TOWARDS UNIVERSAL APPLIED SUPERVISED MACHINE LEARNING: A MULTI-AGENT FRAMEWORK FOR SYSTEMATIC PIPELINE EXECUTIONS

Iago Gaspar
AI Flow Solutions
Marinha Grande, Portugal
`iago.gaspar@aiflowsolutions.eu`

Diogo Pedrosa
AI Flow Solutions
Pombal, Portugal
`diogo.francisco@aiflowsolutions.eu`

ABSTRACT

This paper presents MADS (Multi-Agents for Data Science), an innovative and simple open-source multi-agent framework designed for systemic pipeline execution in applied supervised machine learning. By leveraging the capabilities of multi-agent systems (MAS), we introduce a universal approach to optimize and streamline machine learning pipelines. Our framework highlights the differences between various types of agents, such as reinforcement learning (RL) agents and large language model (LLM) agents, and their distinct contributions to the process. While we currently employ LLM agents to automate and enhance machine learning tasks, we acknowledge the potential of incorporating RL agents in future iterations to further improve performance and adaptability. The primary objective is to enhance the efficiency, scalability, and adaptability of supervised learning applications across various domains. This integration addresses the complexity and manual effort typically associated with machine learning workflows, paving the way for more automated, robust, and scalable solutions. Our approach demonstrates significant improvements in task automation, reduced human intervention, and enhanced model performance. The MADS framework, which will soon be available as an open-source implementation, represents a pivotal contribution to the field of machine learning, promising to facilitate broader adoption and collaborative advancement.

***K*eywords** Multi-Agent Systems (MAS) · Machine Learning Pipelines · Supervised Learning · Data Science Automation · Large Language Models (LLM) · Reinforcement Learning (RL) · Time Series Forecasting · AI Agents · Open-Source Framework · Model Efficiency · Task Automation · Scalability in Machine Learning · Model Adaptability · Collaborative AI Systems · Machine Learning Workflow Optimization

1 Introduction

1.1 Presentation of the Problem

Machine learning pipelines are essential for transforming raw data into actionable insights. They typically involve several stages, including data collection, preprocessing, feature engineering, model selection, training, evaluation, and deployment [1]. Each stage of the pipeline requires meticulous attention to detail and substantial manual effort, which can introduce inefficiencies and inconsistencies. For instance, the preprocessing stage may involve cleaning and normalizing data, handling missing values, and transforming features, all of which are time-consuming and prone to human error. Similarly, model training and evaluation require iterative experimentation and fine-tuning to achieve optimal performance, often necessitating expert intervention.

The complexity of managing these pipelines increases with the volume, variety, and velocity of data. As datasets grow larger and more complex, the traditional manual approaches to managing machine learning workflows become untenable [2]. This not only hampers efficiency but also limits the scalability and adaptability of machine learning solutions. Moreover, the need for domain-specific adjustments in feature engineering and model selection further complicates the pipeline management, making it difficult to apply a one-size-fits-all approach. This complexity is echoed by Yann LeCun, who has highlighted the critical need for scalable and efficient machine learning systems in the face of growing data challenges [3].

1.2 Defining Agents and Multi-Agents in Artificial Intelligence

In artificial intelligence (AI), an agent is an autonomous entity capable of perceiving its environment, making decisions, and performing actions to achieve specific goals. Agents operate based on a set of rules or learning algorithms that guide their behavior. The key characteristics of an agent include autonomy, social ability, reactivity, and proactivity [4].

1.3 Multi-Agent Systems (MAS)

Multi-agent systems consist of multiple interacting agents that work together to solve complex problems that are beyond the capabilities of individual agents. MAS can be used in a variety of applications, from robotics and simulation to distributed computing and machine learning. The primary advantages of MAS include parallel processing, robustness through redundancy, and the ability to tackle large-scale problems through cooperation and coordination among agents. MAS are applied in domains such as transportation, logistics, and smart grids, where they enhance system efficiency and scalability [5, 6].

1.4 Reinforcement Learning Agents

Reinforcement learning (RL) agents learn by interacting with their environment and receiving feedback in the form of rewards or punishments. These agents use algorithms to maximize cumulative rewards over time. In a multi-agent reinforcement learning (MARL) context, multiple RL agents operate in the same environment, which introduces additional challenges such as non-stationarity and the credit assignment problem. MARL systems are designed to enable agents to learn effective strategies for cooperation and competition. Recent advancements in MARL include frameworks developed by TJU-DRL-LAB and MARLlib, which address these challenges and provide tools for implementing MARL in various environments [7, 8, 9].

