OPA - Observe, Preprocess, and Act: A Multi-Agent Framework for Data Preprocessing and Predictive Intelligence

Anonymous ACL submission

Abstract

The growing availability of temporal data across scientific, industrial, and governmental domains has increased the demand for scalable and reliable time series forecasting solutions. Despite advances in deep learning models and automated machine learning (AutoML) frameworks, building an effective forecasting pipeline remains a complex and decision-intensive process, often requiring expert knowledge and extensive manual intervention. To address these challenges, we propose Observe, Preprocess, and Act (OPA), a multi-agent forecasting framework that autonomously constructs end-to-end predictive pipelines for tabular time series data. OPA leverages recent advances in large language models and agentic artificial intelligence to coordinate specialized agents responsible for data inspection, preprocessing, model configuration, training, and evaluation, all supervised by a central orchestration agent. Users can interact with the system through natural-language prompts to specify objectives and constraints, or fully delegate decision-making to the agents when domain knowledge is unavailable. Experimental results show that OPA achieves forecasting performance comparable to standard AutoML pipelines while significantly improving transparency and interpretability through comprehensive, human-readable reports that explicitly document each modeling decision. These results demonstrate the potential of LLM-driven multi-agent systems to enhance the accessibility, explainability, and usability of AutoML for time series forecasting.

1 Introduction

The increasing availability of temporal data across scientific, industrial, and governmental domains has intensified the demand for reliable and scalable time series forecasting deep learning-based solutions (Kong et al., 2025). Despite substantial progress in forecasting algorithms and automated machine learning (AutoML) frameworks, constructing an effective predictive pipeline still requires a series of non-trivial decisions (He et al., 2021). These include data inspection, cleaning, imputation, feature transformation, model selection, hyperparameter tuning, and performance evaluation. For many practitioners, this process remains complex, time-consuming, and error-prone (Shchur et al., 2023).

Even experienced analysts often face challenges associated with heterogeneous data quality, missing contextual information, and the need to adapt workflows to diverse forecasting scenarios. Furthermore, although AutoML tools are designed as low-code solutions that require minimal programming expertise, their interpretability and accessibility could be significantly improved if users were able to directly interact with the system to specify objectives, constraints, and preferences during the model selection and configuration process.

In this context, recent breakthroughs in Large Language Models (LLMs) and Agentic Artificial Intelligence (Agentic AI) have enabled advances in autonomous artificial intelligence systems that explore the integration of multi-agent architectures to emulate, guide, or automate analytical workflows (Zhao et al., 2025). Such systems leverage specialized agents that collaborate to decompose complex tasks into manageable subtasks, enabling more transparent reasoning, modularity, and adaptability. Thus, we propose Observe, Preprocess, and Act (OPA), a multi-agent forecasting framework designed to autonomously construct an end-to-end predictive pipeline.

OPA enables users to supply tabular historical data along with a natural-language prompt describing domain considerations and modeling objectives. When domain knowledge is unavailable, the entire decision-making process can be fully delegated to the agents. The system is composed of specialized agents responsible for dataset inspection, prepro-

cessing, model configuration, training, and evaluation, all orchestrated by a supervisory agent that maintains coherence and transparency throughout the workflow.

In this way, even non–machine learning practitioners can directly interact with OPA by providing high-level, human-centered instructions and receiving comprehensive reports that explain the entire decision-making process. Such reports move toward self-explanatory machine learning, in which model choices and reasoning are explicitly articulated and can be inspected by users, thereby elevating the level of transparency and explainability in AutoML systems.Our key contributions are summarized as follows:

- We introduce a multi-agent framework, named *Observe, Preprocess, and Act (OPA)*[1], capable of perceiving a time series dataset and autonomously making data-science decisions, from preprocessing to the construction of AI-based forecasting models.
- We demonstrate how the LLM-driven multi-agent architecture enables users to explicitly describe the problem and constraints through human-based prompts, enhancing explainability and user control within the predictive modeling workflow.
- We show that integrating AutoML with autonomous agents yields minimal performance degradation while substantially improving the interpretability of the decision-making process and the resulting forecasting pipeline.
- We introduce a comprehensive suite of deterministic automated tests that systematically assess the behavior of the system at each decision stage, enabling rigorous, reproducible validation of the LLM-driven multi-agent workflow and its decision-making processes.

2 Related work

Recent literature defines Agentic AI as systems where LLMs operate within structured decision-making loops to plan and execute tasks, going beyond static content generation (Masterman et al., 2024). Regarding architectural patterns, (Masterman et al., 2024) categorizes these systems into vertical paradigms, where a coordinator manages sub-tasks, and horizontal paradigms, characterized by decentralized agent collaboration. To support autonomous operations, effective memory management is essential. While Retrieval-Augmented Generation (RAG) addresses context window limitations, (Wu et al., 2025) proposes retrieval necessity detection mechanisms, arguing that selective retrieval prevents performance degradation caused by irrelevant information and excessive latency.

In the domain of machine learning workflows, recent studies focus on applying LLM-based agents to preliminary data stages. Specifically, (Bravo-Rocca et al., 2025) and (Wang et al., 2025) demonstrate the utility of agents for feature engineering and data preprocessing, respectively, reducing the reliance on manual human effort. Distinct from these agentic data preparation approaches, the subsequent modeling phase is addressed by Automated Machine Learning (AutoML) frameworks. (Shchur et al., 2023) introduces AutoGluon, a system designed to automate model selection, hyperparameter tuning, and ensembling, providing a robust execution engine for tabular data tasks.

3 Observe, Preprocess, and Act (OPA)

The proposed Observe, Preprocess, and Act (OPA) approach is grounded in an autonomous multi-agent system capable of executing, in a coordinated manner, the stages of data preprocessing and time series forecasting. The system allows the user to provide a tabular dataset containing historical records of a phenomenon or process, along with an initial prompt describing relevant information to be considered during predictive model construction. Alternatively, if the user lacks expertise or detailed knowledge of the data, all decision-making steps can be fully delegated to the agents. As output, the system returns the preprocessed data, the selected predictive model, and a complete record of the reasoning process generated by the multi-agent system.

