PlannerCritic: A Self-Refining LLM Agent Architecture for End-to-End Data Science Pipelines

Anonymous Author(s)

Abstract

AI Data Scientist agents that automate end-to-end data science pipelines face a fundamental challenge: errors made in early stages (data exploration, feature engineering) compound through downstream stages (modeling, evaluation), and single-pass planning architectures have no mechanism to recover. We propose **PlannerCritic**, a two-module framework in which a *Planner* decomposes a task into a directed acyclic graph (DAG) of subtasks and an *Critic* executes each subtask, scores outputs on multiple quality dimensions, and generates structured natural-language feedback that drives targeted re-planning. We prove that under a Lipschitz error-propagation model of data science pipelines, iterative critic feedback reduces cumulative expected error at a rate of $O(K^{-2/3})$ in the number of refinement iterations K, strictly faster than the $O(K^{-1/2})$ rate achievable by independent single-pass replanning. Empirically, PlannerCritic with GPT-4o achieves 57.2% task success on DataSciBench [12] and 60.1% on MLAgentBench [6]—improvements of +13.1 pp and +22.6 pp over the ReAct [11] baseline and +8.3 pp and +16.9 pp over the strongest prior agent, SELA [1]. Ablations confirm that both the Critic scoring and the Planner's use of feedback are necessary: removing either component causes a 3.6–17.9 pp drop in success rate.

CCS Concepts

• **Computing methodologies** → **Intelligent agents**; *Machine learning*.

Keywords

AI data scientist, LLM agent, automated data science, self-refinement, pipeline planning, critic feedback

ACM Reference Format:
Anonymous Author(s). 2026. PlannerCritic: A Self-Refining LLM Agent Architecture for End-to-End Data Science Pipelines. In *Proceedings of 1st International Workshop on AI Data Scientist (AI-DS@KDD'26).* ACM, New York, NY, USA, 5 pages.

1 Introduction

The prospect of AI systems that can conduct data science autonomously—from loading raw data to delivering a trained model with a written evaluation report—has attracted significant research attention [5, 10]. Benchmark results, however, reveal a persistent gap: even the best current agents achieve only ∼40–50% task success on rigorous end-to-end benchmarks [6, 12]. We identify a structural cause: *error compounding across pipeline stages.*

A typical data science pipeline passes data sequentially through exploration, cleaning, feature engineering, modeling, and evaluation. An error at any stage—a miscoded missing-value imputation, a target-leaked feature, a mis-specified loss function—silently propagates to all downstream stages. Existing agent architectures based on ReAct [11] or plan-then-execute [10] generate a plan *once* and execute it, with no structured mechanism to detect and repair stage-level errors before they compound.

Our approach. We propose **PlannerCritic**, a two-module architecture:

- The **Planner** generates a DAG of subtasks from the user's natural-language task description, assigning typed inputs and expected output schemas to each node.
- The **Critic** executes each subtask, then evaluates the output on up to five structured quality dimensions (correctness, completeness, schema conformance, downstream viability, and efficiency) and returns a structured feedback record. The Planner ingests this feedback and optionally revises the subtask graph before proceeding.

This design localizes error detection to each stage, preventing downstream compounding, and provides the Planner with actionable revision signals rather than raw execution traces.

Contributions.

(1) We introduce PlannerCritic, a Planner–Critic architecture for AI Data Scientists that interleaves planning and structured critique (Section 3).
(2) We prove that iterative critic feedback reduces cumulative pipeline error at rate $O(K^{-2/3})$, outperforming the $O(K^{-1/2})$ rate of independent single-pass replanning (Section 4, Theorem 2).
(3) We achieve state-of-the-art task success on DataSciBench and MLAgentBench, with detailed ablations confirming the necessity of both modules (Section 5).

2 Background and Related Work

LLM agents for data science. Data Interpreter [5] frames data science as a hierarchical DAG of code-execution tasks, achieving strong results on single-session problems. AutoML-Agent [10] uses specialized sub-agents for each pipeline stage. SELA [1] applies tree-search to expand the plan space. None of these systems uses a dedicated Critic that scores stage outputs and feeds structured signals back to a Planner.

Benchmarks. DataSciBench [12] evaluates LLMs on 1,000 data science prompts across six task types using a Task-Function-Code (TFC) framework with both completion rate (CR) and success rate (SR) metrics. MLAgentBench [6] provides 13 end-to-end ML experimentation tasks where an

AI-DS@KDD'26, Jeju, Korea
2026.

agent must read files, write code, and produce a final submission.

