are treated as given; a claim block is usable only when it is locally supported and all its dependencies are usable. For a fixed graph and support set, this recursion deterministically propagates dependency usability.

A target is successfully verified when its supporting blocks are verified, and their conclusions collectively derive the required result while satisfying all relevant conditions. The initial evaluation integrates these dependency and requirement checks to flag unresolved or refuted claims along the derivation paths, as well as objectives lacking sufficient justification. These identified gaps and disputed claims constitute the residual set H, which is forwarded to the semantic adjudication module alongside the validated context.

4.3 Semantic Adjudication

The evaluator assesses unresolved claims and missing justifications in the context of the supported argument, using the problem, submitted solution, admitted context, and local evidence. For a missing justification, it supplies a local completion only if the omitted step follows from the available premises and supported intermediate results without introducing any key arguments. Where needed, it clarifies the applicable conventions. For a tool-reported refutation, it checks whether the tool tested the intended claim under applicable conditions and whether the contradiction affects a required conclusion. This distinguishes substantive errors from formalization errors and defects in noncritical side claims. Semantic adjudication also assesses result equivalence left unresolved by tool checks. Revised verification tasks are returned to the tools when needed, and the updated evidence and adjudicated support inform target checking.

4.4 Final Judgment

The final judgment synthesizes tool evidence, dependency analysis, and semantic adjudication to determine whether the solution fulfills the problem requirements. A solution is assigned a `PASS` only if all required results and supporting arguments are established and no target-critical errors remain.

5 Experiments

We evaluate six recently released LLMs across the four datasets: `Claude Sonnet 5` (Anthropic, 2026), `Gemini 3.1 Pro` (Google DeepMind, 2026), `DeepSeek V4 Pro` and `DeepSeek V4 Flash` (DeepSeek-AI et al., 2026), `Kimi K3` (Kimi Team et al., 2026), and `Qwen 3.8 Max` (Alibaba Group, 2026). To ensure a fair comparison, each model is prompted exclusively with the self-contained problem statement. We apply a unified system instruction requiring rigorous mathematical language, complete step-by-step derivations in LaTeX notation, and clearly stated conclusions (the full prompt is detailed in Appendix B.1). Solutions are then assessed following the pipeline in Section 4. Specifically, we employ `GPT-5.6-Luna` as the evaluator backbone for all LLM-driven stages, retaining tool evidence and dependency records to ensure traceable judgments.

5.1 Evaluator Validation

Evaluator backbone selection. We compare `GPT-5.6-Luna`, `DeepSeek V3`, and `MiniMax M3` as alternative LLM backbones for the MLDerive evaluator, testing each across 184 DL-Source solutions generated by `Claude Sonnet 5`. Employing these models across all LLM-driven evaluation stages yields pass rates of 90.22%, 94.02%, and 96.74%, respectively. Human review of disagreement cases reveals that the `GPT-5.6-Luna`-driven evaluator most reliably detects substantive reasoning flaws while robustly accepting valid equivalent formulations. In contrast, the higher

pass rate of MiniMax M3 stems primarily from over-lenience (failing to penalize flawed reasoning), whereas DeepSeek V3 suffers from both missed errors and unwarranted rejections. Table 6 in Appendix C details these behavioral differences, motivating our adoption of GPT-5.6-Luna as the default evaluation engine.

Run-to-run stability. To assess the stability of the GPT-5.6-Luna-driven evaluator, we conduct three independent evaluation runs on a fixed set of 1,292 solutions generated by Claude Sonnet 5 across all four datasets. The pipeline demonstrates high consistency, yielding overall pass rates of 87.69%, 88.47%, and 89.24%. This corresponds to a mean pass rate of 88.47% with a minimal standard deviation of just 0.77 percentage points.

Human agreement. We construct a paired validation set of 100 solutions to 50 problems sampled proportionally from the four datasets. Each pair comprises a correct solution and one featuring a deliberately introduced error. These errors cover a range of reasoning flaws. Three ML researchers independently annotate the correctness of each solution. Our evaluator achieves 98%, 98%, and 99% agreement with the respective reviewers, strongly supporting its ability to distinguish valid derivations from those with substantive errors.

Comparison with an LLM-as-a-judge baseline. We compare the evidence-guided MLDerive evaluator against a direct LLM-as-a-judge baseline (both utilizing GPT-5.6-Luna as the backbone) on 1,292 solutions generated by Claude Sonnet 5. On a stratified, expert-annotated subset of 200 solutions, MLDerive achieves 93% human agreement, outperforming the baseline (86.5%) by 6.5 percentage points. Across the full dataset, the solution-level pass rates are 87.69% for MLDerive and 89.32% for the baseline. Qualitatively, while the baseline's higher pass rate masks significant erratic judgments, MLDerive's dependency-aware assessment effectively mitigates such errors, including spurious rejections (detailed in Appendix C).