3.1 Architecture

The methodology developed in this work aims to construct a workflow composed of specialized agents responsible for data preprocessing and predictive model development. All operations are orchestrated by a Supervisor agent, which coordinates the sequence of tasks according to the current state of the predictive process. In addition,

[1]The code will be available after the review process

long-term memory is implemented through the Retrieval-Augmented Generation (RAG) technique, integrated into the system as an external tool that agents may query whenever additional information is required to support decision-making. The overall agent-orchestration architecture is illustrated in Figure 1.

Within this architecture, agents can be categorized according to the three communication modes they employ. The first category, *agent-call-agent* (highlighted in orange), is represented by the Supervisor agent, which invokes specialized agents to perform specific tasks based on the current phase of the forecasting workflow. The second category, *agent-call-tools*, consists of three preprocessing-dedicated agents, Inspect, Imputator, and Feature Engineer, and an AutoML agent. These agents handle the core stages of the information-extraction and preprocessing pipeline, as well as the forecasting stage. Finally, the *agent-resumer* neither employs additional tools nor invokes other agents; instead, its function is to summarize the system's reasoning process and present it to the user in an interpretable and concise form.

3.1.1 Nodes

The OPA system is composed of a graph containing six agent nodes and two indicative states. The nodes interact in a coordinated manner to achieve the objective defined by the user through a prompt. The Start state marks the beginning of the process, receiving as input a time series dataset (in `.csv` format). The End state indicates the completion of the predictive task after all intermediate stages have been executed. The six agents are connected bidirectionally, enabling information exchange and state updates whenever necessary. These agents operate collaboratively, each with a clearly defined role within the workflow. Their main functions are described below:

Supervisor: This agent is responsible for the global orchestration of the process. Its role is to determine, based on the current state of the pipeline, when and which specialized agent should be activated. Initially, the Supervisor receives the dataset originating from the Start state and forwards it to the Inspect agent for a preliminary analysis. Throughout the workflow, it evaluates both intermediate results and the evolving needs of the pipeline, coordinating the sequential or conditional execution of the remaining agents. It may also consult the RAG memory whenever additional decision support is required.

Inspect: This agent performs an initial inspection of the data, checking for missing values, outlier detection, structural inconsistencies, and other statistical characteristics relevant to the time series. It also produces a descriptive report that guides subsequent decisions, directing the pipeline toward imputation, feature engineering, or directly to AutoML, depending on the data quality.

Imputator: This agent is responsible for handling missing values, noise, and potential anomalies detected by the Inspect agent. It applies appropriate imputation techniques, such as statistical methods, temporal interpolation, or machine learning–based imputation, to ensure the consistency of the time series for the following stages. It also records not only the technique applied but the rationale behind the chosen approach based on the current data state.

Feature Engineer: This agent constructs new variables aimed at optimizing the model's predictive capability. These include lag features, sliding windows, temporal decomposition, seasonal indicators, and mathematical transformations. It also assesses the relevance of the engineered features, prioritizing those that best capture the temporal patterns of the phenomenon.

AutoML: This agent is responsible for testing, comparing, and selecting the most suitable predictive model for the given problem. It evaluates different algorithms, hyperparameter configurations, and validation strategies, ultimately returning the model with the best performance according to predefined metrics. AutoML also produces a performance report, which is later used by the Summarizer agent.

Summarizer: The Summarizer synthesizes the entire reasoning process carried out by the previous agents, generating a clear and interpretable explanation for the user. Its role is to transform technical decisions, such as applied imputations, generated features, evaluated models, and the Supervisor's justifications, into a coherent narrative describing the complete solution flow, from input to the final model delivered.

It is relevant to note that the preprocessing stage in OPA is structured around three specialized nodes Inspect, Imputation, and Feature Engineering. Appendix A provides a detailed description of the tools and procedures implemented within each of these nodes.

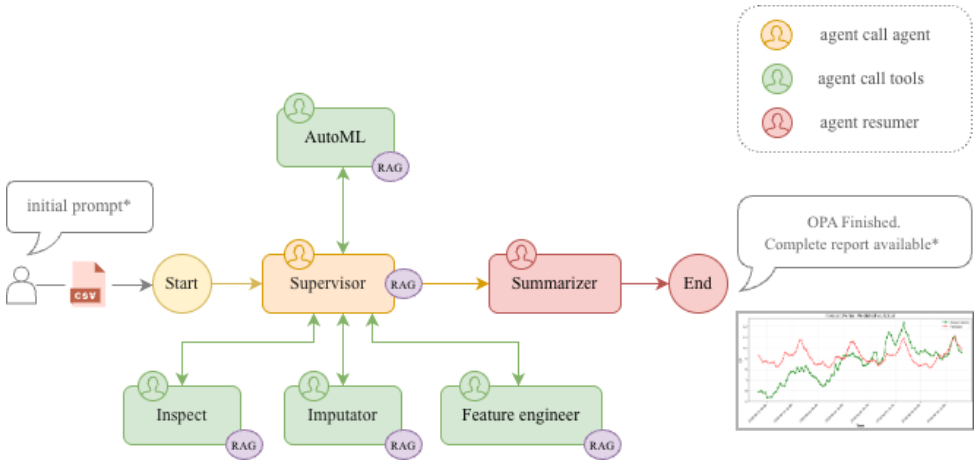


Figure 1: OPA model architecture: from preprocessing to configuration definition

3.1.2 Retrieval-Augmented Generation (RAG)

An important aspect of our methodology is the retrieval of relevant past experiences. The RAG serves as an internal memory mechanism and is available as a tool to every agent node in the workflow. One source of information for the RAG is a document, in which we summarized the best interpolation and feature extraction methods for time series, as well as their applicability. Also, in this architecture, the memory stores insights and outcomes from previous actions that contributed to the improvement of node performance across the workflow. This enables any agent to query the memory when faced with uncertainty about the most appropriate approach for a specific situation.

Our RAG was implemented as a tool in order to let each agent decide if there is a need to call it. Note that it could also be implemented as an optional node connected to every other node, but we chose to avoid unnecessary routing functions in the workflow.

