
R&D-AGENT: AUTOMATING DATA-DRIVEN AI SOLUTION BUILDING THROUGH LLM-POWERED AUTOMATED RESEARCH, DEVELOPMENT, AND EVOLUTION

Xu Yang^{*,†}, Xiao Yang^{*,‡}, Shikai Fang[‡], Bowen Xian[‡], Yuante Li[‡], Jian Wang[‡], Minrui Xu[‡], Haoran Pan[‡], Xinpeng Hong[‡], Weiqing Liu^{†,‡}, Yelong Shen[§], Weizhu Chen[§], Jiang Bian[‡]

[†]Microsoft Research Asia

[§]Microsoft GenAI

{xuyang1, xiaoyang, fangshikai, v-bxian, v-yuantedi, v-jianwan, v-xuminrui, v-haoranpan, v-xhong, Weiqing.Liu, Yelong.Shen, wzchen, Jiang.Bian}@microsoft.com

*Equal contribution. †Corresponding author.

ABSTRACT

Recent advances in AI and ML have transformed data science, yet increasing complexity and expertise requirements continue to hinder progress. While crowd-sourcing platforms alleviate some challenges, high-level data science tasks remain labor-intensive and iterative. To overcome these limitations, we introduce R&D-Agent, a dual-agent framework for iterative exploration. The Researcher agent uses performance feedback to generate ideas, while the Developer agent refines code based on error feedback. By enabling multiple parallel exploration traces that merge and enhance one another, R&D-Agent narrows the gap between automated solutions and expert-level performance. Evaluated on MLE-Bench, R&D-Agent emerges as the top-performing machine learning engineering agent, demonstrating its potential to accelerate innovation and improve precision across diverse data science applications. We have open-sourced R&D-Agent on GitHub: <https://github.com/microsoft/RD-Agent>.

1 INTRODUCTION

Over the past decade, artificial intelligence (AI) and machine learning (ML) have transformed the data science landscape, unlocking new possibilities for tackling problems across industries. From personalized e-commerce recommendations (Ko et al., 2022; Isinkaye et al., 2015) to AI-assisted healthcare diagnostics (Khanna et al., 2022; Shaheen, 2021), organizations have leveraged vast datasets and ever-improving algorithms to deliver extraordinary outcomes. Yet, as data complexity grows, so too does the demand for specialized expertise – experienced data scientists who can craft the right models, interpret nuanced patterns, and iterate effectively to reach optimal solutions.

Responding to this skills gap, crowd-sourcing platforms such as Kaggle have flourished by providing a collaborative forum where thousands of experts and enthusiasts converge. These communities highlight both the value of diverse perspectives and the limitations of current workflows; despite the collective prowess of global contributors, high-level data science problems still require extensive trial-and-error, deep domain knowledge, and significant time investments.

Recently, agents based on large language models (LLMs) (Achiam et al., 2023; Team et al., 2023; Liu et al., 2024) have markedly improved the efficiency and effectiveness of a wide range of tasks (Huang et al., 2023; Li et al., 2022; Liu et al., 2023; Wang et al., 2024). These agents have demonstrated remarkable capabilities in areas such as natural language processing, machine translation, and complex problem-solving and reasoning. Several benchmarks are proposed by different teams for measuring how well AI agents perform at machine learning engineering, including MLE-bench (Chan et al., 2024), DSbench (Jing et al., 2024), DiscoveryBench (Majumder et al., 2024), InsightBench (Sahu et al., 2024), etc. According to the reported results in these studies, the state-

of-the-art (SOTA) solutions utilizing the latest LLMs still exhibit subpar performance in machine learning engineering and far below the capabilities of human experts.

Crucially, data science projects often hinge on iterative insight – the repeated cycle of exploring, testing, and refining approaches as new knowledge emerges. Even when two projects share the same overarching objective, variations in data distribution, sample size, or domain constraints can demand radically different paths towards optimal solutions. Expert data scientists use feedback from each experimental iteration – whether on feature importance, model fit, or resource limitations – to systematically adapt their methods. Because each iteration can be expensive and time-consuming, these projects place a premium on well-conceived exploration strategies rather than relying on random or brute-force attempts.

