No-Code ML Pipeline Development: Leveraging Knowledge Graphs and Language Models

Sarah Sajid[1,2][0009−0006−1557−8085], Antonis Klironomos[1,3][0009−0003−0762−1117], Evgeny Kharlamov[1][0000−0003−3247−4166], Frank Hopfgartner[2,4][0000−0003−0380−6088], and Mohamed H. Gad-Elrab[1][0000−0002−0887−3522]

[1] Bosch Center for Artificial Intelligence, Germany
[2] University of Koblenz, Germany
[3] University of Mannheim, Germany
[4] University of Sheffield, Germany
[5] {first,last}@de.bosch.com
hopfgartner@uni-koblenz.de

Abstract. Constructing machine learning (ML) pipelines is challenging for non-ML experts due to various tasks and methods. Despite several no-code tools, their ML catalogs remain difficult to navigate. To address these challenges, we present an interactive system that simplifies ML pipeline creation through a graphical user interface (GUI) powered by *ExeKGLib*, a knowledge graph (KG)-based ML framework. The GUI features a drag-and-drop interface, allowing users to design ML workflows visually without coding. In addition, a large language model (LLM)-powered assistant provides context-aware recommendations for selecting pipeline steps from the *ExeKGLib* graph. We also utilize ontologies and semantic validation to ensure logical dependencies within the pipeline, guaranteeing usability and correctness. The resulting pipelines are automatically translated into executable KGs and executed by *ExeKGLib*. We demonstrate the system's capabilities through a detailed walkthrough, highlighting its role in streamlining ML workflow creation and execution. This demo showcases the synergy between ontologies, KGs, and LLM-powered recommendations, democratizing ML pipeline development for both experts and non-experts.
Video: http://bit.ly/43TugUP

Keywords: Interactive ML Pipeline Construction · Executable Knowledge Graphs · Data Science Ontologies

1 Introduction

Machine learning (ML) is becoming a vital tool in areas like healthcare, finance, and engineering [3, 1]. However, building ML pipelines can be complex for non-ML experts, as it involves decisions on data preprocessing, model selection, hyperparameter tuning, and evaluation, often requiring knowledge of various tools and frameworks. While existing platforms such as Google's AutoML [6], KNIME

[6] https://cloud.google.com/automl

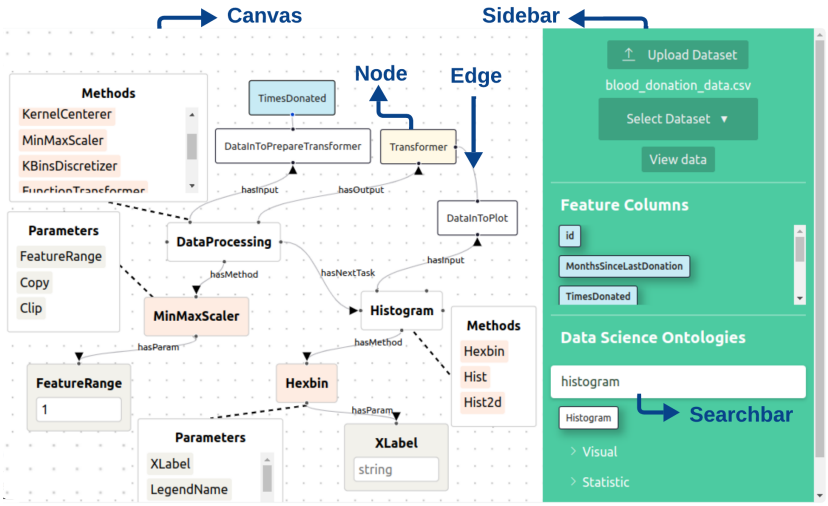

Fig. 1. *ExeKGLib*'s Graphical User Interface (GUI)

[7], and DashAI [8] provide user-friendly interfaces and automated ML capabilities, they typically operate as closed systems and do not leverage linked open data (LOD) for interoperable, reusable pipeline creation. Platforms like KNIME also offer AI-driven recommendations. However, they lack automated evaluation of suggested pipelines to assess their effectiveness.