3.2 Implementation Details

The development was carried out in Python (3.12), using libraries such as `scikit-learn`, `statsmodels`, `pandas`, and `numpy` for statistical analysis and data manipulation. `AutoGluon`, an AutoML framework, was selected due to its ability to automate critical modeling and optimization steps.

The graph structure that composes the OPA system was developed using the *LangGraph*[2] framework, which is an agent orchestration control and provides suitable abstractions for modeling states, nodes, and transitions in multi-agent systems . This choice facilitates the formal definition of the execution flow, the management of dependencies among agents, and the implementation of state-passing mechanisms throughout the workflow.

The RAG's vector database in our implementation is FAISS, an open-source library for similarity search and vector clustering (Douze et al., 2025). Additionally, the embedding model chosen was BAAI/bge-base-en-v1.5 (Xiao et al., 2023).

4 Experimental Design

Experiments was conducted on time-series benchmarks. Different data-loss scenarios (with random gaps or blocks of missing values) will be simulated to assess preprocessing robustness. The large language model (LLM) chosen for the project is *Qwen/Qwen3-Next-80B-A3B-Instruct*.

Maintaining the same range of model options, we configured AutoGluon to use the "*high_quality*" preset. Under this setting, the following models were trained: SeasonalNaive, RecursiveTabular, DirectTabular, NPTS, DynamicOptimizedTheta, AutoETS, Temporal Fusion Transformer (TFT), DeepAR, PatchTST, TiDE, and WeightedEnsemble.

4.1 Baselines

As baselines, we compared the OPA framework against TimeSeriesScientist (TSci) (Zhao et al., 2025), which employs a similar multi-agent strategy, and against raw AutoGluon (Shchur et al., 2023), that is, AutoML used in isolation, in order to assess the effects of OPA's orchestration mechanism.

[2] `https://www.langchain.com/langgraph`

4.2 Datasets

To evaluate the performance of the OPA framework on time series forecasting tasks, we adopt a comprehensive set of benchmarks widely used in the literature. Experiments are conducted on five well-established real-world datasets: ETT (Zhou et al., 2021), Electricity (UCI), Weather (Wetterstation), Exchange (Lai et al., 2018), and ILI (CDC). This diverse collection of benchmarks covers multiple practical application domains, enabling a robust evaluation of the framework's performance.

Dataset	Length	Covariates	Sampling Period
ETTh 1/2	17420	7	1 hour
ETTm 1/2	69680	7	15 min
Electricity	26304	321	1 hour
Weather	52696	21	10 min
Exchange	7588	8	1 day
ILI	966	7	1 week

Table 1: Dataset details of benchmark datasets.

4.3 Evaluation Plan

4.3.1 Predictions metrics

To evaluate the performance of our OPA prediction capability, we used the following forecasting metrics errors:

- Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

- Mean Absolute Percentage Error (MAPE)

$$\text{MAPE} = \sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{2}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value and n is the number of predicted samples.

4.3.2 Per node evaluation

A per-node evaluation of the preprocessing agents was performed using an automated testing suite to verify that the inspection, imputation, and feature engineering nodes fulfilled their intended roles. The use of automated testing is particularly advantageous in this context, as it provides deterministic and reproducible validation of each decision step, enabling systematic assessment of agent behavior and facilitating reliable verification of the multi-agent workflow.

4.3.3 Case Study - Hallucination and Qualitative analysis

We conducted a human evaluation to assess **hallucinations** in our multi-agent setting, focusing on whether agents generated information unsupported by the provided context (with ETTH1 as the source dataset) or violated explicit task constraints. In this work, hallucinations are defined as a phenomenon in which generated content is nonsensical, incorrect, or unfaithful to the source information. Following prior literature, we specifically analyze hallucinations related to fidelity across three dimensions: (a) **logical consistency** which measures internal coherence and whether the reasoning and conclusions follow from the information available, (b) **contextual consistency** which measures whether the answer strictly follows and does not contradict the given context, and (c) **instruction consistency** which measures adherence to the explicit task requirements and constraints(Huang et al., 2025).

To quantify these aspects, we adopted a human scoring protocol based on the scale proposed by (Murugadoss et al., 2025). Each model response was rated on a five-point ordinal scale: 1 — not consistent, 2 — slightly consistent, 3 — moderately consistent, 4 — mostly consistent, and 5 — totally consistent.

We compared two large language model backbones, namely Qwen 3 (80B) and GPT OSS (120B). Both models were integrated into the same multi-agent architecture and evaluated under identical task configurations and prompts. Hallucinations were assessed exclusively through human evaluation, in which annotators examined the agents' outputs for factual correctness, grounding in the provided context, and adherence to task instructions.

In addition to the manual Likert protocol, we implemented an *LLM-as-a-judge* procedure to obtain a lightweight and reproducible evaluation of the quality of analysis reports produced by the multi-agent workflow. Each evaluation instance is constructed from an agent trace sample containing: (i) the agent name and step, (ii) the original prompt, and (iii) the agent's textual output (including intermediate reasoning/report snippets). The judge is instructed to disregard factual correctness and hallucinations, and to score only the structure and communicative quality of the response.

We adopted a five-point Likert scale (1–5) and

requested structured JSON outputs to reduce parsing ambiguities. For each sample, the evaluator assigned integer scores for: local coherence, global coherence, logical progression, clarity of explanation, and alignment with the agent's objective. We conducted the evaluation with three judges on the same set of prompts/samples: one human expert evaluator (manual evaluation) and two independent LLM judges (*LLM-as-a-judge*), running under deterministic decoding (temperature = 0). We then aggregated the scores across samples to characterize the overall quality of the reports and performed a qualitative inspection of discrepancies. This *LLM-as-a-judge* signal is used as a complement to human evaluation, not as a substitute.

5 Results

5.1 Predictions evaluation

The predictions made by OPA are presented in Table 2. The table compares the baselines with OPA on eight benchmarks (ETTh1/2, ETTh1/2, Weather, Exchange, ILI, ECL/Electricity) for the horizons 96 (except for ILI dataset which was executed under the horizons 24). In the table, bold indicates the best score and underlined the second best. The complete table for different horizons are presented in the Appendix C in Table 5.