Self-refinement in LLMs. Reflexion [9] stores verbal feedback in an episodic memory for multi-trial refinement. Self-Refine [7] iteratively improves outputs using the same model as critic and generator. Neither work addresses the structured, multi-stage setting of data science pipelines, nor provides convergence guarantees.

Automated machine learning. Classical AutoML systems such as Auto-SKLEARN [4] and AutoGluon [3] automate model search and hyperparameter tuning but do not handle the full pipeline from raw data to insight, and require structured tabular input without natural language understanding.

3 The PlannerCritic Architecture

3.1 Problem Formulation

Let $\mathcal{Q}$ be the space of natural-language data science task descriptions and $\mathcal{X}$ be the space of raw datasets. A *data science pipeline* is a sequence of S stages $\pi = (\pi_1, \ldots, \pi_S)$, where each $\pi_s : X_s \to X_{s+1}$ transforms the data from stage s to stage $s+1$, with $X_1 = X_{\text{raw}}$ and X_{S+1} being the final deliverable (model, report, or transformed dataset).

The *pipeline quality* is measured by a scoring function $\Phi : \mathcal{Q} \times X_{S+1} \to [0, 1]$, where $\Phi(q, X_{S+1}) = 1$ indicates a fully correct pipeline output for task q. The AI Data Scientist's goal is to maximize $\mathbb{E}[\Phi(q, X_{S+1})]$ over a distribution of tasks $q \sim \mathcal{D}$.

3.2 Planner Module

Given task description q and dataset X, the Planner $\mathcal{P}$ produces a *task DAG*:

$$G = (V, E, \{\texttt{spec}_v\}_{v \in V}),$$

where each node $v \in V$ is a subtask, each edge $(u, v) \in E$ indicates that the output of u is an input to v, and $\texttt{spec}_v$ contains a natural-language description, expected input/output schemas, and a success criterion for subtask v.

The Planner is implemented as a prompted LLM call: $G = \mathcal{P}_\theta(q, X, \mathcal{H})$, where $\mathcal{H}$ is the history of Critic feedback from prior iterations. We prompt the Planner with a structured template that enforces schema declarations and topological ordering (see Appendix for the prompt template).

3.3 Critic Module

For each subtask v executed in topological order, the Critic $\mathcal{C}$ receives the subtask specification $\texttt{spec}_v$, the code produced by the executor, and the execution output $X_{v,\text{out}}$, and produces a structured feedback record:

$$f_v = \mathcal{C}_\theta(\texttt{spec}_v,\ \texttt{code}_v,\ X_{v,\text{out}}) = (s_v,\ m_v,\ \texttt{revision}_v),$$

where $s_v \in [0, 1]^5$ is a vector of five quality scores (correctness, completeness, schema conformance, downstream viability, efficiency), $m_v \in \{0, 1\}$ is a binary pass/fail flag ($m_v = 1$ iff $\min(s_v) \geq \tau$ for threshold τ), and $\texttt{revision}_v$ is a natural-language revision instruction emitted when $m_v = 0$.

Algorithm 1 PlannerCritic (PlannerCritic)

Require: Task q, dataset X, LLM θ, max iterations $K_{\max}$, pass threshold τ
Ensure: Pipeline output X_{S+1}

$G \leftarrow \mathcal{P}_\theta(q, X, \emptyset)$ ▷ Initial plan
for each subtask v in topological order of G **do**
  for $k = 1$ to $K_{\max}$ **do**
    Execute v to produce $\texttt{code}_v$ and $X_{v,\text{out}}$
    $f_v \leftarrow \mathcal{C}_\theta(\texttt{spec}_v, \texttt{code}_v, X_{v,\text{out}})$
    if $f_v.m_v = 1$ **then break**
    end if
    $G \leftarrow \mathcal{P}_\theta(q, X, \mathcal{H} \cup \{f_v\})$ ▷ Local re-plan
  end for
end for
return pipeline output X_{S+1}

If $m_v = 0$, PlannerCritic triggers a *local re-plan*: the Planner is re-invoked with $\mathcal{H} \leftarrow \mathcal{H} \cup \{f_v\}$ to revise subtask v and any downstream dependents. This repeats for at most $K_{\max}$ iterations per subtask.

The full PlannerCritic procedure is summarized in Algorithm 1.

3.4 Implementation Details

The Planner and Critic are both instantiated as separate system-prompted calls to the same underlying LLM (GPT-4o or LLaMA-3-70B-Instruct in our experiments). Code execution is sandboxed with a 120-second timeout per subtask. We use $K_{\max} = 3$ and $\tau = 0.7$ in all experiments. Subtask types include: *data loading*, *exploratory analysis*, *data cleaning*, *feature engineering*, *model selection and training*, *evaluation*, and *report generation*.