5.2 Overall Model Performance

Table 1 summarizes the performance of six models on 1,292 problems, with overall pass rates ranging from 87.69% to 96.83%. Qwen 3.8 Max achieves the highest pass rate on all four subsets, followed overall by Kimi K3 and DeepSeek V4 Pro at 95.74% and 95.36%, respectively. Relative model performance varies between source problems and variants, revealing differences obscured by aggregate rankings. Solution length does not consistently track performance: DeepSeek V4 Pro produces the shortest solutions yet outperforms three models with longer outputs.

Table 1: Overall model performance, measured by solution-level pass rates (%) and average solution lengths (characters). Bold indicates the highest pass rates.

	Source		Variants			
Model	DL	PRML	DL	PRML	Overall	Length (chars)
Problems	184	313	282	513	1,292	
Claude Sonnet 5	90.22	90.73	86.17	85.77	87.69	5,620.8
Gemini 3.1 Pro	91.85	89.14	94.68	92.01	91.87	5,142.9
DeepSeek V4 Pro	94.02	93.61	96.81	96.10	95.36	3,548.3
DeepSeek V4 Flash	92.93	92.33	95.39	94.54	93.96	3,803.9
Kimi K3	94.57	93.61	95.39	97.66	95.74	3,838.5
Qwen 3.8 Max	**96.74**	**94.89**	**97.16**	**97.86**	**96.83**	6,103.5

5.3 Performance Across Tasks and Knowledge Areas

Task-level performance. All six models achieve higher pass rates on proof tasks than on optimization and algorithmic derivation (Figure 4a), with rates pooled across models of 95.73%, 87.67%, and 88.30%, respectively. Model differences are more pronounced in algorithmic derivation: pass rates range from 74.47% to 95.74%, compared with 90.96% to 98.36% for proof, corresponding to spreads of 21.28 and 7.40 percentage points. Relative rankings also vary by task: DeepSeek V4 Flash

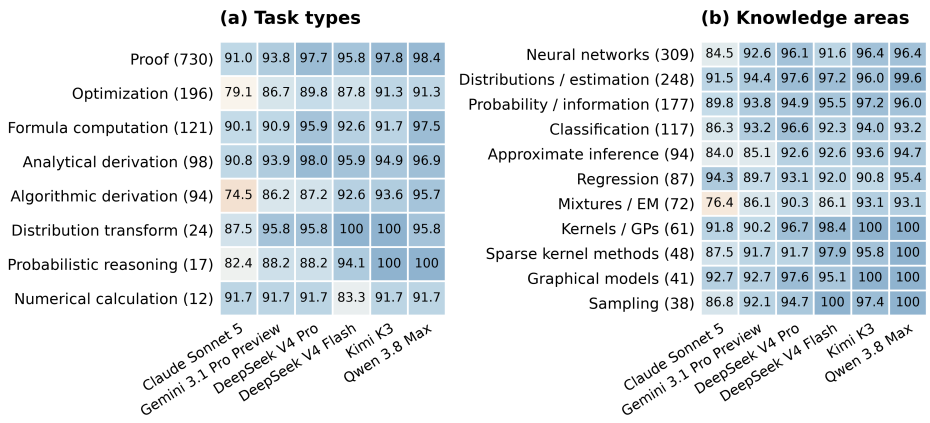

Figure 4: Pass rates (%) across (a) primary task types and (b) chapter-derived knowledge areas.

passes more algorithmic-derivation problems than `DeepSeek V4 Pro` (87/94 vs. 82/94), despite the latter's higher overall pass rate.

Knowledge-area performance. Across all knowledge areas, the six models consistently achieve higher pass rates on distributions and estimation compared to mixture models (EM) and approximate inference (Figure 4b). The corresponding pooled pass rates are 96.03%, 87.50%, and 90.43%, respectively. Despite these shared overarching trends, individual models exhibit distinct domain-specific strengths. For instance, `Claude Sonnet 5` outperforms `Gemini 3.1 Pro` on regression (94.25% vs. 89.66%), whereas `Gemini 3.1 Pro` demonstrates superior performance on neural networks (92.56% vs. 84.47%). Appendix B.2 details the complete numerical breakdowns.

5.4 Performance Across Source–Variant Families

We compare source–variant families, assigning equal weight to each family. Variant pass rates are first averaged within each family and then across families. Among families with a passing source solution, a coverage gap occurs when at least one variant does not pass.

Source and variant performance. Figure 5a shows comparable source and variant pass rates for most models, suggesting that their understanding of underlying principles supports adaptation across related problem settings. `Claude Sonnet 5`'s pass rate decreases from 91.13% on sources to 85.34% on variants, whereas `Kimi K3` and `Qwen 3.8 Max` achieve variant pass rates of 96.47% and 97.17%, respectively. `DeepSeek V4 Pro` and `DeepSeek V4 Flash` have identical source pass rates of 93.84%, but `DeepSeek V4 Pro` attains a higher variant pass rate (96.72% versus 94.46%). These paired results distinguish models with similar source performance and reveal different patterns of success across related problems.