Overall, OPA achieves the greatest number of top-ranked MAE scores (four datasets) and matches the highest count of best MAPE values (three datasets), indicating it most frequently provides the most accurate absolute forecasts while remaining competitive in relative error. AutoGluon secures the best MAE on Weather, Exchange, and ILI and the best MAPE on ETTh1, Weather, Exchange, and ILI, showing strong performance on those series. TSci obtains the lowest MAPE on ETTh2, Weather, and Exchange, suggesting it can excel in relative error accuracy for certain data.

Figure 2 presents a visual comparison of MAE results on the ETTh1 dataset across forecasting horizons. AutoGluon is shown in blue, OPA in green, and TSci in orange. Both OPA and AutoGluon achieve lower MAE values, indicating superior performance across most horizons. OPA outperforms AutoGluon at horizons 96 and 720, while achieving comparable performance at horizons 192 and 336. Equivalent visualizations for all evaluated datasets are reported in Appendix C.

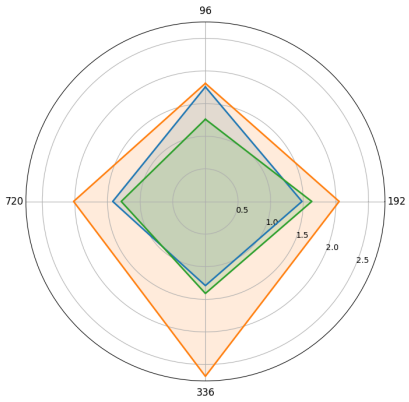
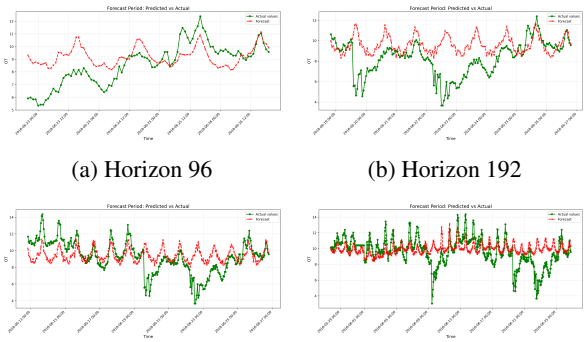


Figure 2: Comparison of MAE performance on the ETTh1 dataset for multiple forecasting horizons. Results are reported for the three evaluated approaches, where lower MAE values correspond to better performance



(a) Horizon 96 (b) Horizon 192

(c) Horizon 336 (d) Horizon 720

Figure 3: Visualization results produced by OPA for the ETTh1 dataset across multiple forecasting horizons. In green the actual values and in red the predicted values.

5.2 Per node evaluation

The automated testing suite initially removes a predefined percentage of data values to evaluate whether OPA can correctly reconstruct them during execution. The process then extracts basic dataset metadata, including the original data shape and the columns containing missing values.

The Inspection node verifies whether OPA can correctly read the dataset and accurately identify its structural properties. The Imputation node assesses the suitability of the selected imputation strategy and reports the chosen method, along with a comparison between original and imputed values through an error column for further validation. The Feature Engineering node evaluates whether new features are effectively generated to enhance predictive performance. Finally, the Supervisor node reports the executed pipeline, detailing the sequence of actions, decisions, and potential loops.

Dataset	AutoGluon		TSci		OPA (Ours)	
	MAE	MAPE (%)	MAE	MAPE (%)	MAE	MAPE (%)
ETTh1	1.75	21.53	1.81	**13.90**	**1.26**	16.84
ETTh2	5.97	17.33	**4.50**	18.90	4.97	**14.44**
Weather	**4.47**	**1.04**	16.30	3.80	5.47	1.28
Exchange	**0.0019**	**0.28**	0.0246	3.80	0.018	2.54
ILI	**0.19**	**14.43**	3.63	40.50	0.34	25.49
ETTm1	1.60	19.80	1.68	15.70	**0.30**	**3.07**
ETTm2	1.53	20.56	3.63	40.50	**0.28**	**2.87**
ECL (Electricity)	1.97	20.01	394.0	**11.20**	**1.56**	23.39
# 1st (Best)	3	**3**	1	2	**4**	**3**
# 2nd Best	**4**	3	0	1	**4**	**4**

Table 2: Time series forecasting results for 96 steps ahead (except the ILI dataset with 24). Lower values indicate better performance. Best results are in **bold**, second-best are underlined and even uses **both**.

Category	Qwen 3 (80B)	GPT OSS (120B)
Context consistency	**5.0**	3.6
Logical consistency	**5.0**	2.6
Instruction consistency	**4.7**	2.4

Table 3: Average scores for each prompt category with ETTH1 as source dataset. Bold indicates higher scores.

A consolidated summary is produced to enable in-depth validation of the entire workflow.

In our experiments, the automation tests passed in all cases, showing deterministically that OPA preprocessing phase was behaving as expected. Appendix B details the construction of the automated testing suite and presents an example of the reports generated by it.

5.3 Case Study - Hallucination and Qualitative analysis

The work assessed the incidence of hallucinations across 28 prompts, including prompts for tasks outside the intended scope of OPA. Table 3 shows the results, the first model (Qwen 3 80B) successfully completed nearly all evaluation instances without producing hallucinated content. In contrast, the second model (GPT OSS 120B) exhibited a high frequency of hallucinations across multiple tasks.

The models showed difficulty in handling prompts that requested information not contained in the datasets but still required some type of numerical data, such as temperature, in a context different from that of the dataset. For instance, Prompt 1 (shown in Appendix D.2) caused even the Qwen model to hallucinate when asked about the temperature of a city, despite the dataset containing only temperature-related variables associated with eletricity transformers.

In addition to a lower incidence of hallucinations, the smaller model also exhibited refusal behavior when confronted with malicious prompts. This improved performance with respect to hallucinations may be related to the higher complexity of larger models with more parameters, which tend to produce lower-entropy outputs and may overestimate the likelihood function in out-of-distribution scenarios.