In light of these challenges, we propose that a successful machine learning engineering agent must actively learn and adapt through iterative exploration. It should synthesize domain insights, generate deeper hypotheses, and refine its approach based on partial findings rather than rely on a single solution path. With these goals in mind, this paper introduces R&D-Agent, a novel framework designed to drive more effective exploration in data-driven projects across a wide array of industries. By boosting both the efficiency of research exploration and the precision of development, R&D-Agent aspires to narrow the gap between automated intelligence and expert-level data science, accelerating innovation where it is needed most.

In particular, the design of the R&D-Agent is guided by two key principles:

1. *Dedicated R&D Role*: the framework employs two specialized agents – the “Researcher” and the “Developer” – which correspond to the two types of feedback provided in each exploration step: solution performance and execution error information. The Researcher agent processes performance feedback to drive efficient exploration (i.e., idea generation), while the Developer agent leverages execution logs to iteratively refine solution implementation (i.e., code generation). Dedicated agents, designed for specific tasks and equipped with corresponding feedback, facilitate more efficient exploration.
2. *Mutually-Enhanced Multiple Traces*: the framework supports multiple exploration traces running in parallel while facilitating advanced mutual enhancements. For instance, you can launch a new exploration trace from an existing checkpoint or merge checkpoints from different traces to generate a more powerful composite solution. These capabilities not only enable the parallel exploration of a single data science task but also enhance the performance of individual exploration traces, thereby significantly boosting overall exploration efficiency.

Notably, the framework’s design offers significant flexibility for integrating diverse research contributions. This capability not only boosts the efficiency of research exploration but also enhances development precision – all through simple API calls provided by our R&D-Agent framework. Ultimately, this synergy results in a more robust data science automation system.

2 R&D-AGENT

The framework of R&D-Agent is demonstrated in Figure 1. The system is composed of two main components: the **research agent**, the **development agent**. The research agent will propose a research idea to the development agent, and the development agent will implement the idea and test the performance of the proposed solution.

2.1 DEDICATED R&D ROLE

Assigning dedicated R&D roles helps tackle complex problems across a range of tasks, including machine learning engineering. When it comes to LLM-based agents, the same design principles apply. This approach mirrors how a team generally assigns different roles to people, such as researchers and developers, allowing us to use established research and development experience. In turn, the system can gather knowledge and insights from its practice, which match human intuition and experience, and can even inspire domain experts. These lessons can then be applied to new tasks. On the other hand, there are various LLM foundation models with different strengths. For

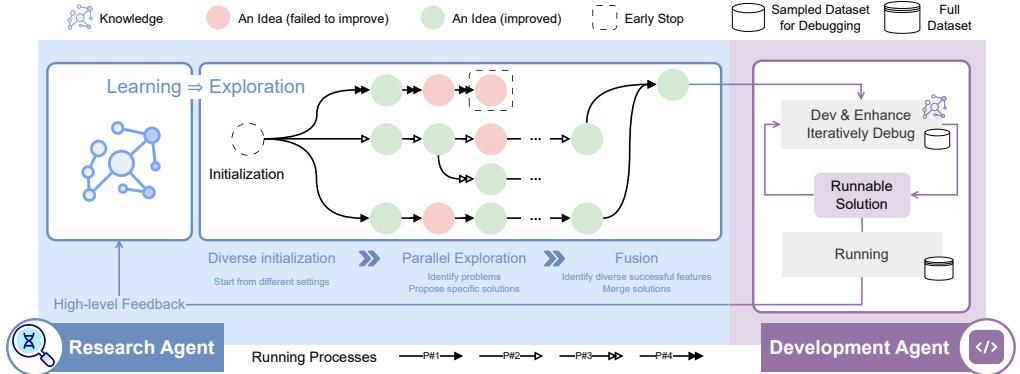


Figure 1: Framework of R&D-Agent.

example, models like o1 are very good at reasoning and coming up with creative ideas, while models like GPT-4.1 are excellent at following instructions and implementing solutions. By assigning each model that best fits its role, we can build a more effective team and achieve better results.

The research agent focuses on **learning** from experience and **exploration** of ideas. It proposes research directions to the development agent, analyzes the feedback it receives, and then refines its ideas. Through this cycle of learning and exploration, the research agent continuously improves and discovers better solutions. The learning process relies on past or external experience. As new knowledge is gained, it is collected and organized into a knowledge base. This knowledge base helps the system refine its ideas or come up with new ones. For exploration process and search strategies, we propose a novel multi-trace idea explorations, which enable parallel, diverse, and collaborative exploration of the solution space. This part will be elaborated in section 2.2.

