AUTOMIND：用于自动化数据科学的自适应知识型代理

Yixin Ou[♠♡*], **Yujie Luo**[♠♡*], **Jingsheng Zheng**[♠♡], **Lanning Wei**[♣♡], **Shuofei Qiao**[♠♡], **Jintian Zhang**[♠♡], **Da Zheng**[♣♡†], **Huajun Chen**[♠♡], **Ningyu Zhang**[♠♡†]
[♠]Zhejiang University [♣]Ant Group
[♡]Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph
{ ouyixin,zhangningyu } @zju.edu.cn zhengda.zheng@antgroup.com
https://github.com/innovatingAI/AutoMind

Abstract

大型语言模型（LLM）代理在解决实际数据科学问题方面显示出巨大潜力。LLM 驱动的数据科学代理有望自动化整个机器学习流程，但其在实际中的效果仍然有限。现有框架依赖于固定的预定义工作流和缺乏灵活性的编码策略；因此，它们仅在相对简单的经典问题上表现出色，并未能体现出人类实践者在复杂创新任务中带来的经验专业知识。在这项工作中，我们引入了 AUTOMIND，一个适应性强、知识丰富的 LLM 代理框架，通过三个关键进展克服了这些缺陷：（1）一个精选的专家知识库，使代理扎根于领域专家知识，（2）一个代理知识树搜索算法，策略性地探索可能的解决方案，以及（3）一个自适应编码策略，动态调整代码生成来适应任务复杂性。在两个自动化数据科学基准上的评估表明，AUTOMIND 相较于最先进的基线提供了优秀的表现。额外分析证实了其在效果、效率和解决方案质量上的优势，突出了 AUTOMIND 作为全自动数据科学的高效而稳健的一步。

1 介绍

数据科学代理旨在利用 LLM 代理来自动化以数据为中心的机器学习任务，这些任务从任务理解、数据探索和分析开始，通过特征工程推进，并最终在模型选择、训练和评估中达到高潮，成为未来 AI 代理实现自主科学发现的重要组成部分。许多与数据科学相关的基准 (Huang et al., 2024a; Jing et al., 2024; Chan et al., 2025; Chen et al., 2024) 已被提出，以提供基于现实世界数据科学挑战的结构化任务，以评估整个问题解决过程的性能。由于这些任务的复杂性，现有的大多数数据科学代理框架依赖于预定义的工作流程，并通过搜索和优化在特定的工作流程之上进行优化 (Jiang et al., 2025; Hong et al., 2024a; Trirat et al., 2024)，或者扩展到多代理框架，以更好地激发每个工作流节点的性能 (Li et al., 2024b)。

然而，目前的数据科学代理都忽视了模型能力的基本限制：尽管在大量基于代码的语料库上进行了训练，这些代理本质上缺乏人类实践者在数据科学任务中积累的丰富经验 (Zheng et al., 2025a)。此外，现有的数据科学代理主要采用一种不灵活的编码策略，并且倾向于在实践中仅为相对简单和经典的任务实现代码 (Guo et al., 2024; Li et al., 2024b)。然而，现实世界问题的多样性和复杂性需要一种动态的、具有上下文感知的编码策略。实际上，解决那些真正复杂甚至前沿的任务——这些任务需要高水平的创造力和创新性——对于数据科学代理在生成适合此类复杂任务的高质量代码时构成了显著挑战。

为了解决这些问题，我们提出了 AUTOMIND，一种适应性强的知识型 LLM 代理框架，用于自动化数据科学挑战。如图 1 所示，AUTOMIND 引入了三个主要创新：一个数据科学专家知识库、代理知识树搜索算法，以及一个自适应编码策略。我们在两个自动化数据科学基准测试上评估了 AUTOMIND，分别使用两种不同的基础模型系列。实验结果表明，AUTOMIND 在这两个基准测试上均表现出优于基线的性能。具体来说，在官方 MLE-Bench 排行榜上，AUTOMIND 超越了 56.8 % 的人类参与者，相比之前的最高水平（SOTA），AIDE，提高了 13.5 %。此外，我们进行了深入分析以检查 AUTOMIND 的有效性和效率，发现 AUTOMIND 在效率上提升了 300 %，并相比之前的 SOTA 减少了 63 % 的 token 成本。

2 预备知识

基于最近在将树搜索策略与 LLM 代理工作流集成方面的成功，我们将由 LLM 代理驱动的自动化数据科学建模为一个优化问题，并应用树搜索算法来解决它。

形式上，我们将某个数据科学任务的可能解记为一个元组 $s = (p, \sigma, \eta)$，其中 p 表示概述所提出方法的文字计划，σ 是对应的 Python 代码片段，而 η 是用于评估执行结果的验证指

[*] Equal Contributions.
[†] Corresponding Author.

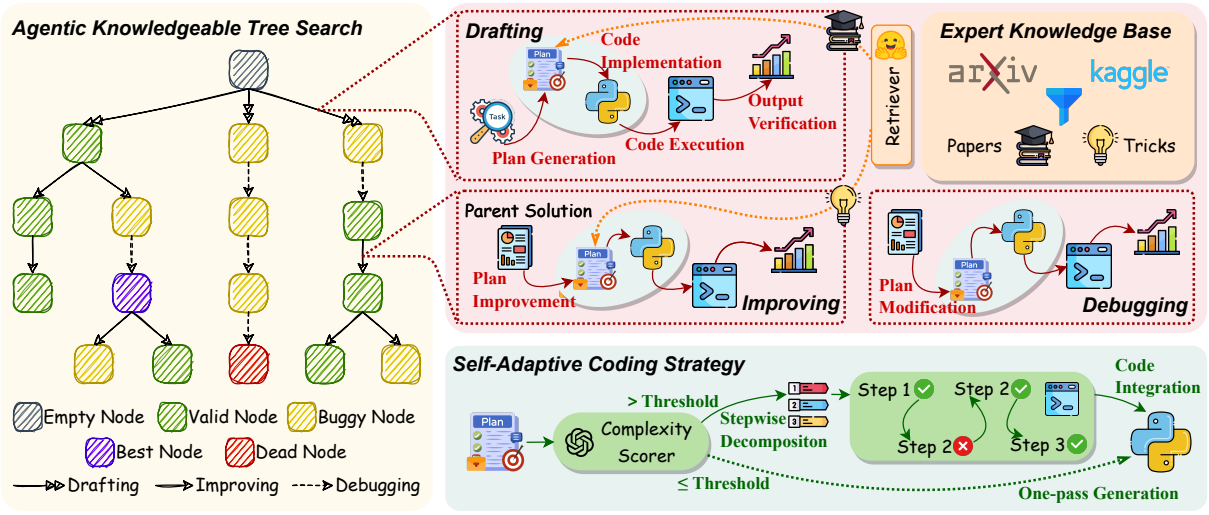

Figure 1: 我们 AUTOMIND 的框架。

标。设 $\mathcal{S}$ 为所有可能解的空间，目标是找到最优解：

$$s^* = \arg\max_{s \in \mathcal{S}} \eta \quad \text{where } \eta^* = \max\{\eta \mid (p, \sigma, \eta) \in \mathcal{S}\} \tag{1}$$

与通用代理 (Yao et al., 2023; Hong et al., 2024b; Shinn et al., 2023; Chen et al., 2023) 不同，后者将问题解决概念化为一个长视野的顺序决策过程，旨在通过基于所有先前观测的行动选择来最大化累积奖励，我们的建模方法通过直接评估和比较数据科学任务的可能解决方案显著简化了目标。

3 AUTOMIND

在本节中，我们介绍了我们的 AUTOMIND，一种旨在解决自动化数据科学挑战的自适应知识丰富的 LLM 代理框架。如图 1 所示，AUTOMIND 引入了三大创新：一个用于数据科学的专家知识库（§3.1），一个代理知识树搜索算法（§3.2），以及一个自适应编码策略（§??）。首先，代理的专业知识检索器从专家知识库中提取多个相关的论文和技巧。接下来，代理知识树搜索模块启动一个迭代循环，在其中它根据搜索策略选择一个父节点，执行一个将任务信息与检索到的知识结合以产生新解决方案的操作，并将结果节点整合到解决方案树中。同时，在每个操作的代码实现阶段，调用自适应编码策略，将解决方案的复杂性与 LLM 的固有编码能力调和。一旦达到迭代限制或时间预算耗尽，选择方程 (1) 识别的解决方案树中最优节点，提交并评估为最终解决方案。以下各节深入探讨每个组件的关键实现细节。