Table 6, available in Appendix D.2, indicates that Qwen 3 (80B) achieves near-ceiling scores across all three fidelity dimensions, with averages of 5.0 for context consistency and logical consistency, and 4.7 for instruction consistency. In contrast, GPT OSS (120B) shows substantially lower scores, especially in logical consistency (2.6) and instruction consistency (2.4), indicating lower reliability in step-by-step reasoning and a higher frequency of deviations from prompt-imposed constraints.

Beyond the quantitative results, we noted a recurring discrepancy when the agent fails to solve a task (due to context limitations, tool constraints, or insufficient information). In these cases, the human expert evaluator frequently assigned high marks for coherence, logical progression, and alignment, as the response remained well-structured and correctly justified the constraints. In contrast, the LLM judges tended to give low marks, apparently conflating the failure to produce the final solution with poor reasoning quality, interpreting "not solving" as "not cohesive" or "not logical." This pattern suggests that, without an explicit answerability/feasibility dimension, automatic evaluation may underestimate the quality of reasoning in scenarios where the correct behavior is to recognize infeasibility, explicitly state limitations, and maintain internal consistency.

Dataset	Horizon	OPA (Ours)		OPA Ablation	
		MAE	MAPE (%)	MAE	MAPE (%)
ILI	24	0.348	25.49	**0.342**	**25.03**
	36	0.247	18.25	**0.241**	**17.74**
	48	**0.341**	**27.32**	0.349	27.33
	60	**1.050**	**48.28**	2.050	139.92
ETTm1	96	**0.300**	**3.07**	0.363	3.83
	192	0.970	10.17	**0.933**	**9.78**
	336	**1.200**	15.18	1.206	**14.32**
	720	**1.290**	**18.70**	1.821	28.12
# 1st (Best)		**5**	**4**	3	4

Table 4: Performance comparison between OPA and its ablation variants on the ILI and ETTm1 datasets. Lower values indicate better performance.

6 Discussion

The results presents in section 5 shows that the modular, agent-driven architecture of OPA not only preserves AutoGluon strong baseline but also leverages coordinated agents to fine-tune preprocessing, feature engineering, and model selection, yielding superior accuracy especially at longer horizons where raw AutoML struggles. Overall, the data confirms that OPA's multi-agent design translates into better or competitive forecasting performance across diverse time-series tasks.

The main advantage of OPA compared to TSci lies in its integration with AutoGluon, which leverages more modern machine learning models for time series forecasting. This benefit comes at the cost of relying on third-party frameworks; however, OPA is designed with a modular architecture that facilitates the integration of additional AutoML tools and frameworks within its agent-based pipeline.

7 Ablation Study

An ablation study was conducted by removing all preprocessing nodes from OPA, leaving only the native preprocessing mechanisms of the AutoML component. The objective of this experiment is to evaluate whether the dedicated OPA preprocessing pipeline has a measurable impact on the final AutoML forecasting performance.

In Table 4, the last row report the number of best results across horizons for each metric, where lower values indicate better performance and ties are counted as first-best for both methods. Our proposed multi-agent system demonstrates consistent superiority over the ablation baseline across key error metrics, achieving significant improvements in both MAE and MAPE. This advantage is particularly pronounced when dealing with longer forecast horizons.

This performance profile is noteworthy: the clear advantage in MAE indicates our system produces more accurate point forecasts with better containment of large errors. Furthermore, the inherent modularity of our architecture allows each specialized agent (e.g., for imputation, feature extraction, or AutoML) to be independently upgraded or replaced as superior methods emerge, ensuring long-term adaptability without systemic redesign.

8 Conclusions

This work presents the development of OPA, a multi-agent framework designed to automate data preprocessing and forecasting for tabular time series data. The experimental results demonstrate that OPA achieves performance comparable to a direct application of AutoGluon, thereby fulfilling its primary objective without degrading predictive accuracy.

Furthermore, the proposed multi-agent architecture shows that a complete data science pipeline can be effectively automated using pre-trained expert LLMs through structured orchestration and carefully designed prompts. Importantly, this automation does not compromise explainability, as all intermediate reasoning steps, decisions, and actions can be audited and interpreted by humans.

Finally, hallucination and coherence evaluations provide additional evidence of the reliability and robustness of the proposed approach, reinforcing OPA as a trustworthy framework for automated time series forecasting.

Limitations

Despite its flexibility and strong empirical performance, the OPA framework presents limitations

that should be acknowledged.

OPA relies on LLMs to coordinate decisions across preprocessing, modeling, and evaluation stages. Consequently, its behavior is bounded by the reasoning consistency, domain knowledge, and failure modes of the underlying LLM. While deterministic testing mitigates variability, suboptimal or misleading reasoning may still occur, particularly in edge cases or under ambiguous data conditions.

The multi-agent design introduces additional computational overhead compared to traditional AutoML pipelines. Sequential agent execution, repeated dataset inspections, and iterative decision-making increase latency and resource consumption, which may limit applicability in real-time or large-scale industrial settings.

OPA prioritizes transparency and explainability over exhaustive search. As a result, it does not guarantee globally optimal model or hyperparameter selection when compared to brute-force or large-budget AutoML methods. Instead, its decisions reflect heuristic reasoning guided by agent prompts and learned priors.

System performance and behavior depend on prompt formulations, tool definitions, and agent boundaries. Although OPA is modular, changes in agent instructions or tool interfaces may affect decision consistency, requiring careful design and validation to ensure stable behavior across deployments.

The current preprocessing agents are tailored primarily to structured time series data. While extensible, the framework may require non-trivial adaptation to handle other data modalities (e.g., text, images, graphs) or domain-specific constraints such as irregular sampling, missing not at random (MNAR) patterns, or strong causal dependencies.

References

Gusseppe Bravo-Rocca, Peini Liu, Jordi Guitart, Rodrigo M Carrillo-Larco, Ajay Dholakia, and David Ellison. 2025. Feature engineering for agents: An adaptive cognitive architecture for interpretable ml monitoring. *Preprint*, arXiv:2506.09742.

CDC. Illness. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *Preprint*, arXiv:2401.08281.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Xiangjie Kong, Zhenghao Chen, Weiyao Liu, Kaili Ning, Lechao Zhang, Syauqie Muhammad Marier, Yichen Liu, Yuhao Chen, and Feng Xia. 2025. Deep learning for time series forecasting: a survey. *International Journal of Machine Learning and Cybernetics*, 16(7–8):5079–5112.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *Preprint*, arXiv:2404.11584.