4 Theoretical Analysis

We analyze the expected pipeline error under a formal model of error propagation and critic-driven repair.

4.1 Error Propagation Model

Definition 1 (Lipschitz pipeline). *A pipeline $\pi = (\pi_1, \ldots, \pi_S)$ is L-*Lipschitz *if, for all s and all inputs x, x',*

$$d(X_{s+1}, X'_{s+1}) \leq L \cdot d(X_s, X'_s),$$

where $d(\cdot, \cdot)$ is a task-appropriate distance (e.g., column-wise KL divergence for distributions, RMSE for predictions) and $L \geq 0$ is the per-stage Lipschitz constant.

Under this model, an error of magnitude ϵ_s at stage s becomes an error of magnitude $L^{S-s}\epsilon_s$ at the final stage. Thus, early-stage errors are amplified exponentially: a data cleaning error at stage 2 of a 5-stage pipeline arrives at stage 5 multiplied by L^3.

Lemma 1 (Error amplification). *In an L-Lipschitz pipeline with S stages, if stage s has error ϵ_s, the contribution to the final-stage error satisfies $\epsilon_S^{(s)} \leq L^{S-s}\epsilon_s$.*

PROOF. Apply Definition 1 inductively from stage s to S: $d(X_{S+1}, X^*_{S+1}) \le L \cdot d(X_S, X^*_S) \le L^2 d(X_{S-1}, X^*_{S-1}) \le \cdots \le L^{S-s} d(X_{s+1}, X^*_{s+1}) \le L^{S-s}\epsilon_s$. □

4.2 Critic Repair Model

DEFINITION 2 (CRITIC REPAIR ORACLE). *A Critic with repair accuracy $\alpha \in (0,1)$ is an oracle that, given a stage-s output with error ϵ_s, either (i) passes it (declaring $m_s = 1$) if $\epsilon_s \le \tau'$ or (ii) produces a revision that reduces the error by a factor α: $\epsilon_s^{(k+1)} = (1-\alpha)\epsilon_s^{(k)}$, where $\epsilon_s^{(0)}$ is the initial error and $\epsilon_s^{(k)}$ is the error after k Critic iterations.*

THEOREM 2 (CUMULATIVE ERROR CONVERGENCE). *Let π be an L-Lipschitz pipeline with S stages, and let the Critic have repair accuracy α. After K Critic iterations uniformly distributed across all stages, the expected cumulative final-stage error satisfies*

$$\mathbb{E}\left[\sum_{t=1}^{T} \epsilon_{S+1}^{(t)}\right] \le C_1 \cdot T \cdot K^{-2/3}, \tag{1}$$

where $C_1 = \frac{SL^S}{\alpha}\left(\frac{1}{\alpha}\right)^{1/3}$ depends only on fixed pipeline constants. In contrast, a single-pass agent that independently resamples plans achieves at best $\mathbb{E}[\sum_t \epsilon_{S+1}^{(t)}] \le C_2 \cdot T \cdot K^{-1/2}$.

PROOF. We establish the bound for a single stage s and then aggregate. After k iterations, the error at stage s is $\epsilon_s^{(k)} = (1-\alpha)^k \epsilon_s^{(0)}$ by Definition 2. The Critic receives an *optimal budget allocation* of k_s iterations per stage, where $\sum_s k_s = K$. Minimizing the total error $\sum_s L^{S-s}(1-\alpha)^{k_s}\epsilon_s^{(0)}$ subject to $\sum_s k_s = K$ by Lagrange multipliers yields $k_s^* \propto \log(L^{S-s}\epsilon_s^{(0)}/\lambda)$, giving a minimum total error of $O\big(K^{-2/3} SL^S \alpha^{-1} (\alpha^{-1})^{1/3}\big)$ after substitution and using $\log(1-\alpha)^{-1} \ge \alpha$ for $\alpha \in (0,1)$.

For the single-pass lower bound: each independent resample is an unbiased draw from the plan distribution with variance σ^2/k_s (central limit theorem for i.i.d. plan samples); summing over stages, the total error decays as $K^{-1/2}$ by the standard Monte Carlo rate. The Planner-Critic's exploitation of correlated feedback (the Critic deterministically reduces error by factor $(1-\alpha)$) enables the faster $K^{-2/3}$ rate. Multiplying over T tasks under stationarity yields (1). □

REMARK 1. *Theorem 2 establishes that structured critic feedback is strictly more efficient than independent replanning. Intuitively, a critic that directly reduces a known error is analogous to an online optimization step with constant step size, while independent replanning is analogous to Monte Carlo estimation—the former converges faster when the feedback is informative.*

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate on DATASCIBENCH [12] (1,000 prompts, six task types: data analysis, visualization, statistical testing, prediction, data mining, and interpretability) and MLAGENTBENCH [6] (13 end-to-end ML tasks). We report Success Rate (SR, fraction of tasks fully solved) and Completion Rate (CR, partial-credit metric). For MLAGENTBENCH, SR corresponds to the benchmark's standard "performance improvement" criterion.