3.1 数据科学的专家知识

数据科学竞赛具有挑战性，因为它们要求在设计有效解决方案方面有高质量的经验 (Chan et al., 2025; Trirat et al., 2024)。仅使用 LLMs 来解决这些竞赛是有挑战性的，因为它们依赖于静态的、预训练的知识，这可能缺乏特定领域或最新的见解。为了解决这个挑战，我们构建了一个知识库，以有效地解决这些问题。它是基于特定领域的资源建立的，包括顶级会议和期刊的论文，以及从高级别竞赛解决方案中采集的专家见解。

3.1.1 知识库构建

根据人类的专业知识，我们从不同的竞赛中收集排名靠前的解决方案技巧，以期充分释放现有方法的能力，并且还收集了论文，以在解决数据科学任务时提供更具表现力的策略。在机器学习任务中，微小但有效的技巧可以显著提升模型性能。为了在我们的框架中融入这些人类智慧，我们收集了排名靠前的解决方案的讨论。具体来说，我们识别所有公开分享解决方案的 Kaggle 竞赛 [1]，然后存档竞赛描述和相关技术论坛帖子的内容。在筛选掉无效的竞赛和帖子后，收集了 455 个具有有效解决方案的 Kaggle 竞赛，包括 3,237 条包含顶级解决方案的公共论坛帖子。然后，将描述竞赛导向的技术类别的不同标签附加到每个解决方案帖子上，以便后续检索。

此外，经过同行评审后接受的论文在解决不同的机器学习任务方面具有高质量的知识。为了利用这些知识，我们首先收集最近三年内在顶级会议上发表的论文，如 KDD、ICLR、

[1] 我们使用来自 https://github.com/faridrashidi/kaggle-solutions 的列表收集了所有附有解决方案的 Kaggle 竞赛。

NeurIPS、ICML、EMNLP 以及领域特定的期刊如 Bioinformatics。对于每篇论文，保存元信息（包括标题、作者、摘要和关键词）以及主要内容，从中我们获得准备好的论文知识。

3.1.2 知识检索

直接仅使用任务描述来检索相关知识是具有挑战性的，因为现实世界的任务描述与可用的技术方法之间的关联性较弱。因此，传统仅依赖于任务描述嵌入的检索方法在我们的背景下被证明是无效的。为了解决这一限制，我们提出了一个标记系统以促进知识的检索、过滤和重新排序。

我们构建层次标签来精确描述技巧的集合。更具体来说，我们首先在所有收集的机器学习任务的基础上借助大型语言模型构建了一个层次标签集，其中包含 11 个顶级类别和相应的子类别（例如，类别是计算机视觉，子类别是图像分类）。然后，为了给每个技巧标注，AUTOMIND 首先选择最相关的顶级类别，然后从相应的子类别中识别出最合适的标签。我们通过五次独立迭代和基于投票的仲裁的自一致性策略来确保标注质量，从而为每个技巧获得精确的标签。与为比赛收集的技巧相比，论文在数据和技术上更加多样化，这给设计层次标签带来了困难。然后，我们使用大型语言模型从数据（包括类型、领域和数据集名称）、机器学习任务、提出的技术和关键贡献的角度为每篇论文生成简要总结。通过这种方式，可以从不同的角度检索论文以解决不同的比赛。

在检索阶段，输入任务使用相同的标记技巧进行分析。对于每个标签，AUTOMIND 在知识库中执行相似性搜索以检索相关知识。然后，在筛选出相同目标任务的解决方案或技巧以避免抄袭之后，检索结果基于上述标签优先顺序重新排序，我们据此获得最终检索的知识。

3.2 主动知识树搜索

解节点（$\mathcal{N}$） 为了便于探索可能的解决方案，我们将搜索空间建模为一个解树 $\mathcal{T}$，其中每个节点 $\mathcal{N} \in \mathcal{T}$ 对应一个不同的解决方案 $s = (p, \sigma, \eta)$，并被标注为有效或有缺陷。每个解节点 $\mathcal{N}$ 存储以下属性信息：

- 计划 p：端到端文本解决方案计划通常包括数据预处理、特征工程、模型训练和推断等顺序阶段。
- 代码 σ：实现所述解决方案计划的 Python 代码片段。
- 指标 η：从代码执行输出中提取的任务特定验证得分；如果执行失败，则不记录任何指标，并将该节点标记为有错误。
- 输出 o：在代码执行过程中生成的终端输出，可以用作反馈信号。
- 摘要 γ：一个由 LLM 驱动的验证器生成的简要报告，该验证器在给定计划 p、代码 σ、指标 η 和输出 o 的情况下，评估每个解决节点并将其标记为有效或存在问题。

搜索策略（π） 在每次 AUTOMIND 的迭代开始时，搜索策略 π 接收解树 $\mathcal{T}$ 的当前状态，选择一个节点作为父节点，并指定当前迭代的动作。搜索策略的算法由概率超参数和基于规则的启发式方法驱动，并在算法 1 中概述。

作用（$\mathcal{A}$） AUTOMIND 的动作空间由三种不同的操作组成：起草 $\mathcal{A}_{\text{draft}}$、改进 $\mathcal{A}_{\text{improve}}$ 和调试 $\mathcal{A}_{\text{debug}}$。如图 1 所示，每个动作都要经过类似的流程——包括计划生成、代码实现和执行，以及输出验证——但主要在计划生成阶段提供的具体输入上有所不同。在起草动作中，代理将任务描述与从专家知识库检索到的相关论文结合起来，以制定初步计划。在改进动作中，代理获得父方案——包括计划 p、代码 σ 和输出 o——以及从专家知识库中检索到的技巧，并被提示相应地改进计划。在调试动作中，代理仅接收有缺陷的父方案，并被提示修改计划以解决错误。代码实现、执行和输出验证的程序在所有动作类型中统一应用。动作完成后，所得方案 s 成为解决方案树 $\mathcal{T}$ 中的一个节点 $\mathcal{N}$，然后开始下一次迭代。

为了应对数据科学工作负载的广泛谱系，从简单的机器学习模型到多阶段的最先进架构，我们在 AUTOMIND 中引入了一种自适应编码机制，将解决方案的复杂性与 LLM 的编码能力相结合，如图 1 所示。在动作的代码实现阶段，我们基于专业评分标准，采用 LLM 作为评判者，对任务和解决方案计划的整体复杂性进行五分制评分。当此评分低于预设阈值时，表明代理认为该计划较为简单，代理会一次性实现该计划的整个代码，以最大化效率。相反，对于超过阈值的计划，代理采纳一步步的策略，通过将计划分解为连续的子步骤，并在每个子步骤中融入执行反馈。具体来说，对于每个子步骤，代理执行抽象语法树（AST）检查，然后在终端会话中执行相应的代码。如果测试通过，代理进入下一个子步骤；否则，代理使用错误信息作为反馈重新生成子步骤的实现。此循环重复进行直到所有子步骤测试成功，此时代理将子步骤的代码整合为一个完整的实现；或者是任何子步骤达到预定义的重试限制，迫使代理放弃当前计划。

Algorithm 1 在 AUTOMIND 中搜索策略 π

Require: $\mathcal{T}$ ▷ Current state of solution tree
Require: N_{init} ▷ Hyper-parameter of the number of initial draft nodes
Require: H_{debug} ▷ Hyper-parameter of the probalitity of whether debug a node
Require: H_{greedy} ▷ Hyper-parameter of the probability of whether select the best node
Ensure: ($\mathcal{N}$, $\mathcal{A}$) ▷ Select one parent node and specify the next action

$N_{\text{draft}} \leftarrow$ Get the number of draft nodes in $\mathcal{T}$
if $N_{\text{draft}} < N_{\text{init}}$ **then return** ($\mathcal{N}_{\text{empty}}$, $\mathcal{A}_{\text{draft}}$) ▷ Draft a new solution
end if
$p_{\text{debug}} \leftarrow$ Get a random floating-point number from 0 to 1
$\mathcal{N}_{\text{buggy}} \leftarrow$ Get a random buggy node in the solution tree
if $p_{\text{debug}} < H_{\text{debug}}$ and $\mathcal{N}_{\text{buggy}}$ is not None **then return** ($\mathcal{N}_{\text{buggy}}$, $\mathcal{A}_{\text{debug}}$) ▷ Debug a buggy solution
end if
$p_{\text{greedy}} \leftarrow$ Get a random floating-point number from 0 to 1
$\mathcal{N}_{\text{greedy}} \leftarrow$ Get the best node in the solution tree
$\mathcal{N}_{\text{non-greedy}} \leftarrow$ Get a random valid node in the solution tree
if $p_{\text{greedy}} < H_{\text{greedy}}$ and $\mathcal{N}_{\text{greedy}}$ is not None **then**
    return ($\mathcal{N}_{\text{greedy}}$, $\mathcal{A}_{\text{improve}}$) ▷ Improve a valid solution
else if $p_{\text{greedy}} \geq H_{\text{non-greedy}}$ and $\mathcal{N}_{\text{greedy}}$ is not None **then**
    return ($\mathcal{N}_{\text{non-greedy}}$, $\mathcal{A}_{\text{improve}}$) ▷ Mitigate getting trapped in local optima
end if
return ($\mathcal{N}_{\text{empty}}$, $\mathcal{A}_{\text{draft}}$) ▷ No solution to debug or improve, draft a new solution