Bhuvanashree Murugadoss, Christian Poelitz, Ian Drosos, Vu Le, Nick McKenna, Carina Suzana Negreanu, Chris Parnin, and Advait Sarkar. 2025. Evaluating the evaluator: Measuring llms' adherence to task evaluation instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19589–19597.

Oleksandr Shchur, Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Yuyang Wang. 2023. Autogluon-timeseries: Automl for probabilistic time series forecasting. *Preprint*, arXiv:2308.05566.

UCI. Electricity. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.

Peiran Wang, Yaoning Yu, Ke Chen, Xianyang Zhan, and Haohan Wang. 2025. Large language model-based data science agent: A survey. *Preprint*, arXiv:2508.02744.

Wetterstation. Weather. https://www.bgc-jena.mpg.de/wetter/.

Hui Wu, Xiaoyang Wang, and Zhong Fan. 2025. Addressing the sustainable ai trilemma: a case study on llm agents and rag. *Preprint*, arXiv:2501.08262.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. 2025. Timeseries-scientist: A general-purpose ai agent for time series analysis. *Preprint*, arXiv:2510.01538.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

A Preprocessing steps

The OPA preprocessing pipeline is structured around three dedicated agent nodes: Inspect, Imputator, and Feature Engineering, each responsible for a distinct stage of data preparation. In the following subsections each of them is detailed.

A.1 Inspection Agent

The inspection agent is responsible for performing an initial, systematic assessment of the input time series dataset. Given a pre-loaded pandas DataFrame, the agent analyzes data quality and structural characteristics to inform subsequent preprocessing decisions. Specifically, it computes descriptive statistics for all available attributes, identifies missing values and their distribution, and detects the presence of invalid numerical entries such as infinite values. To ensure robustness, the agent sanitizes the data by replacing infinite values with null representations prior to analysis. The resulting summary provides a comprehensive overview of the dataset's integrity, highlighting potential issues related to completeness, scale, and consistency. This diagnostic information is then propagated to downstream preprocessing agents, enabling informed decisions regarding imputation strategies and feature engineering operations.

A.1.1 Inspect problem formulation

Let $\mathcal{D} = \{X_t^i\}_{t=1,i=1}^{T,N}$ denote a multivariate time series dataset represented as a pandas DataFrame, where T is the temporal length and N is the number of variables (e.g., sensors or features). The inspection agent operates on $\mathcal{D}$ to perform an initial diagnostic assessment of data quality and structural properties prior to downstream preprocessing and model training.

For each variable X^i, the agent computes descriptive statistics, formally defined as

$$\mathcal{S}(X^i) = \{\mu_i, \sigma_i, \min_i, \max_i, q_{25,i}, q_{50,i}, q_{75,i}\}, \tag{3}$$

where μ_i and σ_i denote the mean and standard deviation, and $q_{25,i}$, $q_{50,i}$, and $q_{75,i}$ correspond to the first, second, and third quartiles, respectively.

The agent further evaluates dataset completeness by quantifying missing values through

$$\mathcal{M}(X^i) = \sum_{t=1}^{T} \mathbb{I}\left(X_t^i = \text{NaN}\right), \tag{4}$$

and identifies numerically invalid entries such as infinite values using

$$\mathcal{I}(X^i) = \sum_{t=1}^{T} \mathbb{I}\left(X_t^i \in \{+\infty, -\infty\}\right), \tag{5}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

To ensure robustness in subsequent preprocessing stages, all infinite values are mapped to null representations prior to analysis. The resulting inspection report provides a structured summary of distributional

A.2 Imputator Agent

The Imputator Agent is designed to autonomously determine the most suitable missing-data imputation strategy based on the characteristics of the dataset identified during the inspection stage. Instead of relying on fixed heuristics, the agent adopts a knowledge-guided decision process that integrates contextual reasoning with retrieval-augmented domain expertise.

A.2.1 Imputator problem formulation

Let $\mathcal{D}$ denote the input dataset and let $\mathcal{C}(\mathcal{D})$ represent a contextual summary extracted during inspection, including information about missing-value patterns, variable types, temporal structure, and inter-variable dependencies. Prior to making a decision, the agent queries an external knowledge base $\mathcal{K}$ containing formal descriptions of interpolation and imputation techniques and their recommended application domains.

The agent's objective is to learn a deterministic decision function

$$f_{\text{imp}} : \mathcal{C}(\mathcal{D}) \times \mathcal{K} \to \mathcal{M}, \tag{6}$$

where $\mathcal{M} = \{\text{KNN}, \text{MICE}, \text{GP}\}$ denotes the set of admissible imputation methods.

The selected method is conditioned on the inferred data properties. KNN imputation is favored for locally correlated data, such as sensor measurements, due to its computational efficiency.

MICE is selected for multivariate datasets exhibiting complex and potentially non-linear dependencies among variables. Gaussian Process (GP) regression is chosen for time series with strong temporal correlations or when explicit uncertainty quantification is required, at the cost of higher computational complexity.

The final output of the Imputator Agent is a structured JSON object

$$\mathbf{y}_{\text{imp}} = \{\text{method}, \text{params}\}, \tag{7}$$

which specifies both the chosen imputation technique and its corresponding hyperparameters. This deterministic and machine-readable representation ensures reproducibility, traceability, and seamless integration with downstream preprocessing and modeling components of the OPA framework.

A.3 Feature Engineering Agent

The Feature Engineering Agent is responsible for autonomously constructing informative feature representations from the preprocessed dataset, with the goal of improving downstream forecasting performance while preserving interpretability and statistical coherence. The agent follows an analysis-driven feature construction strategy, explicitly linking data diagnostics to transformation decisions.