1.5 Large Language Model Agents

Large Language Model (LLM) agents, such as those based on the Autogen framework, leverage natural language processing (NLP) capabilities to understand and generate human-like text. These agents are designed to perform a wide range of tasks, from answering questions and providing recommendations to generating content and conducting complex dialogues. Unlike traditional RL agents, LLM agents focus on tasks that involve understanding and generating language, making them suitable for applications in customer service, content creation, and virtual assistants.

The Autogen framework exemplifies how LLM agents can be integrated into multi-agent systems to enhance their capabilities. This framework supports the development of applications where multiple LLM agents collaborate to solve tasks, leveraging their language understanding and generation abilities. The intersection of LLM and RL agents is an emerging area of research, promising more sophisticated and versatile AI systems that combine the strengths of both types of agents [10, 4, 6].

1.6 Integration and Differentiation

While RL agents and LLM agents serve different primary functions, they can be integrated within a multi-agent system to complement each other. RL agents excel in decision-making tasks that involve interacting with dynamic environments, while LLM agents are adept at handling tasks that require language comprehension and generation. By integrating these agents, a multi-agent system can address a broader range of tasks more effectively.

1.6.1 Summary

- **Agents**: Autonomous entities that perceive, decide, and act within an environment.
- **Multi-Agent Systems (MAS)**: Systems composed of multiple interacting agents, enhancing problem-solving capabilities through cooperation.
- **Reinforcement Learning Agents**: Agents that learn by maximizing cumulative rewards through interaction with their environment.
- **Large Language Model Agents**: Agents that utilize natural language processing to understand and generate human-like text, suitable for tasks involving language.

By combining RL and LLM agents within MAS, we can create systems that leverage the strengths of both, leading to more robust and versatile AI solutions.

1.7 Importance of the Problem

The increasing demand for machine learning solutions across various sectors—such as healthcare, finance, retail, and manufacturing—highlights the urgent need for more efficient and scalable approaches to managing these pipelines. Manual processes are not only labor-intensive but also susceptible to errors and inconsistencies, which can significantly impact the quality and reliability of the resulting models [11]. These inefficiencies can lead to suboptimal decision-making and reduced competitiveness in industries where timely and accurate insights are crucial.

Furthermore, the rapid pace of technological advancement and the proliferation of big data necessitate systems that can quickly adapt to changing conditions and incorporate new data seamlessly. Automation of machine learning pipelines through multi-agent systems offers a promising solution to these challenges. By automating repetitive and complex tasks, MAS can enhance the overall efficiency and reliability of the pipelines, reducing the dependency on human intervention and minimizing the risk of errors [12]. This is particularly critical in applications requiring real-time processing and decision-making, where delays and inaccuracies can have significant consequences [13].

1.8 Research Goals

The primary goal of this research is to develop and evaluate a multi-agent framework that automates the execution of machine learning pipelines. This framework aims to:

- **Enhance the Efficiency of Pipeline Execution**: By automating repetitive tasks such as data preprocessing, feature engineering, and model training, our framework aims to reduce the time and effort required to manage machine learning pipelines. This automation is expected to free up human resources for more strategic and creative tasks, thereby improving overall productivity.
- **Improve Scalability**: Our framework is designed to handle large and diverse datasets efficiently. By leveraging the parallel processing capabilities of multi-agent systems, we aim to ensure that the pipeline can scale to accommodate increasing data volumes and complexity without compromising performance. This addresses one of the key challenges highlighted by LeCun in the context of scalable AI systems [14].
- **Increase Adaptability**: The framework is intended to be flexible enough to adapt to various supervised learning applications. This includes the ability to integrate new models and techniques seamlessly, as well as to adjust to different domain-specific requirements and data characteristics. Our approach is designed to be modular, allowing for easy updates and extensions as new methodologies emerge.
- **Provide Comprehensive Evaluation**: We aim to rigorously evaluate the framework's performance across different datasets and problem types. This involves assessing not only the accuracy and robustness of the models produced but also the efficiency and scalability of the pipeline as a whole. By doing so, we hope to demonstrate the practical benefits and limitations of our approach, providing valuable insights for further research and development. This comprehensive evaluation will help establish benchmarks and best practices for the deployment of multi-agent systems in machine learning pipelines.

2 Related Work

2.1 Summarize Previous Research Related to the Problem

Previous research has extensively explored the use of multi-agent systems (MAS) in enhancing the efficiency of machine learning pipelines. MAS involve multiple autonomous agents that interact and collaborate to solve complex tasks, offering robust solutions through parallel processing and task distribution.