The development agent focuses on **developing** and **enhancing** the engineering aspects of the proposed idea. The proposed idea usually only covers the key thoughts of the solution and is expressed in natural language as a high-level description that needs to be implemented. In many cases, important engineering considerations are not fully addressed (e.g., the solution must finish running within a given resource budget); the development agent develops and enhances these aspects to ensure a more complete and practical solution. To improve development efficiency, the process is divided into two phases: 1) developing a runnable solution and 2) running the solution. In the first phase, the development agent creates a runnable solution by iteratively debugging on a sampled dataset, similar to how human developers work. In the second phase, the development agent runs the solution on the full dataset to evaluate its performance. In data science, training models usually involves large datasets. By having the development agent iterate on a smaller, sampled dataset first, the process becomes much faster and more efficient. This approach allows the agent to quickly test and refine solutions before running them on the full dataset, greatly speeding up overall development.

2.2 MULTI-TRACE IDEA EXPLORATIONS

In complex data science and machine learning engineering tasks, a single linear exploration path is often insufficient for discovering high-quality solutions. R&D-Agent introduces a multi-trace exploration mechanism, which enables parallel, diverse, and collaborative exploration of the solution space. This section elaborates on the motivation behind this design and the architectural principles that underpin it.

Motivation and Design Principles: One of the fundamental challenges in automated machine learning engineering is the risk of converging on suboptimal solutions due to the limitations of a single configuration. An exploration trace is inherently constrained by its initialization — including the choice of backend LLM, prompt structure, available tools, and supporting knowledge base. These constraints can heavily bias the exploration path, leading to stagnation or premature convergence. To address this, R&D-Agent supports the parallel execution of multiple exploration traces, each configured with heterogeneous parameters. These include variations in prompt strategies, model

backends, domain-specific tools, exploration heuristics, and even knowledge scopes. This diversity increases the likelihood of uncovering valuable insights from different perspectives and avoids the narrow search trajectories imposed by uniform assumptions.

Beyond diversity, R&D-Agent is built to scale. Its multi-trace system enables both logical and physical parallelism. Each trace operates as an independent research agent, executing asynchronously across compute nodes, containers, or threads. This design allows the system to scale horizontally across distributed environments, maximizing resource utilization and dramatically reducing time-to-solution. Such parallelism is especially important in high-complexity tasks where brute-force or single-threaded search would be computationally prohibitive.

What's more, parallelism alone is not sufficient. Without coordination, multiple traces can waste resources on redundant explorations or persist with unpromising directions. To solve this, R&D-Agent introduces cross-trace collaboration protocols that govern how traces interact, evaluate progress, and make adaptive decisions. Each trace maintains a performance profile based on metrics such as solution quality, novelty, resource cost, and error resilience. A centralized module will track these profiles and makes dynamic decisions, such as terminating unproductive traces, spawning new ones with modified configurations, or initiating trace fusion. Importantly, traces are also able to share intermediate results — such as effective feature sets or partial models — creating a collective learning process where one trace's success informs others.

This principled combination of diversity, scalability, and collaboration forms the foundation of R&D-Agent's multi-trace exploration, driving efficient and robust progress in data-centric R&D.

Multi-Trace Fusion for Stronger Solutions: An essential outcome of multi-trace exploration is the ability to combine the strengths of multiple traces into a single, high-performing solution — a process we refer to as Multi-Trace Merge. Rather than selecting the best trace in isolation, R&D-Agent provides a mechanism for compositional integration of partial results from several promising traces. This strategy allows the system to capitalize on the complementary advantages discovered by each trace.

The fusion process operates at multiple granularities within the data science workflow. For instance, feature generation techniques from one trace may be combined with model architectures from another, and post-processing heuristics from a third. Each trace's components are evaluated and scored based on utility, novelty, compatibility, and performance impact. A configurable fusion strategy — such as greedy selection, weighted voting, or optimization-guided fusion — is then used to assemble the final solution.