4 实验

4.1 实验设置

4.1.1 基础模型和基线

基础模型 在我们的主要实验中，通过更换基础模型来评估代理。具体来说，我们评估了两个不同的模型：OpenAI 的 `o3-mini` [2] 和 DeepSeek 的 `deepseek-v3` [3] 。

基线代理 考虑到基线复现所需的大量计算资源，我们的主要实验中的定量比较仅限于 AIDE (Jiang et al., 2025)，其表现优于 MLAB (Huang et al., 2024a) 和 OpenHands (Wang et al., 2025)，并代表了 MLE-Bench (Chan et al., 2025) 上的先前最佳 (SOTA)。我们在附录 B 中提供了复现我们不同代理结果所需的详细超参数。

4.1.2 基准和指标

MLE-Bench 我们选择 MLE-Bench (Chan et al., 2025)，它包括 75 个离线 Kaggle 竞赛，用于评估 LLM 代理，作为我们测试基准的一部分。我们进一步应用基于规则的过滤来处理 MLE-Bench 中的任务，详见附录 A 。结果，我们获得了一个简化版本的 MLE-Bench，其中包括 16 个任务 [4]，这些任务根据人类经验和之前 SOTA 的表现分为简单、中等和困难等级。我们在附录 A 中包括了用于评估的 MLE-Bench 任务的完整列表。

顶级 AI 竞赛 对原始 MLE-Bench 的检查表明，大多数任务都是在 2023 年之前策划的，其中一些经典的机器学习任务可以追溯到 2018 年或更早。考虑到基础模型在预训练期间可能已经看过相应的任务，我们补充了两项来自最近顶级 AI 竞赛的任务进行评估。具体来说，我们包括了 KDD Cup 2024 上开放学术图谱 (OAG) 挑战的 WhoIsWho-IND 赛道，评估指标是 ROC 曲线下面积 (AUC)，以及 NeurIPS 2024 比赛的 BELKA 挑战，评估标准为平均精度 (AP)。更多细节见附录 A 。

评价指标 对于 MLE-Bench，我们通过比较 AUTOMIND 和之前 SOTA 的方法与人类参与者在每个 Kaggle 比赛的官方排行榜上的提交结果，并在所有指定的分割中取平均值，来评估其性能。我们报告一个称为 Beats (%) 的指标，定义为 LLM 代理超越的人类参与者的比例。对于两个顶级 AI 比赛，我们采用组织者的官方任务指标，直接通过其原始得分评估代

[2] 03-迷你-2025-01-31
[3] DeepSeek-V3-0324
[4] 为了便于说明，我们仍然将这个简化版本称为 MLE-Bench。

Benchmarks	Split	Metric	Prior SOTA		AUTOMIND w/o Knowledge	AUTOMIND	
			o3-mini	deepseek-v3	deepseek-v3	o3-mini	deepseek-v3
MLE-Bench	Easy	Beats (%)	52.5 10.8	74.0 2.9	81.8 0.7 7.8	71.8 4.3 19.3	81.2 0.1 7.2
		Submissions (#)	93.7 0.2	90.7 33.0	25.8 12.4	14.6 0.2	18.2 3.3
	Medium	Beats (%)	20.3 3.0	29.0 6.4	39.2 5.3 10.2	24.9 6.1 4.6	44.3 0.4 15.3
		Submissions (#)	103.5 44.3	72.3 40.8	15.9 2.2	8.7 1.6	18.1 9.2
	Hard	Beats (%)	10.7 1.2	18.4 1.8	23.6 6.4 5.2	36.6 9.4 25.9	38.7 14.3 20.3
		Submissions (#)	102.1 86.8	36.0 7.1	14.1 6.2	18.9 7.2	6.3 0.7
	All	Beats (%)	30.0 0.2	43.3 3.0	52.1 1.0 8.8	45.4 1.7 15.4	56.8 1.4 13.5
		Submissions (#)	99.5 38.2	70.1 25.9	19.9 2.8	13.4 2.3	15.2 4.7
Top AI Competitions	OAG	AUC âĘŚ	0.56 0.03	0.52 0.03	0.50 0.00 0.02	0.55 0.08 0.01	0.58 0.04 0.06
		Submissions (#)	16.0 15.6	14.5 13.4	48.5 41.3	4.5 2.1	4.0 1.4
	BELKA	AP âĘŚ	0.09 0.06	0.33 0.03	0.19 0.03 0.14	0.44 0.02 0.35	0.39 0.05 0.06
		Submissions (#)	5.0 4.2	2.0 0.0	2.5 2.1	9.0 1.4	8.5 0.7

Table 1: MLE-Bench 和顶级 AI 竞赛的主要结果。“Beats”（% ）表示代理在官方排行榜上超过的人类参与者的百分比，并作为 MLE-Bench 的主要评估指标。“Submissions”（# ）报告代理在 24 小时内所提交的有效次数，仅作为描述性统计而非评估指标提供，因为其数量并不能直接反映代理的能力。AUC 和 AP 分别是 OAG 和 BELKA 竞赛的官方评估指标。相对于基线的性能提升记为 ，而下降记为 。

理。

4.1.3 运行时环境设置

在我们的实验中，LLM 代理被加载到一个 Ubuntu 20.04 的 Docker 容器中，该容器包含为每个任务准备的数据集，以及预安装了用于机器学习的标准 Python 包（例如，PyTorch, scikit-learn）的 Anaconda 环境，从而提供所有代码实现和执行所需的依赖项。容器运行在一个计算节点上，该节点配备 48 个 vCPU，448GB 的内存，9.6TB 的 SSD 存储，以及一块 NVIDIA GeForce RTX 3090 GPU，这些都对代理完全开放。对于 MLE-Bench 和顶级 AI 竞赛中的每项任务，LLM 代理分配了 24 小时的时钟预算来进行最终提交。我们为每个任务重复所有实验两次，以报告性能的平均值和标准误差。

4.2 主要结果

如表所示，AUTOMIND 在 MLE-Bench 和顶级 AI 竞赛中与之前的 SOTA 相比，尽管有效提交次数较少，但还是取得了更好的表现。我们发现，在 MLE-Bench 的官方排行榜上，AUTOMIND（o3-mini ）和 AUTOMIND（deepseek-v3 ）分别超越了 45.4 % 和 56.8 % 的人类参与者，比之前的 SOTA（AIDE）分别提升了 15.4 % 和 13.5 %。此外，AUTOMIND 在 MLE-Bench 的 Hard 分项中表现出显著的优越性，o3-mini 提升了 25.9 %，deepseek-v3 提升了 20.3 %，超过之前的 SOTA。值得注意的是，即便在没有专家知识库的情况下运行 AUTOMIND，仍然在 Beats（%）指标上超过之前的 SOTA 8.8 %。在顶级 AI 竞赛中的 OAG 和 BELKA 挑战赛上，AUTOMIND 的表现至少与之前的 SOTA 相当，并且在大多数情况下超越之前的 SOTA。尤其是，AUTOMIND （o3-mini ）在 BELKA 挑战赛中达到了 0.44 的平均精度，比之前的 SOTA 提高了 0.35 的绝对值。

5 分析

5.1 消融实验

为了验证我们设计的有效性，我们在 MLE-Bench 的 Medium 分割上对 AUTOMIND（deepseek-v3 ）进行消融实验，分别禁用 AUTOMIND 中的两个主要组件：专家知识库和自适应编码策略。

