# Towards Automated Integration of Novel ML Tools Into LLM-Driven AutoML Agents

Demonstration Track

Alexey Lapin
AI Talent Hub, ITMO University
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
a.lapin@itmo.ru

Stanislav Chumakov
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
svchumakov@itmo.ru

Pavel Marian
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
p.marian@itmo.ru

Danil Ezhov
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
doezhov@itmo.ru

Andrey Nosov
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
506029@niuitmo.ru

Igor Hromov
Sber AI Lab
Moscow, Russian Federation
hromov.igor@gmail.com

Mile Mitrovic
Sber AI Lab
Moscow, Russian Federation
milemitrovic888@gmail.com

Dmitry Simakov
Sber AI Lab
Moscow, Russian Federation
dmitryevsimakov@gmail.com

Andrey V. Savchenko
Sber AI Lab
Moscow, Russian Federation
avsavchenko@hse.ru

Nikolay O. Nikitin
AI Institute, ITMO University
Saint-Petersburg, Russian Federation
nnikitin@itmo.ru

## ABSTRACT

Large language models (LLMs) have significantly advanced automated machine learning (AutoML), enabling more exploratory workflows. However, existing LLM-based AutoML systems struggle to integrate specialized external tools, relying instead on internal model knowledge that is often outdated or incomplete for domain-specific libraries. To address this challenge, we propose AutoDS-Tools, a multi-agent framework that enables the understanding and utilization of external data science tools through graph-based retrieval over automatically generated documentation.

**Code:** https://github.com/sb-ai-lab/AutoDS-Tools

**Demonstration video:** https://youtu.be/H_88VTaxsfs

## KEYWORDS

Multi-Agent Systems; AutoML

**ACM Reference Format:**

Alexey Lapin, Stanislav Chumakov, Pavel Marian, Danil Ezhov, Andrey Nosov, Igor Hromov, Mile Mitrovic, Dmitry Simakov, Andrey V. Savchenko, and Nikolay O. Nikitin. 2026. Towards Automated Integration of Novel ML Tools Into LLM-Driven AutoML Agents: Demonstration Track. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/PPZN3366



*Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.* © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/PPZN3366

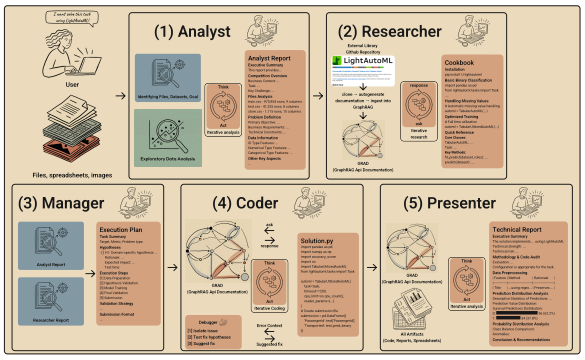


**Figure 1: Proposed AutoDS-Tools architecture**

## 1 INTRODUCTION

Automated Machine Learning (AutoML) has emerged as a critical technology for democratizing data science, enabling users without deep ML expertise to build effective models. Traditional systems such as AutoGluon [2], H2O [7], and LightAutoML [14] rely on fixed search spaces, hyperparameter tuning, and ensembling. While effective, they require substantial configuration and cannot adapt to novel problem formulations.

Large language models (LLMs) have broadened AutoML through natural-language interaction and code generation. Recent agents employ planning, iterative refinement [3], and tree-based search [5] to automate ML workflows. Systems such as AutoKaggle [9] and AIDE [5] show strong benchmark performance.

However, *integrating specialized external tools* remains a key limitation. Real-world tasks often depend on domain libraries (e.g.,

PyTorch-LifeStream [11] for event-sequence prediction and Tsururu [6] for time-series forecasting) that are underrepresented in LLM training data. As a result, agents make stale API calls, spend extra iterations on debugging, and waste context on repetitive error handling.