A.3.1 Feature Engineering problem formulation

Let $\mathcal{D} \in \mathbb{R}^{T \times F}$ denote the input dataset with T time steps and F original variables. The agent first performs an exploratory analysis

$$\mathcal{A}(\mathcal{D}) = \{\text{stats}, \text{variance}, \text{temporal index}\}, \tag{8}$$

where descriptive statistics, variance measures, and temporal resolution are evaluated. In particular, the ratio between standard deviation and mean is used to assess signal noise and guide the selection of smoothing or aggregation-based transformations.

Based on this analysis and on retrieval-augmented domain knowledge $\mathcal{K}$, the agent selects a set of feature transformations

$$f_{\text{feat}} : \mathcal{D} \times \mathcal{A}(\mathcal{D}) \times \mathcal{K} \to \mathcal{D}', \tag{9}$$

where $\mathcal{D}'$ denotes the augmented dataset containing the newly engineered features.

Depending on the inferred signal characteristics, the agent may generate statistical features (e.g., mean, variance, entropy), geometric descriptors (e.g., slopes or amplitudes), or temporal smoothing features such as rolling-window statistics and exponentially weighted moving averages. Feature hyperparameters, such as window sizes or decay factors, are explicitly justified based on data frequency or domain assumptions.

When parameter uncertainty exists, the agent evaluates multiple candidate features and retains only those maximizing informative criteria such as variance or stability, discarding redundant or zero-variance features. All transformations are applied directly to $\mathcal{D}$, ensuring consistency and traceability.

The final output of the Feature Engineering Agent consists of the augmented dataset $\mathcal{D}'$ and a concise textual report enumerating the newly created features along with their conceptual justification. This design promotes transparency, reproducibility, and principled feature construction within the OPA framework.

B Automated Testing

To ensure reliability, reproducibility, and transparency, we design a deterministic end-to-end (E2E) automated testing suite for the OPA multi-agent framework. The suite systematically validates each agent node and their interactions across the workflow, enabling consistent verification of agent behavior across repeated executions and datasets.

The framework executes the full multi-agent pipeline on user-selected datasets and optionally injects artificial missing values at predefined rates into numeric variables. This controlled corruption enables direct comparison between imputed outputs and known ground truth, allowing objective evaluation of imputation accuracy using quantitative error metrics.

Intermediate dataset states corresponding to the inspection, imputation, and feature engineering nodes are reconstructed from execution logs and internal state transitions. This approach enables per-node validation without intrusive instrumentation. Specialized analyzers assess task correctness, decision consistency, and expected structural changes at each node.

Specifically, the Inspector node is evaluated on missing-value detection and dataset characterization, the Imputator node on method selection, execution success, and reconstruction error, and the Feature Engineering node on the number, type, and validity of generated features. Additional evalua-

tions cover visualization generation, supervisory planning quality, error handling, and AutoML execution.

For each dataset, the framework automatically produces a structured report summarizing metrics, agent decisions, and detected issues. When multiple datasets are tested, comparative summaries are generated. Overall, this deterministic and modular testing strategy provides a rigorous validation mechanism for LLM-driven multi-agent systems, where conventional unit testing is insufficient to capture complex decision processes.

B.1 End-to-End AutoML Execution Report

```
========================
END-TO-END TESTING
Dataset: ETTh1
Date: 2025-12-23 09:18:09
========================
DATASET INFORMATION
- - - - - - - - - - - - - -
- Shape Original: (17420, 8)
- Shape Final: (17420, 8)
- Original columns: 8
- Final columns: 8
1. INSPECTION NODE
- - - - - - - - - - - - - -
- % Missing values detected: 13.15%
- Extracted information: Shape Identified, Dtypes Identified,
Statistics Computed, Columns Listed
- Score of extraction: 100.0%
2. IMPUTATOR NODE
- - - - - - - - - - - - - -
- Choosen Method: knn
- Imputed values: 18328
- Execution done: Yes
- Errors vs Original values:
Column MAPE (%) RMSE MAE
HUFL 100.86% 6.6726 4.7000
HULL 109.52% 1.5697 1.1972
MUFL 111.25% 6.4222 4.4104
MULL 111.53% 1.3823 1.0483
LUFL 21.91% 0.9081 0.6310
LULL 43.00% 0.4880 0.3300
OT 83.97% 8.0530 5.9869
MAPE Average overall: 83.15%
3. FEATURE ENGINEERING NODE
- - - - - - - - - - - - - -
- Features Adicionadas: 3
- % Faltante Após: 0.00%
- Lista de Features: day_of_week, hour, month
- Tipos de Features: Temporal (3) - day_of_week, hour, month
5. SUPERVISOR NODE
- - - - - - - - - - - - - -
- Total decisions: 5
- Sequence: inspect → supervisor → imputator → supervisor →
inspect → automl
- Loops: 4
- Total errors: 1 - Solved: 1 - Non solved: 0 - Resolution
rate: 100.0%
- Quality of resolution: 60.0%
6. AUTOML NODE
- - - - - - - - - - - - - -
- Executed: Yes
- Model: AutoGluon
================ FINAL Summary ===============
1. Inspected the dataset and found it contained 17,420 rows
and
8 columns, with all numeric columns (HUFL, HULL, MUFL, MULL,
LUFL, LULL, OT) exhibiting 14.48%–15.31% missing values and
the 'date' column stored as object type rather than datetime.
2. Confirmed the data was chronologically sorted by the
'date' column, validating its suitability for time series
analysis.
3. Applied KNN imputation with n_neighbors=5 to all
numeric columns, successfully reducing missing values while
preserving temporal structure.
4. Converted the 'date' column from object to datetime64[ns]
format, ensuring no missing values were introduced and
enabling proper temporal operations.
5. Engineered comprehensive time series features: lag
features (lags 1–3) for all 7 numeric columns, rolling window
features (mean and std for windows of 3, 6, and 12 hours), and
temporal features (hour, day_of_week, month, is_weekend),
all aligned with energy forecasting domain knowledge.
6. Trained a forecasting model using AutoML with OT as the
target variable, a 24-hour prediction horizon, and MAE as the
evaluation metric; the model achieved MAE=1.45, RMSE=3.55,
and MAPE=10.03%, with predictions and visualizations saved
to disk. 7. Encountered an output parsing error in the
AutoML agent's response format, but confirmed the modeling
process succeeded and all outputs were correctly generated
and stored.
8. Documented a critical insight: AutoML workflows must be
configured with handle_parsing_errors=True to accept valid
model outputs despite JSON formatting issues.
```

C Additional Experiments Results

Table 5 reports the forecasting performance across multiple datasets and prediction horizons, comparing AutoGluon, TSci, and the proposed OPA framework. Results are evaluated using MAE and MAPE, where lower values indicate better accuracy. Overall, OPA demonstrates competitive and often superior performance, particularly for medium and long forecasting horizons, achieving several best and second-best results across diverse datasets. These results highlight the robustness of OPA across different temporal dynamics and data characteristics, while remaining competitive with established AutoML and time-series–specific baselines.