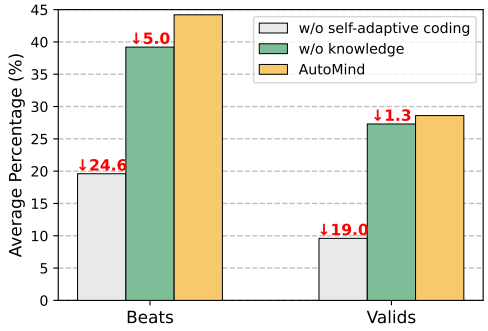

Figure 2: 对 deepseek-v3 进行的消融研究，针对 MLE-Bench 的 Medium 分组的 AUTOMIND 。Beats 表示该代理在官方排行榜上超越的人类参与者的百分比。Valids 表示代理在 24 小时时间预算内所提交的所有解决方案中有效提交的百分比。

专家知识为自我驱动的树搜索提供了额外有效的监督。 当在没有专家知识库访问的情况下运行 AUTOMIND 时，代理被迫完全依赖其内部知识来起草和改进解决方案。图 2 中的结果表明，删除专家知识库导致 Beats（%）和 Valids（%）指标分别下降了 5.0 % 和 1.3 %。图 3 展示

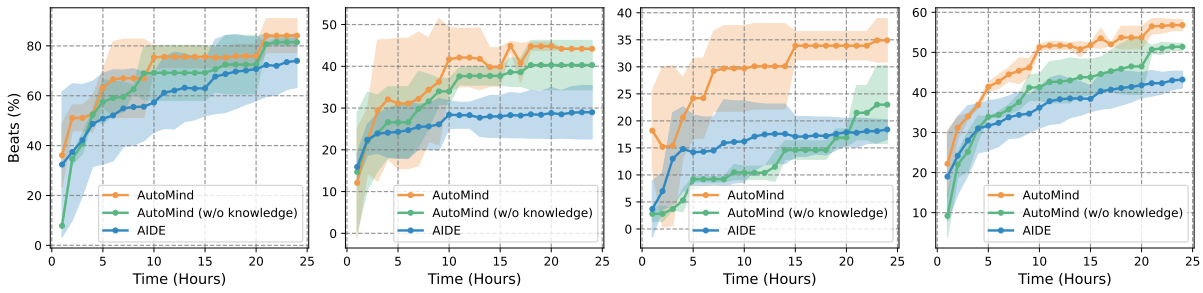

Figure 3: 测试时间扩展结果在 MLE-Bench 上。我们记录了在 deepseek-v3 实验中，代理者在 24 小时时间预算内超越人类参与者最佳解决方案的百分比的每小时快照。

了 24 小时预算内 Beats（%）指标的每小时快照，证明装备了专家知识库的 AUTOMIND 始终优于没有的变体，即使在搜索进度的最初几个小时内也是如此。我们将这些性能提升归因于专家知识的整合，这对代理的解决方案搜索空间施加了额外的约束。通过利用经过人类验证的知识， AUTOMIND 减少了对有限的内部知识的依赖，并将探索集中在更有前途的可能解决方案的搜索空间内。

自适应编码为更复杂计划的实施提供了强有力的支持。 我们通过在 AUTOMIND 的代码实现阶段完全替换为一次编码策略来消解自适应编码机制。图 2 中的结果显示，用一次策略替换自适应编码机制导致 Beats (%) 和 Valids (%) 指标分别下降了 24.6 % 和 19.0 %，突显了其在处理复杂任务和计划方面的显著局限性。我们将这种下降归因于基础 LLMs 的有限编码能力，这在一次生成中不足以应对复杂任务和计划。通过对复杂计划进行步骤分解，并结合 AST 检查和执行反馈，可以最大化地减少一次策略生成的代码早期段中的错误累积，从而保持后续代码段的高效执行。至于较简单的任务和计划，自适应编码策略本身允许使用一次生成，从而在 AUTOMIND 中在效率和鲁棒性之间取得平衡。

5.2 效率分析

测试时缩放 为了评估不同代理框架的效率，我们通过在实验中跟踪 deepseek-v3 中 AUTOMIND 和先前最先进的 AIDE 在 24 小时时间预算内的性能来研究测试时间扩展。如图 3 所示，随着可用测试时间的增加，这两个代理都能够逐步改善其解决方案。值得注意的是，在 MLE-Bench 上， AUTOMIND 平均仅需 6 小时便达到了先前最先进技术在 24 小时内的性能，相当于时间效率提高了三倍。此外，即使在没有运行专家知识库的情况下， AUTOMIND 几乎将时间效率提高了一倍，仅需 13 小时即可达到先前最先进技术的 24 小时性能。

Agents	Input	Output	Total
AIDE (24h)	2.27 ± 0.28	0.22 ± 0.03	2.49 ± 0.31
AUTOMIND(6h)	0.86 ± 0.07	0.11 ± 0.01	0.90 ± 0.04
w/o knowlege (13h)	2.06 ± 0.47	0.26 ± 0.03	2.32 ± 0.50

Table 2: 所有 MLE-Bench 任务的 Token 成本。我们展示了与 deepseek-v3 的实验中，输入、输出及总的 Token 成本，每个都以百万 Token 为单位进行量化。

代币成本 我们对每个代理框架实现之前最新技术（SOTA）的 24 小时性能时的累计令牌成本进行量化。如表 2 所示，由于效率的提高，AUTOMIND 在令牌成本上实现了 63 % 的减少，即使是没有专家知识库的变体也实现了 7 % 的令牌成本减少。

如图 4 所示，我们对 Belka 数据集进行了案例研究，以验证我们提出的 AUTOMIND 的有效性。Belka 竞赛旨在预测小分子（提供该分子的 SMILES 和三个构件）对特定蛋白质靶点（氨基酸序列可用）的结合亲和力。AUTOMIND 首先从构建的知识库中检索到 MolTrans (Huang et al., 2021) 和 DeepDTA (Öztürk et al., 2018) 论文。然后，它利用这些论文设计了一种频繁子序列挖掘策略和双重 CNN 模块。代码脚本是根据设计方案生成并执行的。相反， AUTOMIND（不使用知识）侧重于提取分子的统计特征，仅采用简单的 MLPs 来预测结合概率。对于 AIDE，最终采用了梯度提升模型，这对于解决这一复杂任务来说过于简单。与 AIDE 和 AUTOMIND（不使用知识）相比， AUTOMIND 能够检索潜在的文献，并为复杂任务设计更具表现力的模型，其更高的性能可以证明构建的知识库和检索策略的有效性。