One of R&D-Agent's key strengths is its flexible and customizable fusion design. Users can define domain-specific control and fusion rules at each stage of the process:

- During **trace evolution**, users can specify constraints for early stopping and spawning new traces based on performance thresholds, time used, or exploratory steps.
- During **information exchange**, users can determine which intermediate outputs (e.g., code fragments, error logs, metrics) are shared across traces.
- During the **fusion phase**, users can customize component compatibility rules, aggregation functions, or even plug in learned scoring models.

This flexibility ensures that R&D-Agent can adapt to a wide range of application domains and engineering preferences. Whether in finance, healthcare, or industrial AI, the system's composability and extensibility allow practitioners to shape exploration according to domain-specific requirements.

By supporting modular integration and cross-trace learning, the Multi-Trace Merge mechanism not only improves the final solution quality but also accelerates convergence and strengthens the agent's adaptability. This design is crucial in moving from isolated, trial-and-error automation toward intelligent, collective R&D exploration.

3 EXPERIMENT

3.1 EXPERIMENT SETTING

A typical scenario in applied data science is the Kaggle competitions. R&D Agent leverage its ability in MLE-Bench, which evaluates the agent’s ability to solve Kaggle challenges involving the design, building and training of the machine learning models in GPUs. We align the settings with the benchmark to give R&D Agent 24 hours with a virtual environment, GPU, the dataset, and the competition instructions targeting a solution.

The environment we provided to R&D Agent includes a 12 vCPUs, 220GB of RAM, and 1 V100 GPU with Azure OpenAI services. The targets of the experiments are summarized to:

- Evaluate the R&D ability with dedicated R&D role
- Evaluate the advantages of multi-trace in idea explorations

3.2 RESULTS

Table 1: Performance comparison on MLE-Bench. Best results in each column are in **bold**.

Agent	Low == Lite (%)	Medium (%)	High (%)	All (%)
AIDE (Jiang et al., 2025)				
o1-preview	34.3 ± 2.4	8.8 ± 1.1	10.0 ± 1.9	16.9 ± 1.1
R&D-Agent				
o1-preview	48.18 ± 2.49	8.95 ± 2.36	18.67 ± 2.98	22.4 ± 1.1
o3(R)+GPT-4.1(D)	51.52 ± 6.21	7.89 ± 3.33	16.67 ± 3.65	22.45 ± 2.45
o3(R)+GPT-4.1(D)-Multi.Trace	50.54 ± 2.51	9.86 ± 3.89	20.00 ± 8.16	24.00 ± 0.94

Table 1 summarizes the performance of different agents on the MLE-Bench, a benchmark suite categorizing Kaggle-style competitions by complexity. Each row of the table corresponds to a specific agent configuration: the rows in the part below present variants of the proposed R&D-Agent (two using o3 for research and GPT-4.1 for development, the other using only o1-preview for both roles), while the part above reports results from AIDE o1-preview (Jiang et al., 2025), the previous public best performer. The columns show the success rates (in percentage) of each agent on competitions grouped into three complexity levels—Low (Lite), Medium, and High—alongside the overall performance across all tasks. The benchmark’s complexity categories are defined by the estimated time required for an experienced ML engineer to create a basic solution: Low (Lite) for under 2 hours, Medium for 2–10 hours, and High for over 10 hours (excluding model training time). For each cell, the mean and standard deviation are reported, reflecting variability across multiple experimental runs. To ensure statistical reliability, performance numbers for R&D-Agent o1-preview are averaged over five random seeds and six seeds for R&D-Agent o3(R)+GPT-4.1(D).

In this short technical report, we present selected key results—showcasing the top-performing configurations on the latest models—to highlight the effectiveness and potential of R&D-Agent. The results show that R&D-Agent achieves much better performance than the AIDE baseline when using the same LLM backend, particularly in the Low (Lite) and High categories. This shows that the fundamental system design in R&D-Agent, which aligns more closely with the approach to solving a machine learning engineering problem, results in more robust and high-quality outcomes. In addition, we explored a hybrid strategy using backend LLMs to enable rapid, cost-effective exploration that meets real-world requirements. In comparison with other unpublished experimental results, our approach—combining o3 and GPT-4.1 by assigning o3 as the research agent (leveraging its creative ideation strengths) and deploying GPT-4.1 as the development agent (capitalizing on its superior instruction-following capabilities)—not only meets real-world requirements but also produces results that match or exceed our strongest baseline. This demonstrates the value of assigning dedicated research and development roles.