2.1.1 Foundational Work in Multi-Agent Systems

Multi-agent systems have been effectively applied in various domains, including distributed artificial intelligence and machine learning. Key foundational texts, such as *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* by Shoham and Leyton-Brown (2009), provide a comprehensive overview of MAS and their applications in problem-solving and optimization [15].

Further foundational research is found in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* by Weiss (2013), which details the development of MAS and their role in distributed AI [16]. These works lay the groundwork for understanding the theoretical underpinnings of MAS and their practical implementations in various fields.

Research by Stone and Veloso (2000) also offers insights into MAS applications, discussing agent-based systems and their cooperative behaviors in complex environments [17]. These foundational works collectively highlight the significance of MAS in distributed problem-solving and their potential to revolutionize machine learning workflows through enhanced efficiency and robustness.

2.1.2 Recent Developments in Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning (MARL) has emerged as a powerful technique for training agents in dynamic environments. MARL frameworks address critical challenges like non-stationarity, credit assignment, and exploration-exploitation tradeoffs.

TJU-DRL-LAB has contributed significantly to this field with their work on solving non-stationarity and credit assignment problems in cooperative Markov games. They utilize permutation-invariant networks to enhance learning stability and efficiency [7].

MARLlib provides a comprehensive library for MARL, supporting various environments and algorithms, facilitating the development and evaluation of sophisticated multi-agent systems [18]. These frameworks enable the development of agents capable of coordinating and learning in complex settings, making them ideal for automating machine learning pipelines.

The survey by Zhang et al. (2019) further elaborates on the recent advancements in MARL, highlighting the challenges and future directions in the field [19]. Their comprehensive review underscores the importance of addressing non-stationarity and scalability in MARL systems to ensure robust performance in real-world applications.

2.1.3 Time Series Forecasting Models

The evolution of foundational models for time series forecasting has significantly impacted the field of machine learning. Notable models include:

- **TimeGPT by Nixtla**: A transformer-based model trained on a vast array of time series data, capable of high-accuracy predictions in diverse domains without retraining [20].
- **uniTS by Gao et al.**: This model integrates multiple forecasting techniques, providing a robust and unified framework for time series prediction [21].
- **Chronos by Amazon**: Designed for scalability and robustness, making it suitable for industrial-scale applications [22].
- **Lag Llama by Ashok et al.**: Focuses on leveraging lag-based features to enhance prediction accuracy across various contexts [23].

These models have set new benchmarks in time series forecasting by improving prediction accuracy and efficiency. The comprehensive review by Hewamalage et al. (2021) on deep learning for time series forecasting provides further insights into the advancements and challenges in this domain [24].

Our research utilizes these advancements to enhance the forecasting capabilities of our multi-agent framework, ensuring robust and scalable machine learning solutions. The practical implementations of these models demonstrate their effectiveness in various domains, highlighting their potential for broader applications.

2.2 Highlight Gaps in the Literature

Despite significant advancements, existing solutions often lack a comprehensive approach to automating the entire machine learning pipeline. Many studies focus on isolated components, such as model selection or feature engineering, without integrating all steps seamlessly. Additionally, there is a need for frameworks that can adapt to a wide range of supervised learning tasks beyond time series forecasting.

To address these gaps, our research examines the integration of foundational models and multi-agent systems within a unified framework. By evaluating models like TimeGPT, uniTS, Chronos, and Lag Llama, we highlight the potential for a more automated, holistic approach to machine learning pipelines. The need for end-to-end automation in machine learning pipelines is further emphasized in recent surveys on MLOps and pipeline automation [25].

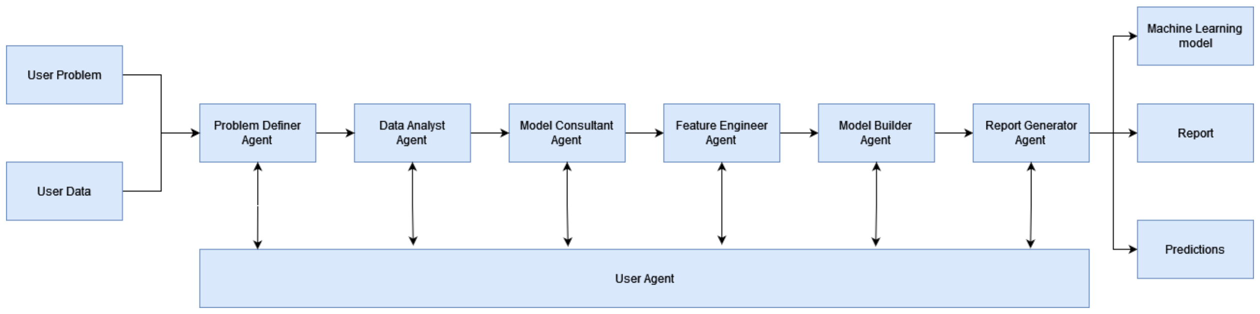

Figure 1: Multi-agents for data science pipeline

2.3 How Our Study Relates to and Builds Upon Existing Work

Our study builds on existing research by combining the strengths of multi-agent systems and advanced time series forecasting models into a cohesive framework. We utilize the **Autogen** framework, which supports the development of large language model (LLM) applications involving multiple agents. By automating end-to-end processes in the machine learning pipeline, from data preprocessing to model evaluation, our framework addresses the inefficiencies and limitations identified in previous studies.