6 相关工作

大语言模型代理 大型语言模型具有卓越的推理 (Qiao et al., 2023; Sun et al., 2025; Chen

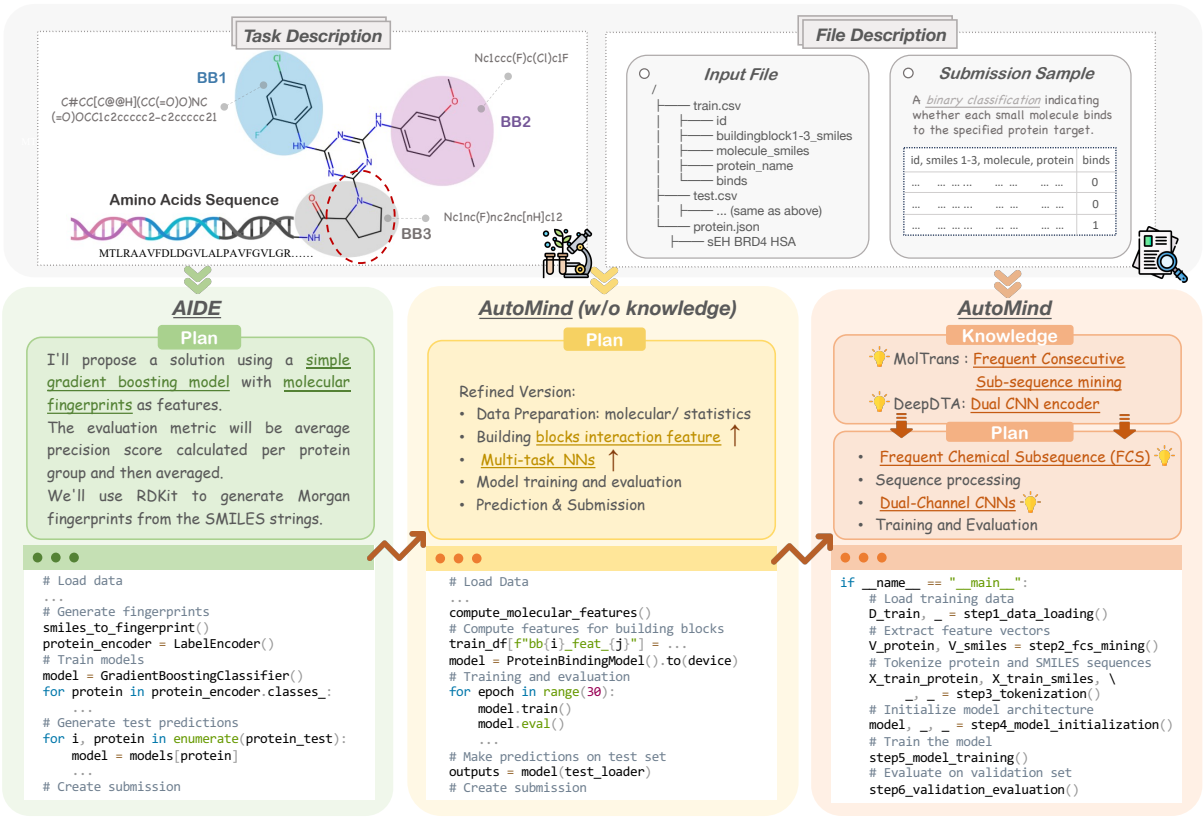

Figure 4: 在 BELKA 挑战中的一个实例。我们比较了由 AIDE 和 AUTOMIND 生成的提议解决方案计划和相应代码实现。

et al., 2025) 和规划 (Huang et al., 2024b; Wei et al., 2025a) 能力，正在成为 AI 代理的核心控制组件 (Wang et al., 2024; Xi et al., 2023; Durante et al., 2024) ,并逐渐应用于软件工程 (Qian et al., 2024; Hong et al., 2024b; Yang et al., 2024; Wei et al., 2025b) 、深度研究 (Li et al., 2025; Zheng et al., 2025b; Wu et al., 2025) 、GUI 操作 (Wu et al., 2024; Lai et al., 2024; Gou et al., 2024; Hu et al., 2024) 、科学发现 (Chen et al., 2024; Hong et al., 2024a; Trirat et al., 2024) 、体现智能 (Ahn et al., 2022; Singh et al., 2023; Song et al., 2023) 等领域。目前大多数的大型语言模型代理框架依赖于两种范式。一种是无需训练的通用架构，依靠基础模型的强大能力和精心定制的工作流程 (Hong et al., 2024b; Qian et al., 2024; Trirat et al., 2024; Li et al., 2024b) 。另一种是针对特定领域进行模型微调。先前的研究主要集中于基于大量轨迹数据的模仿学习 (Chen et al., 2023; Zeng et al., 2023; Wu et al., 2024; Qiao et al., 2024) 。然而，随着类似 GRPO 算法的出现 (Shao et al., 2024; Yu et al., 2025; Yue et al., 2025) ，模型现在可以通过基于规则的奖励 (Song et al., 2025; Jin et al., 2025; Wei et al., 2025b; Lu et al., 2025; Feng et al., 2025) 自我探索来学习完成目标任务。

用于数据科学的大型语言模型代理。 数据科学代理旨在利用 LLMs 自动化以数据为中心的机器学习任务，包括数据分析、数据建模和数据可视化，成为未来 AI 代理实现自主科学发现的重要组成部分。大多数现有方法根据人类专业知识将数据科学任务分解为具有明确界限的不同子任务，并在单个或多个代理中将其作为工作流程执行 (Zhang et al., 2023; Li et al., 2024a; Guo et al., 2024; Li et al., 2024b) 。此外，Hong et al. (2024a); Jiang et al. (2025); Trirat et al. (2024); Chi et al. (2024) 使用反思和基于搜索的优化方法。然而，这些方法忽略了模型能力的基本限制：尽管经过大规模代码数据集的训练，这些模型本质上缺乏数据科学中人类实践者积累的丰富经验。为了整合人类专业知识，DS-Agent (Guo et al., 2024) 通过收集专家 Kaggle 解决方案并应用基于案例的推理来调整这些遗留解决方案以适应新任务，从而采取了一种基于知识的方法。此外，数据科学任务类型的内在复杂性需要多样化的问题解决策略，而当前的解决方案主要在所有任务中采用统一的方法。为了弥补这些差距，本文提出通过将人类专业知识（来自研究论文、Kaggle 竞赛等）作为专家知识库，并实施动态编码策略选择机制，以适应不同的任务要求，从而增强代理能力。

我们介绍了 AUTOMIND，这是一个适应性强、知识驱动的 LLM 代理框架，专为自动化数据科学量身打造。通过整合专家知识库、基于树的推理代理和自适应编码策略，AUTOMIND 在 MLE-Bench 和两项最近的 AI 竞赛中超越了之前的最先进水平（AIDE）。

7

局限性

基准和基线 由于计算资源有限，我们没有评估全部 75 个 MLE-Bench (Chan et al., 2025) 任务集，而是选择了一个有代表性的 16 个任务的子集，这些任务是根据实验需要涵盖整个难度级别和任务类别而选择的。此外，由于 MLE-Bench 只提供整体性能指标而不提供基线代理的每个任务的原始得分，我们不得不在我们的 16 任务子集上重复评估过程，这导致了相当大的计算开销。因此，我们仅以 AIDE (Jiang et al., 2025) 作为基线进行实验，因为它在 MLE-Bench 上表现优于 MLAB (Huang et al., 2024a) 和 OpenHands (Wang et al., 2025)，并且代表了先前的最先进水平。

基础模型的编码能力 AUTOMIND 在很大程度上依赖于基础模型的编码能力。如果基础模型的编码能力不足以实现具有高潜力的复杂解决方案，我们的方法可能会比传统数据科学代理表现不佳。

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, and 24 others. 2022. Do as I can, not as I say: Grounding language in robotic affordances. *CoRR*, abs/2204.01691.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*, abs/2503.09567.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. 2024. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *CoRR*, abs/2410.05080.

Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. 2024. SELA: tree-search enhanced LLM agents for automated machine learning. *CoRR*, abs/2410.17238.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Li Fei-Fei, and Jianfeng Gao. 2024. Agent AI: surveying the horizons of multimodal interaction. *CoRR*, abs/2401.03568.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *Preprint*, arXiv:2504.11536.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for GUI agents. *CoRR*, abs/2410.05243.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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 Lu, and 6 others. 2024a. Data interpreter: An LLM agent for data science. *CoRR*, abs/2402.18679.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024b. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shawn Wang, Xinchen Xu, Shuofei Qiao, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, and 9 others. 2024. Os agents: A survey on mllm-based agents for general computing devices use. https://github.com/OS-Agent-Survey/OS-Agent-Survey/.

Kexin Huang, Cao Xiao, Lucas M Glass, and Jimeng Sun. 2021. Moltrans: molecular interaction transformer for drug–target interaction prediction. *Bioinformatics*, 37(6):830–836.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024a. Mlagentbench: Evaluating language agents on machine learning experimentation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024b. Understanding the planning of LLM agents: A survey. *CoRR*, abs/2402.02716.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. AIDE: ai-driven exploration in the space of code. *CoRR*, abs/2502.13138.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2024. Dsbench: How far are data science agents to becoming data science experts? *CoRR*, abs/2409.07703.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5295–5306. ACM.

Ruochen Li, Teerth Patel, Qingyun Wang, and Xinya Du. 2024a. Mlr-copilot: Autonomous machine learning research based on large language models agents. *CoRR*, abs/2408.14033.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025. Webthinker: Empowering large reasoning models with deep research capability. *Preprint*, arXiv:2504.21776.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. 2024b. Autokaggle: A multi-agent framework for autonomous data science competitions. *CoRR*, abs/2410.20424.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *Preprint*, arXiv:2503.21620.

Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. 2018. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15174–15186. Association for Computational Linguistics.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5368–5393. Association for Computational Linguistics.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3003–3021. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox,

Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 11523–11530. IEEE.

Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 2986–2997. IEEE.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, Yue Wu, Wenhai Wang, Junsong Chen, Zhangyue Yin, Xiaozhe Ren, Jie Fu, Junxian He, Yuan Wu, Qi Liu, and 15 others. 2025. A survey of reasoning with foundation models: Concepts, methodologies, and outlook. *ACM Comput. Surv.*

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2024. Automl-agent: A multi-agent LLM framework for full-pipeline automl. *CoRR*, abs/2410.02958.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 2 others. 2025. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025a. Plangenllms: A modern survey of LLM planning capabilities. *CoRR*, abs/2502.11221.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. 2025b. SWE-RL: advancing LLM reasoning via reinforcement learning on open software evolution. *CoRR*, abs/2502.18449.