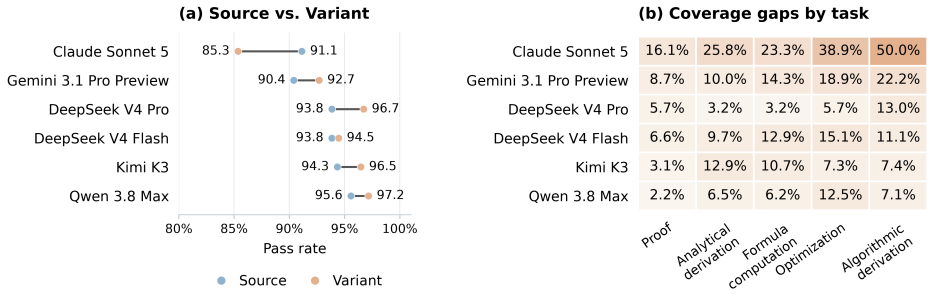

Figure 5: Source–variant performance and coverage gaps. (a) Source and variant pass rates. (b) Percentage of source-passing families containing at least one non-passing variant.

Task-specific gaps after source success. Figure 5b delineates these coverage gaps across distinct task categories. Conditioned on a passing source solution, gaps emerge significantly more often in algorithmic derivation (17.95%) and optimization (16.36%) families compared to proof families (6.97%). This disparity persists across all evaluated LLMs, with `Claude Sonnet 5` exhibiting the most pronounced contrast. Consequently, algorithmic procedures and optimization tasks remain primary bottlenecks for complete family coverage, even when the source problem is successfully solved. Furthermore, we observe that model consistency diverges based on task types. Within optimization tasks, `DeepSeek V4 Pro` demonstrates superior robustness over `DeepSeek V4 Flash`, reducing the coverage-gap rate from 15.09% to 5.66%. We also identify complementary task-specific strengths between other leading models: `Qwen 3.8 Max` achieves more comprehensive coverage in proof families, whereas `Kimi K3` performs better in optimization.

5.5 Reasoning Error Patterns

To diagnose the underlying ML reasoning failures, we analyze model-generated solutions. We identify three recurring error patterns, illustrate each with representative cases below, and provide details in Appendix D.

Applicability and problem fidelity. At the level of problem specification, solutions sometimes apply familiar methods without fully accounting for the conditions and estimation targets specified by the problem. Applicability errors occur when a method fails to accommodate the stated conditions; failures of problem fidelity occur when the derivation changes the problem's assumptions or unknown quantities. In the mixture-weight example, the proposed update handles lower bounds for empty clusters but overlooks nonempty clusters whose bounds can also be active. In the covariance-update example, treating an estimated mean as fixed yields a coherent derivation for a different estimation problem. These cases reveal difficulties in adapting familiar procedures to the full problem specification (Appendix D.1).

Inferential sufficiency. At the level of inference, solutions sometimes draw conclusions stronger than their intermediate results justify. Such errors arise when a sufficient condition is treated as necessary or a local property is extended to a global conclusion without establishing the required assumptions. In the residual-cancellation example, the solution treats the vanishing of an aggregate residual as equivalent to the vanishing of every individual term. Termwise vanishing is sufficient, but nonzero terms can also cancel in the aggregate. The intermediate expressions can therefore be correct while the claimed equivalence is false. These cases reveal gaps in controlling the logical strength of conclusions drawn from established results (Appendix D.2).

Computational and explanatory consistency. At the level of execution and interpretation, solutions sometimes follow an appropriate overall approach but introduce errors in the calculations, constructions, or explanations that realize it. In the exponential-kernel example, `Claude Sonnet 5` applies normalization twice when constructing an explicit feature map, yielding an incorrect kernel representation (Appendix D.3). Furthermore, textual interpretation can also diverge from a correct expression: parameter growth of order D^M is described as exponential in D, although it is polynomial for fixed M. These cases reveal failures to maintain consistency between the intended construction, its mathematical realization, and the interpretation of the resulting expression.

These error patterns underscore that a reasoning gap can invalidate an entire solution trajectory, despite partial correctness. Consequently, we characterize effective ML reasoning as a rigorous, end-to-end process of applying, connecting, and validating knowledge throughout a solution.

6 Conclusion

In this work, we present MLDerive, a benchmark for evaluating open-ended ML reasoning in LLMs. MLDerive combines human-reviewed problems and knowledge-aligned variants with an evidence-guided evaluator that integrates tool verification, dependency-aware target checking, and semantic adjudication. By providing evidence-guided assessments of model solutions, MLDerive enables a reliable and systematic evaluation of LLM reasoning on ML tasks.