This challenge also appears in software engineering. Retrieval-Augmented Code Generation (RACG) methods [8] build code graphs over hierarchies, calls, and dependencies [15], treat repositories as agents [1], and mine Issues and Pull Requests [10]. Our research, unlike methods focused on repository editing, focuses on effective library utilization.

To address this gap, we introduce **AutoDS-Tools** (Fig. 1), a modular multi-agent framework for integrating external data-science tools into AutoML workflows. The demonstration video, code, and supplementary materials are available at https://github.com/AaLexUser/AutoDS-Tools.

## 2 AUTODS-TOOLS

The proposed AutoDS-Tools is a multi-agent system (MAS) designed to streamline the machine learning (ML) and data science (DS) process. It comprises 6 specialized agents, each with distinct roles:

(1) **Analyst:** Explores the dataset structure, available files, and performs exploratory analysis.
(2) **Researcher:** Studies the selected library through Graph RAG API Documentation and generates usage cookbooks with practical examples.
(3) **Manager:** Creates detailed execution plans and specifications for the Coder agent based on Analyst and Researcher reports.
(4) **Coder:** Iteratively refines the solution based on execution results, adjusts model parameters, and optimizes the data processing pipeline.
(5) **Debugger:** Catches errors and attempts to resolve them without affecting the main execution context.
(6) **Presenter:** Audits reproducibility, data handling, leakage risk, and output quality.

Each agent operates in a ReAct [16] loop, alternating between reasoning phases and acting phases. Agents exchange information through structured reports that form a cumulative knowledge chain—each subsequent agent accesses findings from all predecessors, preventing context overflow while preserving essential information flow. This creates a sequential workflow: the Analyst's report guides the Researcher, their combined insights inform the Manager's planning, and the Coder executes the final solution based on the complete accumulated context.

All agents share a common toolkit: bash execution, Jupyter notebook execution, and GRAD-based documentation querying.

To **utilize specialized external libraries** effectively, LLM agents require structured knowledge about APIs that goes beyond what is available in model training data. RAG approaches to raw documentation often fail to capture the structural relationships between code components. While GraphRAG approaches over raw source code preserve structural relationships, they introduce noise from implementation details irrelevant to library usage.

We propose **Graph RAG API Documentation (GRAD)**, a hybrid approach that combines auto-generated structured API documentation with graph-based retrieval. GRAD focuses exclusively on the public API surface and usage examples, filtering out internal implementation complexity. It operates in two phases: (1) automatic extraction of structured API documentation from source code repositories, and (2) ingestion into a GraphRAG system that enables relationship-aware querying.

**Documentation Generation.** GRAD clones the repository and performs static analysis to extract API entities (classes, methods, functions) along with their docstrings, signatures, and type hints. Usage examples are mined from multiple sources (test files, Jupyter notebooks, and documentation) using source-specific extraction strategies. Examples are deduplicated and attached to corresponding API entities.

**GraphRAG Integration.** The generated documentation is ingested into a knowledge graph built on the Cognee framework [13]. A custom prompt guides the LLM to extract entities and establish relationships such as `has_method`, `belongs_to`, and `is_used`, while allowing the model to infer additional semantic connections based on the documentation content.

**Agent Integration.** During task execution, agents access GRAD through the `libq` tool, which accepts a GitHub repository URL and a natural language query. Queries are processed using graph-completion context extension, and the system returns a synthesized answer based on the graph knowledge, a coherent response informed by API definitions and usage patterns.

## 3 DEMONSTRATION

The AutoDS-Tools demonstration presents an interactive web-based interface designed to streamline user interaction with automated data science workflows. Users populate the Graph RAG (GRAD) knowledge base by providing GitHub repository URLs. The system employs session-isolated virtual environments, enabling users to install Python libraries via the integrated Library Installer, upload datasets alongside auxiliary files, and access, preview, or download generated artifacts through the file explorer interface.

Once a dataset is uploaded, users define their task through a natural language query and specify preferred libraries. During agent execution, the interface streams real-time execution flow to users: generated code, tool execution outputs, and intermediate analysis results appear directly in the interface as they are produced.