Junde Wu, Jiayuan Zhu, and Yuyuan Liu. 2025. Agentic reasoning: Reasoning llms with tools for the deep research. *CoRR*, abs/2502.04644.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. OS-ATLAS: A foundation action model for generalist GUI agents. *CoRR*, abs/2410.23218.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *CoRR*, abs/2405.15793.

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 *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, and 8 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *Preprint*, arXiv:2504.05118.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *CoRR*, abs/2310.12823.

Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2023. Data-copilot: Bridging billions of data and humans with autonomous workflow. *CoRR*, abs/2306.07209.

Da Zheng, Lun Du, Junwei Su, Yuchen Tian, Yuqi Zhu, Jintian Zhang, Lanning Wei, Ningyu Zhang, and Huajun Chen. 2025a. Knowledge augmented complex problem solving with large language models: A survey. *Preprint*, arXiv:2505.03418.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025b. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

A 基准测试

A.1 MLE-Bench

原始的 MLE-Bench (Chan et al., 2025) 包含 75 个用于评估 LLM 代理的离线 Kaggle 比赛。我们应用基于规则的过滤来筛选 MLE-Bench 中的任务。具体而言，我们首先排除那些无法从先前 SOTA 中提交有效结果的任务，从而消除对 LLM 代理来说可能过于困难或不适当的任务。从剩下的任务中，我们抽样得到一个平衡的子集，每个任务类别（例如，图像分类，训练 LLMs）保留至少一个且不超过两个任务。因此，我们获得了一个简化版的 MLE-Bench，包含 16 个任务，根据人类经验和先前 SOTA 的表现将任务划分为简单、中等和困难级别。表格 3 中包含了用于评估的 MLE-Bench 任务的完整列表。

A.2 顶级人工智能竞赛

BELKA 数据集来自 Kaggle 比赛 [5]，该比赛预测小分子与特定蛋白质靶点的结合亲和力。它提供了 2.9B 分子-蛋白质对的训练数据。在本文中，我们从完整的训练数据中依据标签分布抽取了 2.2M 行训练数据和 590K 行测试数据，并使用 AP 分数来评估这些方法。

本文使用的 OAG 数据集是从 KDD 杯 Who-is-Who 错误分配检测任务 [6] 中收集的。给定每位作者的论文分配和每篇论文的元数据，该任务是检测每位作者的论文分配错误。在这个任务中，每篇论文包含标题、摘要、作者姓名及相应的组织、关键词、会议地点及年份。训练数据包含若干作者及相应的论文分配。为简单起见，我们从给定的论文列表中抽取了 10k 篇论文，然后保留与这些论文相关的相应作者。这些作者根据训练数据中使用的标签分布被分类为训练和测试数据。评估指标是 $\sum_1^M w_i \times AUC_i$，其中 M 是测试作者数量，

$$w_i = \frac{\#ErrorsOfThisAuthor}{\#TotalErrors}.$$

B 超参数

我们在表格 4 和表格 5 中分别列出了 AUTOMIND 和 AIDE 的详细超参数。

[5] https://www.kaggle.com/competitions/leash-BELKA

[6] https://www.biendata.xyz/kdd2024

C 提示

在本节中，我们展示了一些用于 AUTOMIND 完整流程中的提示，作为参考。

Tasks	Task Type	Dataset Size (GB)	Metric Type	Original Complexity	Split
aptos2019-blindness-detection	Image Classification	10.22	Max	Low	Easy
random-acts-of-pizza	Text Classification	0.003	Max	Low	Easy
spooky-author-identification	Text Classification	0.0019	Min	Low	Easy
google-quest-challenge	Training LLMs	0.015	Max	Medium	Easy
stanford-covid-vaccine	Tabular	2.68	Min	High	Easy
predict-volcanic-eruptions-ingv-oe	Signal Processing	31.25	Min	High	Easy
lmsys-chatbot-arena	Text Classification	0.18	Min	Medium	Medium
us-patent-phrase-to-phrase-matching	Text Regression	0.00214	Max	Medium	Medium
mlsp-2013-birds	Audio Classification	0.5851	Max	Low	Medium
statoil-iceberg-classifier-challenge	Image Classification	0.3021	Min	Medium	Medium
petfinder-pawpularity-score	Image Regression	1.04	Min	Medium	Medium
tensorflow-speech-recognition-challenge	Audio Classification	3.76	Max	Medium	Medium
denoising-dirty-documents	Image to Image	0.06	Min	Low	Hard
new-york-city-taxi-fare-prediction	Tabular	5.7	Min	Low	Hard
tgs-salt-identification-challenge	Image Segmentation	0.5	Max	Medium	Hard
ventilator-pressure-prediction	Forecasting	0.7	Min	Medium	Hard

Table 3: 我们的工作中用于评估的 MLE-Bench 中任务的完整列表。

Hyperparameter	Value
agent.retriever.model	gpt-4o-mini-2024-07-18
agent.analyzer.model	gpt-4o-mini-2024-07-18
agent.planner.model	& TARGET_MODEL
agent.coder.model	& TARGET_MODEL
agent.improver.model	& TARGET_MODEL
agent.verifier.model	gpt-4.1-mini-2025-04-14
agent.steps	500
agent.search.num_drafts	5
agent.search.max_debug_depth	20
agent.search.debug_prob	1
agent.search.trick_prob	0.8
agent.search.greedy_prob	0.8
agent.time_limit	86400
exec.timeout	32400

Table 4: AUTOMIND 的超参数。 & TARGET_MODEL 是被评估的基础模型。agent.search.num_drafts 是初始草稿节点的数量。agent.search.debug_prob 是调试一个节点的概率。agent.search.trick_prob 是使用技巧来改进一个节点的概率。agent.search.greedy_prob 是选择最佳节点的概率。

Hyperparameter	Value
agent.code.model	& TARGET_MODEL
agent.feedback.model	gpt-4o-mini-2024-07-18
agent.steps	500
agent.search.num_drafts	5
agent.search.max_debug_depth	20
agent.search.debug_prob	1
agent.time_limit	86400
exec.timeout	32400

Table 5: AIDE 的超参数。 & TARGET_MODEL 是被评估的基础模型。agent.search.num_drafts 是初始草稿节点的数量。agent.search.debug_prob 是调试一个节点的概率。

Prompt for plan generation of drafting

Introduction
You are an expert machine learning engineer attempting a task. In order to complete this task, you need to come up with an excellent and creative plan for a solution, which will be implemented by another engineer. We will now provide a description of the task.

Task description
{ task_description }

Memory
Take the Memory section into consideration when proposing the solution plan, don't propose the similar solution but keep the evaluation metric exactlty the same.
{ memory }

Knowledge
Some of the tricks that have proved useful for the same type of task are provided as follows:
{ tricks }
You should carefully consider these tricks when designing your solution.

Data Analysis
{ data_analysis }

Instructions

Response format
Your response should be a detailed outline/sketch of your proposed solution in natural language. You do not need to implement the solution but you should provide enough detail for another engineer to implement it in Python code. There should be no additional headings or text in your response. Just natural language text followed by a newline.

Installed Packages
Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Plan guideline
- DO NOT CHEAT ON EVALUATION. The solution should calculate the evaluation metric described in the task description on a hold-out validation set.
- If the evaluation metric is not provided, you should propose a reasonable evaluation metric for the task and calculate it.
- The solution should print the evaluation metric computed on the hold-out validation set at the last step of the solution.
- Try to come up with more modern and powerful methods to feature engineering and modelling and avoid using outdated methods. For example, if the task is a classification task, you should use modern transformer-based models instead of traditional models like CNN or LSTM.

- The solution should adopt appropriate methods to prevent model overfitting, such as data augmentation, early stopping, regularization, dropout, and others.
- Don't suggest to do model ensembling.
- Don't suggest to do Exploratory Data Analysis.
- Don't suggest to do hyperparameter tuning.
- The data is already prepared and available in the './input' directory. There is no need to unzip any files.
- The solution should use os.walk to get the paths of all available files in the '. /input' directory for data loading.
- If a 'sample_submission.csv' file existes, directly load it and use it as a template for the 'submission.csv' file. The solution should save predictions on the provide unlabeled test data in the 'submission.csv' file in the ./submission/ directory.

Prompt for plan generation of debugging

Introduction
You are an expert machine learning engineer attempting a task. You are provided with the plan, code and execution output of a previous solution below that had a bug and/or did not produce a submission.csv, and should improve it in order to fix the bug. For this you should first propose an reasonanle improvement and accordingly outline a detailed improved plan in natural language, which will be implemented by another engineer. We will now provide a description of the task.