AI USE STATEMENT

We utilized generative AI tools to assist with literature retrieval and manuscript text refinement. Notably, generative AI was also employed to drive dataset synthesis (specifically constructing problem variants) and the evaluation of candidate solutions, both core components of our research methodology. All AI-assisted outputs were reviewed and verified by the authors. The authors take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Section 3 describes the dataset sources, construction, human review, and source–variant links. Sections 4 and 5 specify the evaluation procedure, experimental protocol, judgment criteria, and family-level metrics. Appendix A provides evaluation details and a worked example. Appendix B contains the generation prompt and detailed results. Appendices C and D present mathematical checks of evaluator disagreements and model reasoning errors. Datasets and code are provided in the supplementary materials.

REFERENCES

Alibaba Group. Alibaba unveils Qwen3.8-Max: Its largest and most capable flagship model to date, 2026. URL https://www.alibabagroup.com/en-US/document-2021044032125272064.

Anthropic. Introducing Claude Sonnet 5, 2026. URL https://www.anthropic.com/news/claude-sonnet-5.

Slim Barkallah, Luke Bailey, Kaiyue Wen, Mohammed Abouzaid, and Tengyu Ma. Pseudo-formalization for automatic proof verification. *arXiv preprint arXiv:2605.20531*, 2026. URL https://arxiv.org/abs/2605.20531.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2024. doi: 10.1007/978-3-031-45468-4. URL https://www.bishopbook.com/.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Iddo Drori, Sarah J. Zhang, Reece Shuttleworth, Sarah Zhang, Keith Tyser, Zad Chin, Pedro Lantigua, Saisamrit Surbehera, Gregory Hunter, Derek Austin, Leonard Tang, Yann Hicke, Sage Simhon, Sathwik Karnik, Darnell Granberry, and Madeleine Udell. From human days to machine seconds: Automatically answering and generating machine learning final exams. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3947–3955. ACM, 2023. doi: 10.1145/3580305.3599827. URL https://arxiv.org/abs/2206.05442.

Google DeepMind. Gemini 3.1 Pro model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Mohammed Saidul Islam, Negin Baghbanzadeh, Farnaz Kohankhaki, Afshin Cheraghi, Ali Kore, Shayaan Mehdi, Elham Dolatabadi, and Arash Afkanpour. Fine-grained benchmark generation for comprehensive evaluation of foundation models. *arXiv preprint arXiv:2605.18824*, 2026. URL https://arxiv.org/abs/2605.18824.

Kimi Team et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Wenjie Ma, Andrei Cojocaru, Neel Kolhe, Haihan Zhang, Vincent Zhuang, Matei Zaharia, and Sewon Min. Reliable fine-grained evaluation of natural language math proofs. In *International Conference on Learning Representations*, volume 2026, pp. 72740–72778, 2026.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05229.

Tidor-Vlad Pricope. HardML: A benchmark for evaluating data science and machine learning knowledge and reasoning in AI. *arXiv preprint arXiv:2501.15627*, 2025. URL https://arxiv.org/abs/2501.15627.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. SciBench: Evaluating college-level scientific problem-solving abilities of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50622–50649. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24z.html.

Xiao-Wen Yang, Zihao Zhang, Jianuo Cao, Zhi Zhou, Zenan Li, Lan-Zhe Guo, Yuan Yao, Taolue Chen, Yu-Feng Li, and Xiaoxing Ma. FormalML: A benchmark for evaluating formal subgoal completion in machine learning theory. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.02335.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Kuo Zhou and Lu Zhang. Step-wise formal verification for LLM-based mathematical problem solving. *arXiv preprint arXiv:2505.20869*, 2025. URL https://arxiv.org/abs/2505.20869.

A Evaluation Details

A.1 Evaluation Procedure

Algorithm 1 summarizes the evidence-guided evaluation. The binary outcome is `PASS` or `NON-PASS`; diagnostic records distinguish confirmed critical errors (`FAIL`) from unestablished requirements without a confirmed critical error (`INCOMPLETE`) within `NON-PASS`.

Algorithm 1 Evidence-guided assessment of a candidate solution

Input: Problem q, candidate solution s, evaluator context C
Output: Judgment $y \in \{\texttt{PASS}, \texttt{FAIL}, \texttt{INCOMPLETE}\}$