To address these challenges, we present an interactive system with a graphical user interface (GUI) for *ExeKGLib* [2]. This system integrates a large language model (LLM)-powered assistant to simplify and enhance the flexibility of ML pipeline creation. With a drag-and-drop interface, users can build workflows visually without requiring multiple tools or complex coding, while also receiving context-aware recommendations from the assistant.

ExeKGLib's LOD-driven representation makes ML pipelines reusable and interoperable. The use of knowledge graphs (KGs) and data science ontologies provides a structured representation of ML workflows, encoding relationships between tasks, methods, and datasets for transparent and reusable workflow construction, and ensuring that the pipelines are executable [2]. This integration of visual design, AI assistant, and ontology-driven structuring lowers the technical barrier to ML pipeline development, making it accessible to both experts and non-ML practitioners.

In the following sections, we highlight the system's capabilities through a usage scenario and outline its architecture, execution flow, and key features. We will also discuss future directions for enhancing usability.

[7] https://www.knime.com/
[8] https://github.com/DashAISoftware/DashAI/

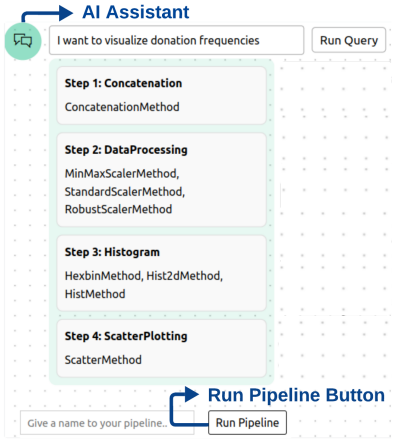

Fig. 2. Large Language Model Based AI Assistant

2 Usage Demonstration

To illustrate the functionalities of our system, we showcase how users can construct and execute ML pipelines using the GUI. The following steps outline the user interaction flow, with GUI elements illustrated in Figure 1:

- **Upload Dataset:** The user starts with uploading a dataset, and the system extracts metadata including column names and types.
- **Ask AI Assistant:** The user can get context-aware recommendations by describing the problem in the LLM-based assistant chat box illustrated in Figure 2. For example, "We need to predict possible machine failure earlier to plan maintenance".
- **Search Task:** The user can search for a specific task (*e.g.*, binary classification) by typing keywords into the search bar.
- **Task and Method Descriptions:** Hovering over tasks and methods shows brief tooltips with descriptions, aiding users in understanding their function.
- **Construct Pipeline:** The user drags and drops ML tasks supported by *ExeKGLib* (*e.g.*, feature selection, plotting) from the sidebar onto the canvas. The sidebar displays a hierarchy of tasks automatically populated from the data science ontology. For a task node on the canvas, the compatible set of methods is retrieved based on semantic relationships in the KG and shown to the user. Method parameters are populated similarly based on the ontology. The user configures the pipeline by choosing appropriate methods and their parameters and adds edges between tasks to establish connections.
- **Execute Pipeline:** When the user hits the 'Run Pipeline' button (see Figure 2), the constructed pipeline is automatically transformed into an executable KG, validated using data science ontologies, and executed within *ExeKGLib*, with real-time feedback provided to the user.

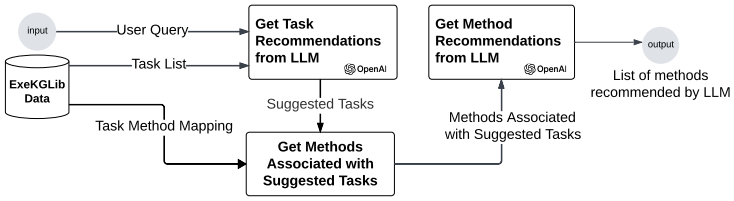

Fig. 3. LLM-Powered Recommendations Flow

3 System Design

The system consists of a front and backend, with the backend integrating *ExeKGLib* and an LLM-powered assistant.