Task description
{ task_description }

Previous Solution

Previous Plan
{ prev_plan }

Previous Code
{ prev_code }

Previous Execution Output
{ prev_output }

Data Analysis
{ data_analysis }

Instructions

Response format
First, provide a brief explanation of your reasoning for the proposed improvement to the previous plan (wrapped in <think></think>). Then, provide a detailed outline/sketch of your improved solution in natutal language based on the previous plan and your proposed improvement (wrapped in <plan></plan>). You do not need to implement the solution but you should provide enough detail for another engineer to implement it in Python code.

Installed Packages

Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Improve guideline
- You should pay attention to the execution output of the previous solution, and propose an improvement that will fix the bug.
- The improved plan should be derived by adapting the previous plan only based on the proposed improvement, while retaining other details of the previous plan.Don't suggest to do Exploratory Data Analysis.
- Don't suggest to do hyperparameter optimization, you should use the best hyperparameters from the previous solution.
- If a 'sample_submission.csv' file existes, directly load it and use it as a template for the 'submission.csv' file. The solution should save predictions on the provide unlabeled test data in the 'submission.csv' file in the ./submission/ directory.
- When describing your improved plan, do not use phrases like 'the same as before' or 'as in the previous plan'. Instead, fully restate all details from the previous plan that you want to retain, as subsequent implementation will not have access to the previous plan.

Prompt for plan generation of improving with tricks

Introduction
You are an expert machine learning engineer attempting a task. You are provided with the plan, code and execution output of a previous solution below and should improve it in order to further increase the test time performance. For this you should integrate integrate several useful tricks provided and accordingly outline a detailed improved plan in natural language, which will be implemented by another engineer. We will now provide a description of the task.

Task description
{ task_description }

Memory
Take the Memory section into consideration when proposing the solution plan, don't propose the similar solution but keep the evaluation metric exactlty the same.
{ memory }

Previous Solution

Previous Plan
{ prev_plan }

Previous Code
{ prev_code }

Previous Execution Output

{ prev_output }

Knowledge
Here are some tricks that have proved useful for the task:
{ tricks }
You should carefully consider these tricks when designing your solution.

Data Analysis
{ data_analysis }

Instructions

Response format
First, provide a brief explanation of your reasoning for the proposed improvement to the previous plan (wrapped in <think></think>). Then, provide a detailed outline/sketch of your improved solution in natutal language based on the previous plan and your proposed improvement (wrapped in <plan></plan>). You do not need to implement the solution but you should provide enough detail for another engineer to implement it in Python code.

Installed Packages
Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Improve guideline
- You should focus ONLY on integrating the provided tricks in the knowledge section into the previous solution to fully leverage their potentials.
- Make sure to fully integrate these tricks into your plan while preserving as much details as possible.
- Ensure that your plan clearly demonstrates the functions and specifics of the tricks.
- Identify the key areas in the previous solution where the knowledge can be applied.
- Suggest specific changes or additions to the code or plan based on the knowledge provided, and avoid unnecessary modifications irrelevant to the tricks.
- If a 'sample_submission.csv' file existes, directly load it and use it as a template for the 'submission.csv' file. The solution should save predictions on the provide unlabeled test data in the 'submission.csv' file in the ./submission/ directory.
- When describing your improved plan, do not use phrases like 'the same as before' or 'as in the previous plan'. Instead, fully restate all details from the previous plan that you want to retain, as subsequent implementation will not have access to the previous plan.

Prompt for plan generation of improving without tricks

Introduction
You are an expert machine learning engineer attempting a task. You are provided with the plan, code and execution output of a previous solution below and should improve it in order to further increase the test time performance. For this you should first propose a reasonable improvement

and accordingly outline a detailed improved plan in natural language, which will be implemented by another engineer. We will now provide a description of the task.

Task description
{ task_description }

Memory
Take the Memory section into consideration when proposing the solution plan, don't propose the similar solution but keep the evaluation metric exactlty the same.
{ memory }

Previous Solution

Previous Plan
{ prev_plan }

Previous Code
{ prev_code }

Previous Execution Output
{ prev_output }

Data Analysis
{ data_analysis }

Instructions

Response format
First, provide a brief explanation of your reasoning for the proposed improvement to the previous plan (wrapped in <think></think>). Then, provide a detailed outline/sketch of your improved solution in natutal language based on the previous plan and your proposed improvement (wrapped in <plan></plan>). You do not need to implement the solution but you should provide enough detail for another engineer to implement it in Python code.

Installed Packages
Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Improve guideline
- You should conduct only one expert-level actionable improvement to the previous solution.
- This improvement should be atomic so that the effect of the improved solution can be experimentally evaluated.
- The improved plan should be derived by adapting the previous plan only based on the proposed improvement, while retaining other details of the previous plan.
- Don’t suggest to do Exploratory Data Analysis.

- Don’t suggest to do hyperparameter optimization, you should use the best hyperparameters from the previous solution.
- If a 'sample_submission.csv' file existes, directly load it and use it as a template for the 'submission.csv' file. The solution should save predictions on the provide unlabeled test data in the 'submission.csv' file in the ./submission/ directory.
- When describing your improved plan, do not use phrases like 'the same as before' or 'as in the previous plan'. Instead, fully restate all details from the previous plan that you want to retain, as subsequent implementation will not have access to the previous plan.

Prompt for complexity scorer

Introduction
You are an expert machine learning engineer attempting a task. In order to complete this task, you are given a discription of the task and a solution plan proposed by another engineer and need to assess the complexity of the task and the proposed solution. We will now provide a description of the task.

Task description
{ task_description }

Proposed Solution
{ proposed_solution }

Data Analysis
{ data_analysis }

Instructions

Response format
First, provide a brief explanation of your reasoning for the assessment of the complexity of the task and the proposed solution (wrapped in <think></think>). Then, provide ONLY ONE average complexity score as floating point number between 1 and 5, which can contain 0.5 points (wrapped in <score></score>).

Task complexity scoring criteria
- 5 = Extremely complex and cutting-edge task with high levels of innovation required. This involves solving a unique or highly specialized problem that may push the boundaries of existing knowledge or technology.
- 4 = Complex task that involves advanced techniques or methodologies, requiring considerable expertise in the domain, such as building novel algorithms, optimization methods, or handling advanced data.
- 3 = Moderately complex task that requires significant problem-solving, such as integrating different methods or creating custom algorithms for specific use cases.
- 2 = Simple task with some level of complexity, such as basic algorithms that need some degree of fine-tuning or adjustment to meet the specific requirements of the project.
- 1 = Very simple task that requires minimal effort, such as basic data manipulation or applying standard algorithms without any customization.

Proposed solution complexity scoring criteria

- 5 = A groundbreaking or transformative solution that pushes the envelope in the field. It introduces a novel approach that is scalable, efficient, and offers long-term value or sets a new standard.
- 4 = A highly original and effective solution that shows a deep understanding of the problem domain and offers a significant contribution to the field. The solution is well-optimized and efficient.
- 3 = An original and creative solution with a reasonable level of complexity. It involves designing and implementing custom solutions or combining existing methods in a new way.
- 2 = A somewhat original solution that involves adapting existing tools or methods with some customization to meet the needs of the project. There may be room for optimization or improvement.
- 1 = Very basic solution, perhaps using standard, off-the-shelf tools with minimal adaptation, lacking originality or novel contributions.

Complexity scoring guideline
- Evaluate the complexity of the task and the proposed solution, and assign a score between 1 and 5.
- Assign an average score between 1 and 5, consider factors such as the task's complexity, the proposed solution, the dataset size, and the time and hardware resources required for implementation and execution.

Prompt for code implementation through one-pass coding

Introduction
You are an expert machine learning engineer attempting a task. In order to complete this task, you are given a discription of the task and a solution plan proposed by another engineer and need to assess the complexity of the task and the proposed solution. We will now provide a description of the task.

Task description
{ task_description }

Proposed Solution
{ proposed_solution }

Data Analysis
{ data_analysis }

Instructions

Response format
Your response should be a single markdown code block (wrapped in "') which implements this solution plan and prints out and save the evaluation metric.