To further evaluate the effectiveness of RD-Agent’s multi-trace exploration and fusion capabilities, we designed a dedicated experiment under the MLE-Bench setting that explicitly leverages parallel exploration, information exchange, and final trace fusion.

In this setup, we used the heterogeneous configuration of o3 as the Research Agent and GPT-4.1 as the Development Agen, allowing each to specialize in their respective strengths—ideation and execution. We launched two independent exploration traces, each allowed to run for up to **11 hours** under the same task constraints. During this stage, each trace pursued distinct solution strategies, guided by diverse prompts, varying knowledge base configurations, and different toolchains. To increase diversity and reduce redundancy, we implemented an information exchange protocol: before launching the second trace, it was provided access to the exploration history and failure cases from the first trace. This ensured that the second trace could avoid repeating ineffective strategies and instead focus on novel directions.

In the **final two hours**, the agent initiated a **fusion phase**. During this phase, it merged:

- Code modules (e.g., feature engineering, model training routines),
- Ideas (e.g., hypotheses about task analysis or model design),
- Performance and error feedback from both traces.

This resulted in a composite solution that preserved the most promising elements of each exploration while resolving inconsistencies. When time allowed, the system continued to iterate on the fused solution until the 24-hour time limit was reached. Finally, **all valid solution candidates**—including those from individual traces and the fusion—were evaluated. The agent selected the final submission based on a composite scoring function, which considered validation performance, solution robustness, and overfitting risk, as derived from the score curves and model diagnostics.

In ongoing work, we are exploring additional settings to further validate and extend this strategy. These include:

- **Alternative early-stop policies**, such as terminating traces based on stagnating performance rather than fixed time budgets;
- **Domain knowledge injection** into one of the traces, simulating expert hints or historical insights to guide exploration;
- **Adaptive fusion timing**, where fusion is triggered dynamically based on trace progress rather than a fixed time split.

The ablation studies on these configurations are currently under development and will be released in a future version of this report.

4 CONCLUSION AND FUTURE WORK

In this technical report, we introduced the R&D-Agent framework for a specific scenario, machine learning engineering, and showed how it can automate the process of building data-driven AI solutions. We also shared some very preliminary results to provide an overview of our current research. R&D-Agent is a flexible framework that can support different types of solutions, and our promising early results suggest it is a valuable direction worth exploring. In addition, while machine learning engineering is just one area of research and development, we believe R&D-Agent can be used in many other scenarios as well. In future work, we plan to provide more technical details about our approach and present more comprehensive experimental results.

5 ACKNOWLEDGEMENTS

We would like to thank our colleagues Yuge Zhang, Zehua Liu, Lewen Wang, Yang Liu, and Shizhao Sun for their ongoing contributions, as well as their invaluable feedback and suggestions during our regular discussions.

REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- 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 Madry. Mle-bench: Evaluating machine learning agents on machine learning engineering. 2024. URL <https://arxiv.org/abs/2410.07095>.
- Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.
- Folasade Olubusola Isinkaye, Yetunde O Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3):261–273, 2015.
- Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixin Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. 2025. URL <https://arxiv.org/abs/2502.13138>.
- Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. Dsbench: How far are data science agents to becoming data science experts? *arXiv preprint arXiv:2409.07703*, 2024.
- Narendra N Khanna, Mahesh A Maindarker, Vijay Viswanathan, Jose Fernandes E Fernandes, Sudip Paul, Mrinalini Bhagawati, Puneet Ahluwalia, Zoltan Ruzsa, Aditya Sharma, Raghu Kolluri, et al. Economics of artificial intelligence in healthcare: diagnosis vs. treatment. In *Healthcare*, volume 10, pp. 2493. MDPI, 2022.
- Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.
- Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. Discoverybench: Towards data-driven discovery with large language models. *arXiv preprint arXiv:2407.01725*, 2024.
- Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, et al. Insightbench: Evaluating business analytics agents through multi-step insight generation. *arXiv preprint arXiv:2407.06423*, 2024.
- Mohammed Yousef Shaheen. Applications of artificial intelligence (ai) in healthcare: A review. *ScienceOpen Preprints*, 2021.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Opendedvin: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.