The accompanying demonstration video illustrates these capabilities using the Spaceship Titanic [4] classification problem from Kaggle as the machine learning task, with AutoDS-Tools powered by the Qwen3-235B-A22B-Instruct model [12].

## 4 CONCLUSION

We presented AutoDS-Tools, a multi-agent framework that automates the synthesis of ML pipelines. Our multi-agent architecture, with seamless integration of specialized external libraries via the GRAD mechanism, demonstrates significant improvements over existing approaches, particularly for specialized library tasks. Designed for extensibility, AutoDS-Tools supports on-demand indexing of new public Python libraries.

## ACKNOWLEDGMENTS

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010.

## REFERENCES

[1] Linyao Chen, Zimian Peng, Yingxuan Yang, Yikun Wang, Wenzheng Tom Tang, Hiroki H. Kobayashi, and Weinan Zhang. 2025. EnvX: Agentize Everything with Agentic AI. *arXiv preprint arXiv:2509.08088* (2025).

[2] 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).

[3] Siyuan Guo, Cheng Deng, Jing Wen, Zhiyuan Chen, Zhixuan Liu, Yichao Li, Hang Li, et al. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. *arXiv preprint arXiv:2402.17453* (2024).

[4] Addison Howard, Ashley Chow, and Ryan Holbrook. 2022. Spaceship Titanic. https://kaggle.com/competitions/spaceship-titanic. Kaggle.

[5] Zhengyao Jiang et al. 2025. AIDE: AI-Driven Exploration in the Space of Code. *arXiv preprint arXiv:2502.13138* (2025).

[6] Alina Kostromina, Kseniia Kuvshinova, Aleksandr Yugay, Andrey Savchenko, and Dmitry Simakov. 2025. Tsururu: a Python-based time series forecasting strategies library. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*. 11077–11081.

[7] Erin LeDell and Sebastien Poirier. 2020. H2O AutoML: Scalable Automatic Machine Learning. In *Proceedings of the AutoML Workshop at ICML*, Vol. 2020. 24.

[8] Jia Li, Xianjie Shi, Kechi Zhang, Ge Li, Zhi Jin, Lei Li, Huangzhao Zhang, Jia Li, Fang Liu, Yuwei Zhang, Zhengwei Tao, Yihong Dong, Yuqi Zhu, and Chongyang Tao. 2025. GraphCodeAgent: Dual Graph-Guided LLM Agent for Retrieval-Augmented Repo-Level Code Generation. *arXiv preprint arXiv:2504.10046* (2025).

[9] Ziming Li et al. 2024. AutoKaggle: A Multi-Agent Framework for Autonomous Data Science Competitions. *arXiv preprint arXiv:2410.20424* (2024).

[10] Bohan Lyu, Xin Cong, Heyang Yu, Pan Yang, Yujia Qin, Yining Ye, Yaxi Lu, Zhong Zhang, Yukun Yan, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025. Enhancing Open-Domain Task-Solving Capability of LLMs via Autonomous Tool Integration from GitHub. *arXiv preprint arXiv:2312.17294* (2025). Accepted by ACL 2025 Main Conference.

[11] Artem Sakhno, Ivan Kireev, Dmitrii Babaev, Maxim Savchenko, Gleb Gusev, and Andrey Savchenko. 2025. PyTorch-Lifestream: Learning Embeddings on Discrete Event Sequences. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*. 11104–11108.

[12] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[13] Topoteretes. 2024. Cognee: Build and Manage AI Memory. https://github.com/topoteretes/cognee.

[14] Anton Vakhrushev, Alexander Ryzhkov, Maxim Savchenko, Dmitry Simakov, Rinchin Damdinov, and Alexander Tuzhilin. 2021. LightAutoML: AutoML Solution for a Large Financial Services Ecosystem. *arXiv preprint arXiv:2109.01528* (2021).

[15] Huacan Wang et al. 2025. RepoMaster: Autonomous Exploration and Understanding of GitHub Repositories for Complex Task Solving. *arXiv preprint arXiv:2505.21577* (2025).

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.