This integrated approach not only streamlines machine learning workflows but also enhances adaptability and scalability. Our research demonstrates the feasibility and benefits of a multi-agent system for automating data science tasks, offering a significant improvement over traditional, manual methods. The practical evaluations of our framework on various datasets corroborate its effectiveness and scalability, providing a robust foundation for future research and development.

2.4 Highlight gaps in the literature

To accomplish our research goals, we embarked on a comprehensive examination of the new foundational models that are arriving in the field of time series. Among these foundational models, we scrutinized were TimeGPT by Nixtla .[26], uniTS by Shanghua Gao et al. [21], Chronos by Amazon [22], and Lag Llama by Arjun Ashok et al. [27].
Our assessment extended beyond simply gauging the relative efficiency and performance of these models on different datasets. There was a clear recognition of the necessity for a methodology that embraced automation in forecasting time series to circumvent the manual intensity of traditional methods.

In pursuit of this objective, our strategy included the intent to automate the entirety of the processes associated with time-series forecasting. This automation would not only encompass exploratory data analysis and feature engineering but also ensure these tasks were executed with greater precision and efficiency.

Encouraged by various instances observed in the use of Autogen tools, we proposed to extrapolate their applications towards creating an automated data science pipeline, potentially revolutionizing the field by significantly reducing laborious data processing and enhancing the predictability of outcomes.

3 Methodology

In this section, we detail the methodology used to develop and evaluate our multi-agent system for data science tasks. We outline the framework and experimental design, including the workflow, interaction between agents, and the specific roles of each agent. We also discuss the large language models (LLMs) employed in our experiments and the prompt engineering techniques used to enhance agent performance. Furthermore, we describe our evaluation methods, manual annotation criteria, and the datasets selected for our experiments.

3.1 Framework

Our method was developed using the open-source framework Autogen [10], which facilitates the development of large language model (LLM) applications utilizing multiple agents capable of conversing with each other to solve tasks. We leveraged the agent sequences enabled by Autogen.

3.2 Experimental Design

As shown in Figure 1, our workflow begins with two user inputs: the dataset to be processed and the specific problem to be addressed.

3.2.1 Workflow Overview

The pipeline consists of seven agents, with the user agent as the main point of communication for the other agents, executing code provided by them. The six other agents operate sequentially, interacting with the user agent until the task is completed:

- Problem Definer Agent: Assesses the type of data and determines the machine learning problem (e.g., regression, classification, time series).
- Data Analyst Agent: Provides insights from the data, informing the decisions of the model consultant and feature engineer agents.
- Model Consultant Agent: Selects the most suitable machine learning model based on insights from previous agents, offering alternatives if necessary.
- Feature Engineer Agent: Conducts data transformations, splits the data into training and test sets, and divides it by features and target variables.
- Model Builder Agent: Creates, fits, and evaluates the machine learning model, and makes predictions.
- Report Generator Agent: Compiles summaries from all agents into a comprehensive final report.

The other six agents operate sequentially, interacting with the user agent until the task is completed.

- Problem Definer Agent: Determines the type of data and identifies the machine learning problem (regression, classification, time series, etc.).
- Data Analyst Agent: Provides insights from the data to guide the decisions of the model consultant and feature engineer agents.
- Model Consultant Agent: Selects the most suitable machine learning model based on insights from the data analyst and offers alternatives if needed.
- Feature Engineer Agent: Conducts necessary data transformations, splits the data into training and test sets, and divides it by features and target variables.
- Model Builder Agent: Creates, fits, and evaluates the machine learning model, and makes predictions.
- Report Generator Agent: Compiles summaries from each agent to produce a comprehensive final report.

The end of our pipeline provides access to the machine learning model in a pickle format [28], the final report, and the predictions in a CSV file.