Installed Packages
Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages

are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Code guideline
- The code should **implement the proposed solution** and **print the value of the evaluation metric computed on a hold-out validation set**,
- **AND MOST IMPORTANTLY SAVE PREDICTIONS ON THE PROVIDED UNLABELED TEST DATA IN A 'submission.csv' FILE IN THE ./submission/ DIRECTORY.**
- The code should save the evaluation metric computed on the hold-out validation set in a 'eval_metric.txt' file in the ./submission/ directory.
- The code should be a single-file python program that is self-contained and can be executed as-is.
- No parts of the code should be skipped, don't terminate the code before finishing the script.
- DO NOT WRAP THE CODE IN A MAIN FUNCTION, BUT WRAP ALL CODE in the '__main__' module, or it cannot be executed successfully.
- All class initializations and computational routines MUST BE WRAPPED in 'if __name__ == "__main__":'.
- DO NOT USE MULTIPROCESSING OR SET 'num_workers' IN DATA LOADER, as it may cause the container to crash.
- Your response should only contain a single code block.
- All input data is already prepared and available in the './input' directory. There is no need to unzip any files.
- DO NOT load data from "./data" directory, it is not available in the environment.
- Do not save any intermediate or temporary files through 'torch.save' or 'pickle.dump'.
- Try to accelerate the model training process if any GPU is available.
- DO NOT display progress bars. If you have to use function intergrated with progress bars, disable progress bars or using the appropriate parameter to silence them.
- Don't do Exploratory Data Analysis.

Prompt for stepwise decomposition

Introduction
You are an expert machine learning engineer attempting a task. In order to complete this task, you are given the proposed solution and supposed to decompose it into multiple steps. We will now provide a description of the task.

Task description
{ task_description }

Proposed Solution
{ proposed_solution }

Instructions

Response format
- Your response should be a single JSON code block (wrapped in "') which contains the decomposition steps of the proposed solution.
- The generated JSON should have the following format:
```
{
   "decomposed steps": [
      {
```

```
            "step": "Name of the step",
            "details": "Detailed description of the step",
        } ,
        ...
    ],
}
```

Solution decomposition guideline
- You should decompose the proposed solution into multiple steps, and provide detailed descriptions of each step.
- DO NOT MODIFY THE PROPOSED SOLUTION. In the description of each step, you should keep as many details of the proposed solution as possible, especially the exact hyperparameters and sample code.
- DO NOT CHEAT ON EVALUATION. The solution should calculate the evaluation metric described in the task description on a hold-out validation set.
- If the evaluation metric is not provided, you should propose a reasonable evaluation metric for the task and calculate it.
- The solution should save the evaluation metric computed on the hold-out validation set in a 'eval_metric.txt' file in the ./submission/ directory.
- The solution should use os.walk to get the paths of all available files in the '. /input' directory for data loading.
- If a sample_submission.csv file existes, directly load it and use it as a template for the 'submission.csv' file. The solution should save predictions on the provide unlabeled test data in the 'submission.csv' file in the ./submission/ directory.
- You should **print the value of the evaluation metric computed on a hold-out validation set** in the last step of the decomposed steps.
- Don't do Exploratory Data Analysis in the decomposition steps.
- If you find improvements suggestions in the plan, you should take them in serious consideration and include them in the decomposition steps.
- You do not need to implement the code in the decomposed steps.
- Note that the order of the decomposed steps determines the order in which the code is implemented and executed.

Prompt for code implementation through stepwise coding

Introduction
You are an expert machine learning engineer attempting a task. In order to complete this task, you are given the code for previous steps and need to implement the current step of a natural language solution plan proposed by another engineer in Python code. We will now provide a description of the task.

Task description
{ task_description }

Current Step
{ current_step }

Previous Steps Code
You should continue the following code for previous steps to implement the current step of the solution plan, but do not repeat it:

{ prev_steps }

Data Analysis
{ data_analysis }

Instructions

Response format
First, provide suggestions for the current step based on the previous steps and the failed last try step if provided (wrapped in <think></think>). Then, provide a single markdown code block (wrapped in "') which implements the current step of a solution plan.

Installed Packages
Your solution can use any relevant machine learning packages such as: 'torch-geometric==2.6.1', 'xgboost==2.1.3', 'torchvision==0.17.0', 'lightgbm==4.5.0', 'transformers==4.44.2', 'matplotlib==3.9.2', 'scipy==1.11.4', 'statsmodels==0.14.4', 'pandas==2.1.4', 'torch==2.2.0', 'optuna==4.0.0', 'timm==0.9.7', 'scikit-learn==1.2.2', 'numpy==1.26.2', 'bayesian-optimization==1.5.1', 'seaborn==0.13.2'. Feel free to use any other packages too (all packages are already installed!). For neural networks please use PyTorch because of the unavailability of TensorFlow in the environment.

Code guideline
- You should first provide suggestions for the current step based on the previous steps and the failed last try step if provided, and then implement the current step of the solution plan.
- **You should ONLY implement the code for the current step of the solution plan, rather than the entire solution plan.**
- DO NOT MODIFY THE CURRENT STEP. You should implement the current step exactly as it is.
- You should **print the value of the evaluation metric computed on a hold-out validation set** if it is calculated in the current step.
- You should save the evaluation metric computed on the hold-out validation set in a 'eval_metric.txt' file in the ./submission/ directory if it is calculated in the current step.
- DO NOT PRINT ANYTHING ELSE IN THE CODE, except for the evaluation metric and completion message for the current step.
- The code should be a single-file python program that is self-contained and can be executed as-is.
- DO NOT WRAP THE CODE IN A MAIN FUNCTION, BUT WRAP ALL CODE in the '__main__' module, or it cannot be executed successfully.
- All class initializations and computational routines MUST BE WRAPPED in 'if __name__ == "__main__":'.
- DO NOT USE MULTIPROCESSING OR SET 'num_workers' IN DATA LOADER, as it may cause the container to crash.
- No parts of the code should be skipped, don't terminate the code before finishing the script.
- **DO NOT REPEAT the code for previous steps, you should only import them from prev_steps.py.**
- DO NOT REPETITIVELY IMPORT THE SAME MODULES IN PREVIOUS STEPS, but you can import other modules if needed.
- **AND MOST IMPORTANTLY SAVE PREDICTIONS ON THE PROVIDED UNLABELED TEST DATA IN A 'submission.csv' FILE IN THE ./submission/ DIRECTORY.** if predictions are involved in the current step.

- All input data is already prepared and available in the './input' directory. There is no need to unzip any files.
- DO NOT load data from "./data" directory, it is not available in the environment.
- Do not save any intermediate or temporary files through 'torch.save' or 'pickle.dump'.
- You can reference to the based code to implement the current step, but do not completely repeat it.
- Try to accelerate the model training process if any GPU is available.
- DO NOT display progress bars. If you have to use function intergrated with progress bars, disable progress bars or using the appropriate parameter to silence them.
- Don't do Exploratory Data Analysis.

Prompt for output veirification

Introduction
You are an expert machine learning engineer attempting a task. You have written code to solve this task and now need to evaluate the output of the code execution. You should determine if there were any bugs as well as report the empirical findings.

Task description
{ task_description }

Code
{ code }

Execution Output
{ execution_output }

Tool
```
{
   "type": "function",
   "function": {
      "name": "submission_verify",
      "description": "Verify the execution output of the written code.",
      "parameters": {
         "type": "object",
         "properties": {
            "is_bug": {
               "type": "boolean",
               "description": "true if the output log shows that the execution failed or has some bug, otherwise false.",
            } ,            "is_overfitting": {
               "type": "boolean",
               "description": "true if the output log shows that the model is overfitting or validation metric is much worse than the training metric or validation loss is increasing, otherwise false. ",
            } ,            "has_csv_submission": {
               "type": "boolean",
               "description": "true if the code saves the predictions on the test data in a 'submission.csv' file in the './submission/' directory, otherwise false. Note that the file MUST be saved in the ./submission/ directory for this to be evaluated as true, otherwise it should be evaluated as false. You can assume the ./submission/ directory exists and is writable.",
```

```
            } ,
            "summary": {
              "type": "string",
              "description": "write a short summary (2-3 sentences) describing the empirical findings. Alternatively mention if there is a bug or the submission.csv was not properly produced. You do not need to suggest fixes or improvements.",
            } ,
            "metric": {
              "type": "number",
              "description": "If the code ran successfully, report the value of the validation metric. Otherwise, leave it null.",
            } ,
            "lower_is_better": {
              "type": "boolean",
              "description": "true if the metric should be minimized (i.e. a lower metric value is better, such as with MSE), false if the metric should be maximized (i.e. a higher metric value is better, such as with accuracy).",
            } ,
          } ,
          "required": ["is_bug", "is_overfitting", "has_csv_submission", "summary", "metric", "lower_is_better"],
        } ,
    } ,
}
```