3.1 Frontend and Backend Architecture

The frontend, built using Vue.js [9] and Vue Flow [10], provides an interactive drag-and-drop interface for constructing ML workflows while communicating with the backend for LLM-powered recommendations, semantic validation, pipeline storage and execution, and real-time feedback via the Axios API [11].

The backend processes user-defined workflows by converting them into KG representations and executing them using *ExeKGLib*. It integrates semantic validation through the data science ontology, ensuring that ML pipelines adhere to logical dependencies and best practices. The LLM assistant provides context-aware task and method recommendations based on user queries and dataset metadata. Once validated, the pipeline is executed within *ExeKGLib*, and the results are sent back to the frontend for real-time monitoring and refinement.

3.2 LLM-Powered Recommendations

The LLM-powered assistant in our system generates recommendations by analyzing user input in the context of the dataset used in pipeline construction. When a user requests a recommendation, the system dynamically constructs an input query that includes: (1) User request (*e.g.*, "Which model should I use for binary classification?"), (2) Dataset metadata (*e.g.*, column names and data types), and (3) Ontology-based constraints (*e.g.*, ensuring suggested methods are compatible with tasks).

The query is processed using OpenAI's GPT-4o LLM[12]. The workflow, shown in Figure 3, consists of two main steps: (1) *Task recommendation*, where the LLM suggests relevant ML tasks based on the processed query and the tasks supported

[9] https://vuejs.org/
[10] https://vueflow.dev/
[11] https://axios-http.com/
[12] https://openai.com/index/gpt-4/

by *ExeKGLib*, and (2) *Method recommendation*, where the LLM selects the most suitable methods for each task in the pipeline, again based on the processed query and ontology constraints. At intermediate stages, the input to the LLM is refined using classes from the ontology, ensuring that the recommended tasks and methods remain semantically relevant.

To evaluate LLM-generated results, we conducted experiments with ML problems from OpenML [13]. Using a task-specific accuracy metric, we compared the LLM's suggested pipeline steps to those implemented in OpenML. The results showed that almost 60% of the recommendations matched the benchmark.

We also conducted a usability study with 10 participants, where the LLM-powered assistant received a usefulness rating of 0.9. Participants cited its ability to suggest relevant tasks and methods as a significant advantage.

4 Future Work

We plan to enhance the GUI's usability by providing proactive next-step pipeline component suggestions and real-time error detection. Another improvement involves integrating automated pipeline summarization and explanation features, which would improve pipeline exploration and interpretability. User feedback can also be used to continuously improve pipeline recommendations.

A current limitation of the system is the use of a general-purpose language model (GPT-4o) for generating recommendations. While effective, such models are not specifically optimized for machine learning or programming tasks, which may limit the accuracy of advanced pipeline suggestions. In future work, we plan to explore the use of domain-specialized or fine-tuned LLMs to improve the precision and contextual relevance of the assistant's outputs.

Acknowledgments. The work was partially supported by EU projects Graph Massivizer (GA 101093202), SMARTY (GA 101140087), and enRichMyData (GA 101070284).

References

1. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)
2. Klironomos, A., Zhou, B., Tan, Z., Zheng, Z., Mohamed, G.E., Paulheim, H., Kharlamov, E.: Exekglib: knowledge graphs-empowered machine learning analytics. In: European Semantic Web Conference. pp. 123–127. Springer (2023)
3. Shailaja, K., Seetharamulu, B., Jabbar, M.: Machine learning in healthcare: A review. In: 2018 Second international conference on electronics, communication and aerospace technology (ICECA). pp. 910–914. IEEE (2018)

[13] https://openml.org/