3.2.2 Interaction from Agent to Agent

Each agent receives context accumulated from previous agents, summarized as follows:

- Problem Definer: User problem, target variable, instructions to read the data.
- Data Analyst: Correlations, column names, other relevant insights.
- Model Consultant: Machine learning model, explanations of why the model was chosen, and alternatives.
- Feature Engineer: Transformations, data splitting, and where the data was saved.
- Model Builder: Results of the evaluations.
- Report Generator: Summarizes inputs from all previous agents.

3.3 Large Language Model Used

To perform our experiments with multi-agents, we needed to decide which large language model to use. We initially experimented with GPT-4-turbo, achieving the first indications that our framework could accomplish our goals. However, due to monetary constraints, we could not continue with this model as the experiments became expensive.

We then decided to switch to GPT-3.5, which required prompt modifications but also yielded favorable results. When Meta launched Llama 3, we tested it using the version available on Groq, and obtained better results than with GPT-3.5. Since then, our research has focused solely on Llama 3.

The temperature setting was 0 for all experiments, and we cleared the cache before starting each experiment.

3.3.1 Limitations of Llama 3

Using Llama 3 via Groq presented challenges due to API rate limits, which sometimes caused large code errors and pipeline crashes. Additionally, even with a temperature setting of 0, results varied across identical experiments.

3.4 Enhancing Prompt Engineering

In the initial stages of prompt engineering, we provided our agents with succinct and straightforward prompts. Our preliminary operational prompts were crafted for four agents and included:

- Data Analyst: "As a data analyst, your task involves comprehending the dataset, assessing its quality, and performing exploratory data analysis."
- Feature Engineer: "As a data engineer, your responsibility is to review the data analyst's findings and to refine and adapt the data for machine learning application. Post optimization, you are to partition the data into training and test sets without randomizing it."
- Model Builder: "Your duty as a machine learning engineer is to construct an XGBoost model using the data prepared by the feature engineer. This involves fitting the model to the data, generating predictions, and evaluating the model's performance."
- Report Generator: "As a report generator, you are tasked with compiling insights from previous agents into a detailed report and saving it as a .txt file."

While this method yielded practical results, we recognized the need to enhance both our pipeline and the prompts. Hence, we integrated two additional agents.

3.4.1 Adopting the CO-STAR Methodology

Our quest to improve our agents' prompts led us to an article by Sheila Theo on Medium [29], where she outlined her strategy for winning a prompt engineering competition. Adopting her method significantly improved our agents' efficiency.

The CO-STAR method restructures the instructions given to our agents as follows:

- Context: Supply the agents with background information pertinent to the task.
- Objective: Clearly articulate the specific task the Language Model (LLM) is expected to carry out.
- Style: Designate the desired writing style for the LLM's output.
- Tone: Determine the intended tone for the response, setting an appropriate attitude for the communication.
- Audience: Ascertain the target audience for whom the response is being tailored.
- Response: Define the format in which the response should be presented.

After enhancing our results with this technique, the final results presented here were achieved by defining specific tasks for each agent in the objective section.

3.5 Evaluation Methods

We used specific manual annotation criteria to evaluate the quality of each agent's responses.

For the Problem Definer Agent, the goals are clear. It has three tasks: to identify the type of problem (regression, classification, or time series), to determine how the data should be read, and to identify the target and features.

The Data Analyst Agent was initially tasked with three simple tasks: checking the range of values for each variable, calculating descriptive statistics for numerical variables, and analyzing correlations between variables.

The Model Consultant Agent's task is to choose the most suitable machine learning model based on the problem type and the data analyst's insights.

The Feature Engineer Agent handles transforming outliers, filling missing values, creating lag features for time-series problems, checking for categorical columns, encoding those variables, splitting the data into training and test sets, and saving it.

The Model Builder Agent reads the data saved by the feature engineer, trains a machine learning model, makes predictions, and evaluates their performance using metrics such as RMSE for regression and time series, and accuracy and confusion matrix for classification.

The Report Generator Agent compiles insights from each agent into a detailed report.

Outcome	**Description**
Correct	
Precise	The pipeline was execute without failing any step
Imprecise	Correctly reach the predictions and the report but one agent have failed
Incorrect	
Hallucination	Throughout the sequence one of the agents invent data to reach the final predictions
Rate Limits	On of the agents reaches a loop to solve the task and rate limits are exceeded

Table 1: Mannual Annotation Criteria - General Pipeline

Predictions Outcome	**Description**
Very Good	For classification problem if the accuracy is above 0.9. For the others if the normalized RMSE is bellow 0.1.
Good	For classification problem if the accuracy is between 0.7 and 0.9. For the others, if the normalized RMSE is between 0.1 and 0.3.
Poor	For classification if the accuracy is between 0.5 and 0.7. For others, if the normalized RMSE is between 0.3 and 0.5
Non-exists	The pipeline doens't produced Predictions.