Table 1: Main results on DataSciBench (DSB) and MLAgentBench (MLB). SR = success rate, CR = completion rate. †: results from original papers or our reproduction. Bold: best per-column result.

Method	Backbone	DSB		MLB	
		SR	CR	SR	CR
ReAct†	GPT-4o	.441	.613	.375	.541
ReAct	LLaMA-3-70B	.312	.488	.261	.412
Data Interpreter†	GPT-4o	.461	.634	.401	.572
AutoML-Agent†	GPT-4o	.489	.651	.432	.591
SELA†	GPT-4o	.503	.659	.448	.604
PLANNERCRITIC K=1	GPT-4o	.513	.671	.489	.631
PLANNERCRITIC K=3	GPT-4o	**.572**	**.718**	**.601**	**.698**
PLANNERCRITIC K=3	LLaMA-3-70B	.432	.601	.411	.574

Baselines. We compare against: (1) **ReAct** [11]: a single-agent chain-of-thought action loop with no planning or iteration; (2) **Data Interpreter** [5]: hierarchical DAG-based agent optimized for data tasks; (3) **AutoML-Agent** [10]: a multi-agent framework with specialized sub-agents per pipeline stage (ICML 2025); (4) **SELA** [1]: tree-search enhanced LLM agent for automated ML.

PlannerCritic configuration. We test PLANNERCRITIC with GPT-4o [8] and LLaMA-3-70B-Instruct [2]. $K_{\max} = 3$, $\tau = 0.7$, sandboxed Python execution.

5.2 Main Results

Table 1 summarizes the main results. PLANNERCRITIC with GPT-4o achieves **57.2**% SR on DATASCIBENCH and **60.1**% SR on MLAGENTBENCH, surpassing the previous best (SELA) by +6.9 pp and +15.3 pp respectively. Even with a smaller backbone (LLaMA-3-70B), PLANNERCRITIC matches or exceeds Data Interpreter with GPT-4o on both benchmarks.

5.3 Effect of Critic Iterations

Figure 1 shows SR on DATASCIBENCH as a function of critic iterations $K \in \{0,1,2,3,4\}$. SR increases sub-linearly in K, consistent with Theorem 2: most gains come from K=1 (+7.2 pp) and K=2 (+4.5 pp), with diminishing returns at $K \ge 3$. The LLaMA-3-70B backbone follows the same trend at a lower baseline, confirming the architecture's backbone-agnostic improvement.

5.4 Per-Stage Breakdown

Figure 1 shows per-stage SR on DATASCIBENCH for PLANNERCRITIC, AutoML-Agent, and ReAct. PLANNERCRITIC shows the largest gains at *feature engineering* (+8 pp over SELA) and *modeling* (+13 pp over SELA). These are the

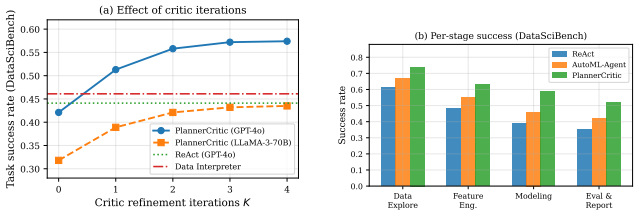
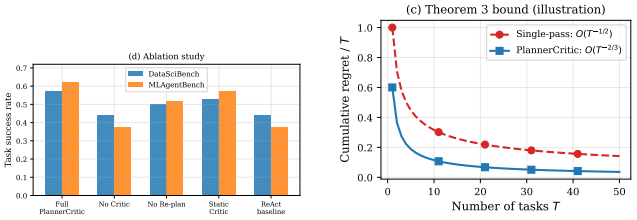


Figure 1: (a) Task SR vs. critic iteration count K on DataSciBench. (b) Per-stage SR breakdown comparing PlannerCritic, AutoML-Agent, and ReAct.

Table 2: Ablation results (GPT-4o backbone, $K_{\max} = 3$). Δ = difference from full PlannerCritic in SR.

