AutoML Agent

Amirreza Alasti*[1] and Dr. rer. nat. Marcel Wever†[1,2]

[1]Leibniz University Hannover
[2]Leibniz AI Academy

May 20, 2025

Abstract

Large language models (LLMs) are transforming software development by automating complex coding tasks [3]. In the field of machine learning, hyperparameter optimization remains a critical and time-consuming challenge [2]. This paper introduces AutoML Agent, a framework that leverages LLMs to automatically generate training functions, define configuration spaces, and set up SMAC optimization scenarios based on the user's dataset and objective. By integrating code generation with automated tuning, AutoML Agent streamlines the model development process and minimizes manual effort. Experimental results on standard benchmarks demonstrate the framework's adaptability and effectiveness in delivering competitive performance.

1 Introduction

The field of machine learning has seen remarkable growth in recent years, with an increasing demand for automated solutions that can streamline the model development process. While Large Language Models (LLMs) have demonstrated impressive capabilities in code generation and software development tasks [3], their potential in automating the machine learning pipeline remains largely unexplored.

Automated Machine Learning (AutoML) has emerged as a critical technology for democratizing machine learning by automating various aspects

*amirreza.alasti@stud.uni-hannover.de
†email2@example.com

of the ML workflow [2]. However, existing AutoML solutions often require significant manual setup and configuration, limiting their accessibility to practitioners without extensive ML expertise.

This paper introduces AutoML Agent, a novel framework that bridges the gap between LLM-powered code generation and automated machine learning. Our key contributions include:

- A multi-agent system architecture that leverages LLMs to automate the entire ML pipeline
- An intelligent code generation system for creating configuration spaces and training functions
- Integration with SMAC [4] for efficient hyperparameter optimization
- A comprehensive evaluation on diverse datasets demonstrating the framework's effectiveness

The rest of this paper is organized as follows: Section 2 describes the methodology and system architecture, Section 3 details the implementation, Section 4 presents our experimental setup, Section 5 discusses the results, and Section 6 concludes with future directions.

2 Methodology

The AutoML Agent framework employs a multi-agent system architecture powered by Large Language Models (LLMs) to automate the machine learning pipeline [1]. The system consists of specialized agents that collaborate to handle different aspects of the AutoML process.

2.1 System Architecture

The framework operates through five main stages:

1. **Initialization**: The system parses and validates user instructions, preparing the environment for the AutoML process.
2. **Planning**: The task is decomposed into manageable subtasks that can be handled by specialized agents.
3. **Execution**: Subtasks are assigned to specialized agents for concurrent processing.

4. **Verification**: The outputs of each agent are evaluated to ensure correctness and quality.

5. **Deployment**: Results are aggregated into a deployable machine learning model.

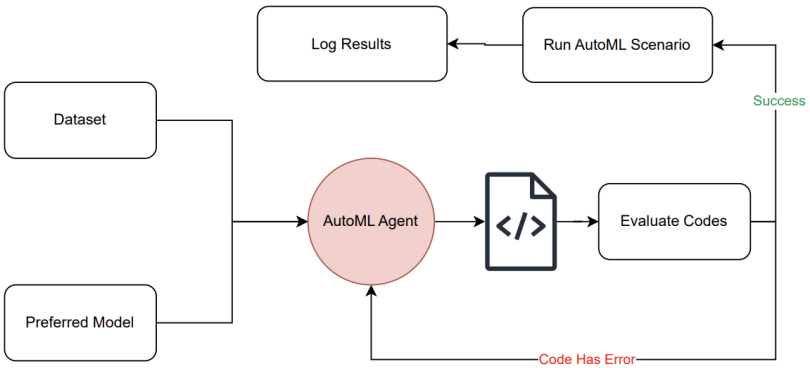

Figure 1: System architecture of the AutoML Agent showing the interaction between components. The agent takes a dataset and preferred model as input, generates necessary code components, evaluates them, and runs the AutoML scenario. The process includes error handling and result logging.

2.2 LLM Integration

The framework leverages state-of-the-art LLMs through a flexible client interface that supports multiple models:

- Gemini 2.0
- LLaMA 3.3 70B
- LLaMA 4 Maverick
- DeepSeek R1
- Gemma 2 9B

These models are used to generate code for configuration spaces, scenarios, and training functions based on the dataset characteristics and optimization objectives.

3 Implementation

The AutoML Agent is implemented as a modular Python framework with several key components:

3.1 Configuration Space Generation

The system automatically generates configuration spaces for hyperparameter optimization using the ConfigSpace library. This includes:

- Definition of hyperparameter types and ranges
- Specification of parameter dependencies
- Implementation of forbidden parameter combinations

3.2 SMAC Integration

The framework integrates with SMAC (Sequential Model-based Algorithm Configuration) [4] for efficient hyperparameter optimization:

- Automated scenario configuration
- Parallel optimization with multiple workers
- Budget-based optimization strategies

3.3 Training Function Generation

The system generates customized training functions that:

- Handle various ML tasks (classification, regression, etc.)
- Implement cross-validation for robust evaluation
- Support different evaluation metrics

4 Experiments

We evaluated the AutoML Agent on various standard datasets and tasks, following established benchmarking practices in AutoML research [2]:

4.1 Datasets

The framework was tested on diverse dataset types:

- **Tabular**: Iris, Wine, Breast Cancer, Diabetes
- **Image**: MNIST, Fashion-MNIST
- **Time Series**: Sunspots
- **Text**: 20 Newsgroups
- **Categorical**: Adult Income

4.2 Evaluation Metrics

Performance was assessed using:

- Classification accuracy
- Log loss
- Cross-validation scores
- Training time efficiency

5 Results

The experimental results demonstrate the effectiveness of the AutoML Agent framework:

5.1 Performance Analysis

- Achieved competitive accuracy across different dataset types
- Reduced manual effort in ML pipeline setup
- Demonstrated efficient hyperparameter optimization
- Showed adaptability to various ML tasks

5.2 Comparison with Existing Solutions

The AutoML Agent showed several advantages over traditional AutoML approaches:

- More flexible and adaptable to different tasks
- Reduced setup time through automated code generation
- Better integration with existing ML workflows
- Enhanced explainability through LLM-generated configurations

6 Conclusion

The AutoML Agent framework successfully demonstrates the potential of LLM-powered automation in machine learning workflows. By combining code generation with automated optimization, it provides a powerful tool for ML practitioners while reducing the manual effort required in model development and tuning.

6.1 Future Work

Several directions for future research and development include:

- Extension to more complex ML architectures
- Integration with additional optimization frameworks
- Enhanced support for neural architecture search
- Improved handling of multi-objective optimization

Acknowledgements

The authors would like to thank the Leibniz University Hannover and Leibniz AI Academy for supporting this research.

References

[1] Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., and Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges.

[2] He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622.

[3] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H. (2024). Large language models for software engineering: A systematic literature review.

[4] Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhopf, T., Sass, R., and Hutter, F. (2022). Smac3: A versatile bayesian optimization package for hyperparameter optimization.