Extract admitted premises P and required targets T from (q, C)
Construct the solution graph $G = (P \cup B, E)$ from s, using P, T, and (q, C)
Initialize verification records $\mathcal{V}$
Local tool verification
for each claim block $b \in B$ **do**
    Select tools appropriate for b
    Verify the claim in b under its stated assumptions and dependencies
    Store the supported, refuted, or unresolved outcome and its scope in $\mathcal{V}[b]$
end for
Dependency-aware target checking
Identify locally supported claim blocks $S \subseteq B$ from $\mathcal{V}$
Propagate usability U_S in G and check the requirements of each $t \in T$
Collect residual gaps and disputed claims H
Semantic adjudication
if $H \neq \varnothing$ **then**
    Review H using (q, s, C), G, $\mathcal{V}$, and the supported context
    if a verification task is incorrectly formalized **then**
        Revise the task and rerun the relevant tools
        Update the corresponding verification records in $\mathcal{V}$
    end if
    Update S from $\mathcal{V}$ and admitted semantic findings; update dependencies in G
    Recompute U_S and recheck the requirements of each $t \in T$
end if
Final judgment
if all required results and arguments are established and no admitted target-critical error remains **then**
    return `PASS`
else if an admitted target-critical error remains **then**
    return `FAIL`
else
    return `INCOMPLETE`
end if

A.2 VERIFICATION TOOLS

Table 2 maps claim types to supported verification tools. Each result retains the assumptions and scope of its check; numerical evidence applies to the tested instances.

Table 2: Verification tasks, supported tools, and their evidence.

Task	Tool	Check	Evidence
Algebra and calculus	`SymPy`	Expressions, derivatives, integrals, matrix relations	Symbolic relations under stated conditions
Constraint reasoning	`Z3`	Entailment under formalized premises	SMT entailment or countermodel
Statistical computation	`NumPy` / `SciPy`	Distribution quantities, integration, linear algebra	Numerical agreement or discrepancy
Gradient verification	`PyTorch`	Automatic differentiation	Computed gradients for comparison
Convex optimization	`CVXPY`	Specified convex optimization instances	Solver status and numerical solutions
Probabilistic inference	`pgmpy`	Discrete inference and factor calculations	Computed marginals and factor values
Graph structure	`NetworkX`	DAG structure and dependency paths	Structural validity and reachability

The evaluator decomposes each solution into claim blocks. These blocks represent argumentative units and may contain multiple calculation steps. The claim-block count $|B|$ measures the granularity of the solution decomposition; it excludes admitted premises and separately recorded targets.

A.3 EVALUATION EXAMPLE

Figure 6 illustrates the assessment of PRML Exercise 4.25. The solution matches the slopes of the logistic function and a scaled probit function at the origin. The logistic slope is $1/4$, whereas the slope of $\Phi(\lambda a)$ is $\lambda/\sqrt{2\pi}$, where Φ is the standard normal cumulative distribution function. Equating them gives $\lambda = \sqrt{\pi/8}$ for positive scaling. The derivative claims form blocks `B0` and `B1`, checked using derivative and chain-rule schemas, respectively. Block `B2` depends on both and uses the equal-slope condition to derive $\lambda^2 = \pi/8$, with the algebra checked by `SymPy`. Dependency-aware target checking then establishes the required target $\lambda^2 = \pi/8$ from these usable blocks. This example has three claim blocks and one separately recorded target.

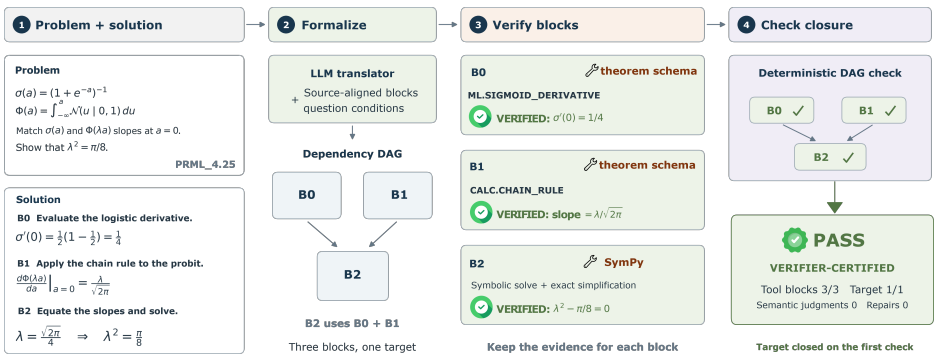

Figure 6: An example of MLDerive's evidence-guided evaluation for PRML Exercise 4.25.

B Experimental Details and Results

B.1 Solution Generation

All solvers receive the following system instruction and the problem as the user message.

```
You are a top-tier expert in machine learning and statistics.
Please solve the provided machine learning exercises.
Follow these instructions:

1. Use rigorous mathematical language and provide a complete
step-by-step derivation.

2. Wrap all mathematical variables and formulas in standard LaTeX
using $ or $$.

3. Clearly state the final conclusion or proof result.
```

B.2 Task and Knowledge-Area Results

Tables 3 and 4 provide the numerical values behind Figure 4. Task labels identify the primary task type of each problem. Knowledge areas follow source chapters, with each variant inheriting its source's area. Each problem contributes once per model; the tables report category sizes alongside pass rates.