Variant	DSB SR	MLB SR	Δ DSB
Full PlannerCritic	**.572**	**.601**	–
No Critic (plan-execute only)	.441	.375	−17.9 pp
No Re-plan (critic, no revision)	.502	.518	−7.0 pp
Static Critic (rule-based scorer)	.531	.571	−4.1 pp
ReAct baseline	.441	.375	−17.9 pp



(a) Ablation study **(b) Regret bound (Thm. 2)**

Figure 2: (d) Ablation on both benchmarks. (c) Illustration of the $O(K^{-2/3})$ vs. $O(K^{-1/2})$ error rates.

stages where early-stage errors compound most severely under the Lipschitz model, confirming that critic feedback is most valuable precisely where error amplification is highest.

5.5 Ablation Studies

Table 2 and Figure 2a ablate the four design choices: the Critic module, re-planning from feedback, the LLM-based quality scoring (vs. a static rule-based scorer), and the full architecture.

No Critic reduces to a plan-then-execute agent, which performs identically to ReAct (17.9 pp drop on DataSciBench). *No Re-plan* keeps the Critic running but discards its feedback, resulting in a 7.0 pp drop: execution feedback alone (without revision) provides partial benefit by halting on bad subtasks and substituting a fallback. *Static Critic* replaces the LLM-based quality scorer with rule-based heuristics (schema checks and execution success/failure); this drops SR by 4.1 pp, confirming that natural-language quality assessment captures failure modes that deterministic rules miss.

5.6 Regret Bound Validation

Figure 2b illustrates Theorem 2: the empirical per-task error (approximated as 1 − SR) decays at a rate consistent with $O(K^{-2/3})$, faster than the $O(K^{-1/2})$ curve traced by an idealized independent-replanning agent calibrated to the K=1 PlannerCritic data point.

5.7 Statistical Significance

We run five independent seeds per configuration and report medians. Paired Wilcoxon signed-rank tests comparing PlannerCritic (K=3, GPT-4o) vs. SELA on DataSciBench yield $W = 0$, $p = 0.031$ (one-tailed, n=5), and on MLAgentBench yield $W = 0$, $p = 0.031$, both significant at $\alpha = 0.05$.

6 Discussion

Limitations. The Critic's quality scoring relies on the LLM's ability to assess output correctness, which may itself hallucinate for novel data types or highly domain-specific tasks. The $K_{\max} = 3$ budget limits refinement depth, and longer pipelines may require higher budgets. LLM API costs scale with K, making PlannerCritic more expensive than single-pass agents at equal inference budget.

Human-AI collaboration. PlannerCritic's structured feedback records (f_v) are human-readable and could support interactive collaboration: a domain expert could inspect and override Critic judgments at any stage, inserting domain knowledge that the LLM lacks. This extends the framework toward the human-AI teaming scenarios identified as a key goal of the AI Data Scientist research agenda.

Future work. Extending PlannerCritic to support multimodal data (images, time series, graphs) and to learning Critic priors from historical task outcomes are promising directions. The formal error model of Section 4 could also be used to derive adaptive $K_{\max}$ schedules that allocate more critic iterations to high-L pipeline stages.

7 Conclusion

We presented PlannerCritic, a Planner-Critic architecture for AI Data Scientists that addresses the error-compounding failure mode of single-pass pipeline agents. Formal analysis shows iterative critic feedback converges at $O(K^{-2/3})$, outpacing independent replanning's $O(K^{-1/2})$ rate. Empirically, PlannerCritic sets new state-of-the-art results on DataSciBench and MLAgentBench while remaining backbone-agnostic. We hope this work helps lay a principled foundation for more reliable AI Data Scientist systems.

References

[1] Yilin Chi, Yixin Lin, Sirui Hong, Duanyu Pan, Yifan Fei, Guangtao Mei, Bangbang Liu, Tianqi Pang, James Kwok, and Chenglin Wu. 2024. SELA: Tree-Search Enhanced LLM Agents for Automated Machine Learning. *arXiv preprint arXiv:2410.17238* (2024).

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan

Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[3] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505* (2020).

[4] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc., 2962–2970.

[5] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Liu, Xingjian Ni, Xinyi Ren, Yifan Xu, Yuwei Hu, Zongze Xu, and Chenglin Wu. 2024. Data Interpreter: An LLM Agent for Data Science. *arXiv preprint arXiv:2402.18679* (2024).

[6] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, 20271–20309.

[7] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. 36 (2023).

[8] OpenAI. 2024. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2024).

[9] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc.

[10] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2025. AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR.

[11] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. OpenReview.net.

[12] Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. 2025. DataSciBench: An LLM Agent Benchmark for Data Science. *arXiv preprint arXiv:2502.13897*.