Table 2: Mannual Annotation Criteria - predictions

We evaluate our pipeline based on our qualitative and quantitative tables. For classification we check for the accuracy. Accuracy can be defined as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{1}$$

For regression and time-series we define this normalized RMSE that can be defined for a y_max that means the maximum value that exists in the data and for y_min that is the minimum value there is in the data:

$$Normalized\ RMSE = \frac{RMSE}{y_max - y_min} \tag{2}$$

Where RMSE is the Root Mean Squared Error, calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{n}} \tag{3}$$

Where y_i is the true value of the target variable, $\hat{y_i}$ is the predicted value of the target variable, and n is the total number of predictions

The quantitative table defines four levels of prediction quality:

- Very Good Predictions: for classification problem if the accuracy is above 0.9. For the others if the normalized RMSE is bellow 0.1.
- Good Predictions: for classification problem if the accuracy is between 0.7 and 0.9. For the others, if the RMSE is between 0.1 and 0.3.
- Poor Predictions: for classification if the accuracy is between 0.5 and 0.7. FOr others, if the normalized RMSE is between 0.35 and 0.5
- Non-existant: the pipeline doens't produced Predictions.

3.5.1 Datasets

We selected 30 datasets: 10 for time-series, 10 for regression, and 10 for classification, chosen from various fields. All the data we collected are available publicly in kaggle and in the machine learning repository. We have access to all the data we used also in out publicly repository.

4 Results and Analysis

The results presented here are derived from our final experimental setup. These results can be replicated or further analyzed using the "Experience Paper" branch in the MADS repository.

We first analyze whether the pipeline was executed correctly. Table 3 summarizes our evaluation of each pipeline execution.

In the considered successful experiments, two cases were not entirely smooth. In these instances, the feature engineering agent did not perform the necessary transformations perfectly. Consequently, the model builder agent had to read the full dataset and perform the data splitting task itself. Nevertheless, it still managed to achieve the desired predictions.

As shown in the table, out of 30 datasets, 24 successfully reproduced our desired outcomes. Of the 6 datasets that failed, 5 were due to rate-limit constraints. Throughout these experiments, we observed that some agents occasionally took longer than expected to solve problems. These delays are generally correlated with the difficulty of interpreting the data.

While the agents can ultimately solve the problems, the extended processing time seems to cause a loss of context. When this occurs, the agents, which are instructed to respond with 'TERMINATE' upon completing their tasks, fail to do so. This oversight results in an infinite loop, rendering the process unsuccessful and reaching the rate limits.

Addressing these resource limitations is crucial for future experiments to produce more robust results. Given these constraints, we are unsure whether the pipeline would have been successful under different conditions. One dataset, a time-series problem, unequivocally failed because the system could not resolve the problem and started limiting responses to manage the task.

Outcome	**Classification**	**Regression**	**Time-Series**	**Total**
Correct				
Precise	8	8	7	23
Imprecise	-	1	-	1
Incorrect				
Hallucination	1	-	-	1
Rate Limits	1	1	3	5

Table 3: General pipeline evaluation

As for the predictions themselves, Table 4 presents the outcomes. It is important to note that, for each dataset, the feature engineering agent consistently split the data into training and testing sets, allocating 80% of the data for training and 20% for testing.

An intriguing finding is that the model consultant agent tended to select the same models across different datasets, regardless of the varying characteristics of each dataset within its category. Specifically, for regression problems across nine datasets, the model consultant chose the Random Forest Regressor five times and Linear Regression four times. For time-series problems, ARIMA was chosen for all but one dataset, where SARIMA was selected instead. For classification problems, the agent consistently opted for the Random Forest Classifier.

This behavior suggests a significant limitation in our pipeline. Although there is no consensus in the literature regarding the best machine learning model for each problem type, we believe the agents' choices reflect the models most frequently encountered during the training of the LLM.

Despite this limitation, our prediction results are quite promising. Overall, our pipeline produced 11 excellent results, 9 good results, and 4 poor results. We encountered only six datasets where we could not generate predictions.

	Classification	**Regression**	**Time-Series**	**Total**
Very Good	5	5	1	11
Good	1	4	4	9
Poor	2	-	2	4
No-exists	2	1	3	6

Table 4: Predictions evaluations

Regarding the number of tokens used in this experiment, the overall mean was 37,875 tokens. By defining specific tasks, we were able to reduce the number of tokens each agent required. In table 5 you can check the mean of tokens by each type of categorie.