B.3 Solution-Level and Claim-Block Outcomes

Table 5 pairs solution-level pass rates with the proportion of locally accepted claim blocks in non-passing solutions. The former measure completion of the whole task; the latter describes how much of the represented argument has local support when the overall judgment is `NON-PASS`.

For solution i, let B_i be its set of material claim blocks and $S_i \subseteq B_i$ the locally supported blocks after tool verification and semantic adjudication. Local acceptance means membership in S_i. Let $\hat{y}_i$ denote its recorded judgment and $\mathcal{N} = \{i : \hat{y}_i \neq \texttt{PASS}, |B_i| > 0\}$. We report the mean per-solution claim-block acceptance rate

$$R_{\text{block}|\text{non-PASS}} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \frac{|S_i|}{|B_i|}.$$

Solution-level pass rates include all problems. Claim-block acceptance includes both tool and semantic support; it does not require dependency usability or target establishment. Admitted premises and targets are not counted in this metric.

Across the 479 non-passing solutions with a solution graph, the mean accepted claim-block fraction is 81.39%; the block-weighted counterpart is 82.21% (2,666/3,243). Thus, a non-passing solution often retains substantial locally supported reasoning. The unresolved or refuted portion can nevertheless contain a decisive inference.

C Evaluator Disagreement Analysis

Disagreements across evaluator backbones. Table 6 presents illustrative disagreements on DL-Source when `GPT-5.6-Luna`, `DeepSeek V3`, and `MiniMax M3` serve as alternative backbones of the same MLDerive evaluation pipeline. The cases contrast the evaluator configurations' judgments with human assessments.

Disagreements between MLDerive evaluator and LLM-as-a-judge. We present one illustrative case from the disagreements between MLDerive and the direct LLM-as-a-judge baseline on solutions generated by `Claude Sonnet 5`.

For `PRML_GEN_3_22_V2_1fe6a62c`, the baseline returns `FAIL`, whereas MLDerive returns `PASS`. The solution derives a stationary-point update for the noise precision in Bayesian linear

Table 3: Pass rates (%) by task type. N denotes the number of problems per model. Avg. denotes the mean pass rate across the six models.

Category	N	Claude Sonnet 5	Gemini 3.1 Pro	DeepSeek V4 Pro	DeepSeek V4 Flash	Kimi K3	Qwen 3.8 Max	Avg.
Proof	730	90.96	93.84	97.67	95.75	97.81	**98.36**	95.73
Optimization	196	79.08	86.73	89.80	87.76	**91.33**	**91.33**	87.67
Formula computation	121	90.08	90.91	95.87	92.56	91.74	**97.52**	93.11
Analytical derivation	98	90.82	93.88	**97.96**	95.92	94.90	96.94	95.07
Algorithmic derivation	94	74.47	86.17	87.23	92.55	93.62	**95.74**	88.30
Distribution transform	24	87.50	95.83	95.83	**100.00**	**100.00**	95.83	95.83
Probabilistic reasoning	17	82.35	88.24	88.24	94.12	**100.00**	**100.00**	92.16
Numerical calculation	12	**91.67**	**91.67**	**91.67**	83.33	**91.67**	**91.67**	90.28

Table 4: Pass rates (%) by chapter-derived knowledge area. Avg. denotes the mean pass rate across the six models. Bold marks the highest rates.

Category	N	Claude Sonnet 5	Gemini 3.1 Pro	DeepSeek V4 Pro	DeepSeek V4 Flash	Kimi K3	Qwen 3.8 Max	Avg.
Neural networks	309	84.47	92.56	96.12	91.59	**96.44**	**96.44**	92.93
Distributions / estimation	248	91.53	94.35	97.58	97.18	95.97	**99.60**	96.03
Probability / information	177	89.83	93.79	94.92	95.48	**97.18**	96.05	94.54
Classification	117	86.32	93.16	**96.58**	92.31	94.02	93.16	92.59
Approximate inference	94	84.04	85.11	92.55	92.55	93.62	**94.68**	90.43
Regression	87	94.25	89.66	93.10	91.95	90.80	**95.40**	92.53
Mixtures / EM	72	76.39	86.11	90.28	86.11	**93.06**	**93.06**	87.50
Kernels / GPs	61	91.80	90.16	96.72	98.36	**100.00**	**100.00**	96.17
Sparse kernel methods	48	87.50	91.67	91.67	97.92	95.83	**100.00**	94.10
Graphical models	41	92.68	92.68	97.56	95.12	**100.00**	**100.00**	96.34
Sampling	38	86.84	92.11	94.74	**100.00**	97.37	**100.00**	95.18

Table 5: Solution-level pass rates and mean per-solution claim-block acceptance rates among non-passing solutions (%). n is the number of non-passing solutions with a solution graph used for the claim-block statistic.

Model	Solution PASS	Non-PASS block rate	n
DL-Source (184 problems)			
Claude Sonnet 5	90.22	74.09	17
Gemini 3.1 Pro	91.85	78.52	12
DeepSeek V4 Pro	94.02	79.50	9
DeepSeek V4 Flash	92.93	76.86	10
Kimi K3	94.57	79.37	7
Qwen 3.8 Max	96.74	73.00	5
PRML-Source (313 problems)			
Claude Sonnet 5	90.73	73.03	29
Gemini 3.1 Pro	89.14	78.34	34
DeepSeek V4 Pro	93.61	77.43	20
DeepSeek V4 Flash	92.33	78.27	23
Kimi K3	93.61	68.06	16
Qwen 3.8 Max	94.89	72.02	15
DL-Variants (282 problems)			
Claude Sonnet 5	86.17	84.96	39
Gemini 3.1 Pro	94.68	88.73	15
DeepSeek V4 Pro	96.81	88.78	9
DeepSeek V4 Flash	95.39	87.50	13
Kimi K3	95.39	89.84	13
Qwen 3.8 Max	97.16	93.53	8
PRML-Variants (513 problems)			
Claude Sonnet 5	85.77	82.90	73
Gemini 3.1 Pro	92.01	85.69	41
DeepSeek V4 Pro	96.10	89.32	20
DeepSeek V4 Flash	94.54	82.55	28
Kimi K3	97.66	85.44	12
Qwen 3.8 Max	97.86	81.44	11
Pooled	93.58	81.39	479

Table 6: Examples of disagreements across evaluator backbones on DL-Source.

Case	Human review	`GPT-5.6-Luna`	`DeepSeek V3`	`MiniMax M3`
DL_5.23	FAIL; replaces the explicitly defined function	FAIL	PASS	PASS
DL_6.12	PASS; benign omission	PASS	FAIL	FAIL
DL_9.18	INCOMPLETE; unjustified uniqueness	INCOMPLETE	PASS	PASS

regression. With design matrix $\Phi \in \mathbb{R}^{N\times M}$ and precisions $\alpha, \beta > 0$, it establishes

$$C_\beta = \beta^{-1} I_N + \alpha^{-1}\Phi\Phi^\top \succ 0, \qquad N - \gamma_\beta = \beta^{-1}\,\mathrm{Tr}(C_\beta^{-1}),$$

where γ_β is the effective number of parameters. The baseline rejects a subsequent division by $N - \gamma_\beta$ because positivity is not proved separately. The displayed identity already gives $N - \gamma_\beta > 0$, since C_β^{-1} is positive definite and has positive trace. MLDerive connects this established property to the final rearrangement.

D Reasoning Error Analysis

The following checks develop the three error patterns discussed in Section 5.5, linking submitted claims to explicit calculations and counterexamples.

D.1 Applicability and Problem Fidelity

A normalized mixture update outside the feasible set. For `PRML_GEN_9_11_V1_72e6eda5`, the mixture-weight subproblem is

$$\max_{\pi} \sum_{k=1}^{K} N_k \log \pi_k, \qquad \sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq \delta,$$

where the hard-assignment counts satisfy $N_k \geq 0$, $\sum_k N_k > 0$, and $0 < \delta < 1/K$. `Gemini 3.1 Pro` assigns weight δ to each empty cluster and distributes the remaining mass proportionally among nonempty clusters. For counts $(0, 1, 99)$ and $\delta = 0.1$, this gives

$$\pi^{\mathrm{cand}} = (0.1, 0.009, 0.891).$$

The weights sum to one, but the second component violates its lower bound.

The KKT conditions, with multipliers $\mu_k \geq 0$, give

$$\frac{N_k}{\pi_k} - \lambda + \mu_k = 0, \qquad \mu_k(\pi_k - \delta) = 0, \qquad \pi_k^* = \max\{\delta, N_k/\lambda\}, \quad \sum_k \pi_k^* = 1.$$

Here the optimum is $\pi^* = (0.1, 0.1, 0.8)$, with $\lambda = 123.75$ and $\mu = (123.75, 113.75, 0)$. These values satisfy feasibility, stationarity, and complementary slackness; concavity gives global optimality. A nonempty cluster can activate its lower bound, so handling only empty clusters is insufficient. This feasibility check was performed after evaluation; the recorded non-passing judgment concerned a separate tied-centre argument.

Replacing an estimated mean with a fixed mean. In `PRML_2.37`, `Claude Sonnet 5` derives a covariance update under a known, fixed mean, whereas the problem requires the estimated mean to change as observations arrive. For observations $0, 2$, the maximum-likelihood variance around their estimated mean is 1; averaging squared deviations from a fixed mean of zero gives 2. The substituted derivation therefore estimates a different quantity.