	Classification	**Regression**	**Time-Series**	**Total**
Tokens used	36941	38996	37688	37875

Table 5: Mean of tokens by categories

For a more detailed overview of the types of inputs used and the areas corresponding to each type of problem, please refer to the appendix where detailed tables are provided.

5 Conclusion and Future Work

This study developed and evaluated a multi-agent framework for automating data science tasks using large language models (LLMs) within the Autogen framework. The pipeline, comprising seven specialized agents, successfully processed 24 out of 30 datasets, producing accurate predictions and comprehensive reports. Key Findings of our research:

- Pipeline Performance: The pipeline demonstrated an 80% success rate across diverse machine learning tasks, highlighting its robustness and potential for automation.
- Prediction Quality: The framework produced very good results for 11 datasets, good results for 9, and poor results for 4, with no predictions generated for 6 datasets.
- Model Selection: The model consultant agent tended to favor familiar models, indicating a potential area for improvement in model diversity and selection criteria.
- Resource Efficiency: Optimization of agent tasks reduced the average token usage to 37,875 tokens, demonstrating efficient resource utilization.

Limitations of our research:

- Rate-Limit Constraints: API rate limits caused failures in complex datasets, indicating a need for improved workflow optimization.
- Model Selection Bias: The model consultant agent's preference for certain models suggests a need for more diverse training or refined selection algorithms.

- Hallucinations and Infinite Loops: Issues with agent hallucinations and infinite loops highlight the need for better error handling and context management.

This research demonstrates the feasibility of a multi-agent approach to automate data science workflows, paving the way for more efficient and scalable solutions in machine learning automation.

By refining the framework and addressing identified challenges, this work lays the groundwork for the development of fully autonomous data science pipelines, promising significant advancements in the field of automated machine learning.

In conclusion, our multi-agent framework represents a significant step towards automating the end-to-end data science process. The successful execution of tasks across diverse datasets, coupled with promising prediction results, validates the effectiveness of our approach. While challenges remain, the findings from this study provide a robust foundation for future research and development in the realm of automated data science, promising more efficient and scalable solutions for complex machine learning tasks.

5.0.1 Future Work

Future research should focus on addressing API rate limits, enhancing model selection diversity, and improving error handling mechanisms. Additionally, expanding the range of datasets can provide more comprehensive validation of the pipeline's capabilities.

Beyond enhancing our current approach, we aim to equip our agents with a diverse set of tools to further improve our methodology. Specifically, we will address the challenges faced by our model consultant agent by researching which models may best adapt to different types of problems and domains. For time-series models, we plan to test whether providing the model consultant/model builder with the current state-of-the-art foundation models enables them to enhance the overall approach.

Regarding the general pipeline, as mentioned at the beginning of the paper, we are actively researching the application of reinforcement learning to our pipeline. Existing research indicates that reinforcement learning can significantly enhance the performance of multi-agent systems, and we believe it will offer substantial benefits.

We encourage everyone to regularly check our library, as it will be frequently updated.

Acknowlegments

We would like to express our deepest gratitude to the AI Flow Solutions team for their invaluable support and contributions to this work. This research has been a significant endeavor for the past year, and it is incredibly rewarding to share our findings with the broader community.

Our sincere thanks go out to all the dedicated researchers who have inspired and guided us along the way. However, we owe the greatest debt of gratitude to the open source community in AI and Machine Learning. Your generosity in sharing research, code implementations, and knowledge has been instrumental in our growth and success. This paper is our humble attempt to contribute back to a community that has given us so much and continues to enrich our work every day.

Thank you for your ongoing support and collaboration.

References

[1] IBM. What is a machine learning pipeline? `https://www.ibm.com/cloud/learn/machine-learning-pipelines`, 2021.

[2] Stonebranch. Mlops and automation for machine learning pipelines. `https://www.stonebranch.com/solutions/mlops`, 2023.

[3] Yann LeCun. The critical need for scalable and efficient machine learning systems. `https://www.facebook.com/yann.lecun/posts/10159815944837143`, 2021.

[4] Zapier. What are ai agents? a comprehensive guide, 2023.

[5] Wikipedia. Multi-agent system, 2023.

[6] The Alan Turing Institute. Multi-agent systems, 2023.

[7] TJU-DRL-LAB. Multiagent-rl: The official code releasement of publications in marl field of tju rl lab, 2023.

[8] MARLlib. One repository for multi-agent reinforcement learning (marl), 2023.

[9] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

[10] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[11] Analytics Vidhya. Machine learning pipelines: Why adopted by data scientists? `https://www.analyticsvidhya.com/blog/2021/05/machine-learning-pipelines-why-adopted-by-data-scientists/`, 2021.