D.2 Inferential Sufficiency

Aggregate cancellation does not require termwise cancellation. For `DL_GEN_8_11_V1_b2415dec`, `DeepSeek V4 Flash` derives the Hessian $H(w)$ and generalized Gauss–Newton

matrix $G(w)$ of a summed softmax cross-entropy loss. Let $p_r = \mathrm{softmax}(a_r(w))$, with fixed simplex targets q_r and twice continuously differentiable logits. Their difference is

$$H(w) - G(w) = O(w) = \sum_{r=1}^{M} O_r(w), \qquad O_r(w) = \sum_{k=1}^{K} (p_{rk}(w) - q_{rk}) \nabla_w^2 a_{rk}(w).$$

The solution asserts that $O(w) = 0$ if and only if $O_r(w) = 0$ for every observation.

Take $M = K = 2$, a scalar parameter w, logits $a_1(w) = a_2(w) = (w^2, 0)$, and targets $q_1 = (0, 1)$ and $q_2 = (1, 0)$. At $w = 0$, both probability vectors equal $(1/2, 1/2)$ and the logit second derivatives are $(2, 0)$. Thus

$$O_1(0) = 1, \qquad O_2(0) = -1, \qquad O(0) = 0.$$

The individual residuals cancel. Subtracting the common logit shift $w^2/2$ gives centered logits $(w^2/2, -w^2/2)$ and leaves the residuals unchanged. Termwise vanishing is therefore sufficient, but not necessary, for equality of the Hessian and generalized Gauss–Newton matrix at a given parameter value.

Independence is stronger than variance additivity requires. In `PRML_1.10`, `Qwen 3.8 Max` correctly expands the variance but states that additivity requires independence. For finite-variance variables,

$$\mathrm{Var}\left(\sum_i X_i\right) = \sum_i \mathrm{Var}(X_i) + 2 \sum_{i<j} \mathrm{Cov}(X_i, X_j).$$

Pairwise zero covariance is sufficient. Independence is a stronger sufficient condition, not a necessary one.

D.3 Computational and Explanatory Consistency

Repeated scaling in an exponential-kernel feature map. For `PRML_GEN_6_6_V2_4623161e`, the task requires an explicit feature representation of $\exp(k(x, y))$ for a finite-valued real symmetric positive-semidefinite kernel k. Writing $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$, `Claude Sonnet 5` defines both

$$\langle u, v \rangle_{\mathcal{G}} = \sum_{m=0}^{\infty} \frac{1}{m!} \langle u_m, v_m \rangle_{\mathcal{H}^{\otimes m}}, \qquad \Psi_m(x) = \frac{\phi(x)^{\otimes m}}{\sqrt{m!}},$$

with $\mathcal{H}^{\otimes 0} = \mathbb{R}$ and $\phi(x)^{\otimes 0} = 1$. Substitution gives

$$\langle \Psi(x), \Psi(y) \rangle_{\mathcal{G}} = \sum_{m=0}^{\infty} \frac{k(x, y)^m}{(m!)^2},$$

which applies the factorial normalization twice. For the valid constant kernel $k(x, y) = 1$, the expression is approximately 2.2795853023, rather than $e \approx 2.7182818285$. A correct construction uses the ordinary Hilbert direct-sum inner product with the scaled components above. The resulting feature map has squared norm $\exp(k(x, x)) < \infty$. Equivalently, the weighted inner product can be paired with unscaled tensor components.

An incorrect tangent propagated through a correct rule. For `DL_GEN_8_15_V3_a6026581`, `DeepSeek V4 Flash` propagates $\dot{w}_i = \partial w_i / \partial a$ through a computation graph with independent real inputs $(w_1, w_2, w_3) = (a, b, c)$ and input tangents $(1, 0, 0)$. The relevant nodes are

$$w_4 = a + b, \qquad w_5 = a - c, \qquad w_9 = a + c,$$
$$w_{10} = \sin(w_9), \quad w_{11} = w_4 w_5 + \log(1 + e^b), \quad H = w_{11} w_{10}.$$

The solution writes the correct addition rule $\dot{w}_9 = \dot{w}_1 + \dot{w}_3$, but evaluates it as 2 rather than $1 + 0 = 1$. The correct output derivative is

$$\frac{\partial H}{\partial a} = (w_4 + w_5) w_{10} + w_{11} \cos(w_9).$$

The submitted expression instead doubles its second term. Setting $b = c = 0$ gives $H(a, 0, 0) = (a^2 + \log 2) \sin a$, whose derivative at $a = 0$ is $\log 2$; the submitted derivative gives $2 \log 2$.

Arithmetic and interpretation errors. In `DL_GEN_8_8_V2_c677e6cb`, `Kimi K3` obtains three matrix contributions correctly but combines an entry as $1 + 1 + 2 = 3$ instead of 4, changing the final matrix and determinant. In `PRML_1.16`, `Claude Sonnet 5` correctly gives parameter growth of order D^M for fixed M, but describes its dependence on D as exponential rather than polynomial. These cases concern consistency between intermediate calculations, the final expression, and its interpretation.