[12] AWS. Building, automating, managing, and scaling ml workflows using amazon sagemaker pipelines. `https://aws.amazon.com/sagemaker/pipelines/`, 2023.

[13] Yann LeCun. Real-time processing and decision-making in ai systems. `https://www.facebook.com/yann.lecun/posts/10159815944837143`, 2021.

[14] Yann LeCun. Scalable ai systems. `https://www.facebook.com/yann.lecun/posts/10159815944837143`, 2015.

[15] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[16] Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 2013.

[17] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[18] MARLlib. One repository for multi-agent reinforcement learning (marl), 2023.

[19] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *ArXiv preprint*, 2019.

[20] Nixtla. Timegpt: A model for time series forecasting, 2023.

[21] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: Building a unified time series model. *arXiv preprint arXiv:2403.00131*, 2024.

[22] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024.

[23] Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2024.

[24] H Hewamalage, C Bergmeir, and K Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[25] Debdoot Banik, Sourav Das, Debolina Ghosh, and Nirmalya Bandyopadhyay. Mlops: Machine learning operations, 2022.

[26] Azul Garza and Max Mergenthaler-Canseco. Timegpt-1, 2023.

[27] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024.

[28] Python core team. pickle — python object serialization, 2023.

[29] Sheila Teo. How i won singapore's gpt-4 prompt engineering competition. `https://towardsdatascience.com/how-i-won-singapores-gpt-4-prompt-engineering-competition-34c195a93d41`, 2023.

A Appendix

For a more detailed overview of the types of inputs used and the areas corresponding to each type of problem, please refer to the tables below.

The table 6 shows the full results that the agents have in the regression problems.

The table 7 shows the full results that the agents have in the regression problems.

The table 8 shows the full results that the agents have in the regression problems.

Data set	**Input**	**Report Generated**	**Normalized RMSE**	**Tokens Used**
Second Hand Car Price	I want to predict car price	yes	0.1048	31248
Cholesterol	I want to predict cholesterol	yes	0.0757	37728
Crab Age	I want to predict crab age	yes	0.1400	39753
Bike Rents for the Day	I want to predict bike rents for the day	yes	0.0125	65447
Fuel Consumption	I want to predict Fuel Consumption	yes	0.1078	42462
House Sales in King County	I want to predict house prices	Rate Limits	-	-
Medical Insurance Cost	I want to predict the cost of medical insurance	yes	0.0451	33197
Elevator Predictive maintenance	I want to predict the vibration	yes	0.2390	34177
Happiness Index	I want to predict the overall rank of happiness	yes	0.03222	40633
Student Performance	I want to predict students Performance	yes	0.0225	26316

Table 6: Detailed table in Regression

Data set	**Input**	**Report Generated**	**Accuracy**	**Tokens Used**
Airline Costumer Satisfaction	I want to predict whether future customers will be satisfied	yes	0.9574	34598
Anemia type	I want to predict the type of anemia	yes	0.9883	26771
Employee Attrition	I want to predict employee attrition	yes	0.8776	30568
Mushroom Classification	I want to predict the mushroom class (is edible or poisenous)	yes	1	30746
Obesity Classification	I want to predict the type of obesity	yes	0.9091	29742
Machine Predictive Maintenance	I want to predict the variable target	Rate Limits	-	-
Telecom costumer churn	I want to predict costumer Churn Category and Churn Reason	Hallucinates	-	-
Thyroid disease	I want to predict recurrence of thyroid cancer, if yes or no	yes	0.9870	22578
White wine quality	I want to predict the classification of the wine quality	yes	0.6594	21877
Red wine quality	I want to predict the classification of the wine quality	yes	0.6735	24864

Table 7: Detailed table in Classification

Data set	**Input**	**Report Generated**	**Normalized RMSE**	**Tokens Used**
Air Passengers	I want to forecast the passengers rate	yes	0.2253	30011
Daily Minimum Temperatures	I want to forecast the daily minimum temperature	Rate Limits	-	-
Eletric Production	I want to forecast the eletrical production	yes	0.2322	27856
Hourly Gasoline Prices	I want to forecast gasoline prices	yes	0.0184	39678
Microsoft Stock	I want to forecast microsoft stock	yes	0.3651	28895
Montlhy Beer Production	I want to forecast the monthly beer production	yes	0.2439	24720
Energy Consumption	I want to forecast energy consumption	yes	0.1396	82912
National Population	I want to forecast population growth	Rate Limits	-	-
Water Pum Rul	I want to forecast RUL of water pump	Rate Limits	-	-
Sales of Shampoo	I want to forecast the shampoos sale	yes	0.3555	19925

Table 8: Detailed table in Time-series