C.1 Vizualization

The complete visualization of MAE results is presented in Figure 4.

D Prompts

D.1 OPA Agents Prompt Specification

The following prompts define the system-level instructions used to configure the agents responsible for data preprocessing and time series forecasting reported in this paper. The prompts are included verbatim to ensure reproducibility.

Dataset	Horizon	AutoGluon		TSci		OPA (Ours)	
		MAE	MAPE (%)	MAE	MAPE (%)	MAE	MAPE (%)
ETTh1	96	1.75	21.53	1.81	**13.9**	**1.26**	16.84
	192	**1.48**	**22.82**	2.05	31.0	1.63	25.23
	336	**1.29**	**16.49**	2.68	31.7	1.41	18.34
	720	1.42	16.02	2.02	23.3	**1.29**	**16.00**
ETTh2	96	5.97	17.33	4.50	18.9	4.97	**14.44**
	192	6.44	19.30	4.47	12.8	**4.13**	**11.69**
	336	**3.47**	**9.88**	3.81	10.7	3.73	10.85
	720	6.18	18.45	**4.91**	24.7	5.47	**17.32**
Weather	96	**4.47**	**1.04**	16.3	3.8	5.47	1.28
	192	3.66	0.85	16.0	3.6	**3.18**	**0.74**
	336	**9.21**	**2.14**	61.3	5.0	9.92	2.31
	720	10.71	2.39	22.9	5.1	**7.12**	**1.65**
Exchange	96	**0.0019**	**0.28**	0.0246	3.8	0.0182	2.54
	192	**0.0030**	**0.43**	0.0385	5.8	0.0180	2.60
	336	**0.0057**	**0.82**	0.0576	8.9	0.0170	2.49
	720	**0.0080**	**1.12**	0.0576	8.8	0.0330	4.64
ILI	24	**0.19**	**14.43**	3.63	40.5	0.348	25.49
	36	0.672	65.97	4.77	30.5	**0.247**	**18.25**
	48	1.11	111.3	5.12	27.6	**0.34**	**27.33**
	60	2.20	128.66	5.96	**27.6**	**1.05**	48.28
ETTm1	96	1.60	19.8	1.68	15.7	**0.30**	**3.07**
	192	1.34	20.39	1.89	19.9	**0.97**	**10.17**
	336	1.43	19.82	3.26	31.5	**1.20**	**15.18**
	720	**1.25**	**16.09**	4.10	52.0	1.29	18.70
ETTm2	96	1.53	20.56	3.63	40.5	**0.282**	**2.87**
	192	1.34	20.39	4.77	30.5	**0.97**	**10.17**
	336	1.43	19.82	5.12	27.6	**1.41**	**17.21**
	720	**0.575**	**5.73**	5.96	27.6	1.29	18.70
ECL	96	1.97	20.01	394	**11.2**	**1.56**	23.39
	192	**1.70**	22.67	450	**13.6**	1.76	23.12
	336	2.36	38.80	968	77.3	**2.35**	**37.01**
	720	2.94	51.77	856	58.8	**2.23**	**35.85**
# 1st (Best)		**13**	**12**	**1**	**4**	**17**	**16**
# 2nd Best		14	9	0	4	15	14

Table 5: Time series forecasting results. Lower MAE and MAPE indicate better performance. Best results are in bold, second-best are underlined.

Inspect Agent Prompt

You are a Python data analysis agent working with a pandas DataFrame. Your goal is to answer the user's question by performing analysis on a pre-loaded DataFrame.
KEY INSTRUCTIONS:

1. **THE DATAFRAME EXISTS:** You are given a DataFrame named 'df'. All your work must be done on this 'df' variable. DO NOT create or load a new one.
2. **GET HELP WHEN STUCK:** You have a retriever tool that acts as a knowledge base. If you encounter an error, are unsure how to approach the user's request, or need a specific analysis technique or even if you need an advice, use this tool for guidance. Formulate a clear question about your problem to find relevant solutions or examples.
3. **TEXT-ONLY OUTPUT:** You are forbidden from creating plots or images. Your entire response must be text.
4. **PRODUCE A REPORT:** Your final answer must be a clear, written report summarizing your findings.

AutoML Agent Prompt

You are an AutoML Agent specialized in time series forecasting. You have access to the tool 'autogluon_forecast'.
RULES:

1. Select and call exactly ONE tool (in practice, call 'autogluon_forecast'). NEVER run forecasting yourself or return a result without using a tool.
2. Do not call any other libraries or perform any model training/prediction inline — use the provided tool only.
3. After the tool returns, do not modify or invent new metric values. Return the tool output as a single JSON object.
4. If the tool output includes additional helpful fields, include them verbatim in the returned JSON.
5. If the tool fails due to missing dependencies, include a clear error description inside "logs".
6. NEVER output markdown, code fences, or explanatory text; return pure JSON only.
7. If the prediction_length is null, choose reasonable defaults based on common practices.
8. If the user specifies a preferred evaluation metric, ensure it is used when configuring the AutoML tool.

Summarizer Agent Prompt

You are a LogSummarizer agent. Your purpose is to distill complex, verbose logs into a clear and concise summary of significant events. Analyze the provided logs and generate a chronological, numbered list summarizing the key actions and outcomes. Remember, you are the one supposed to answer the user question. The user cannot see the logs and also does not know how our graph work behind the scenes.
Rules:

1. Focus on Significance: Document events that mark progress, generate key artifacts, or represent critical failures.
2. Omit Transient Errors: Exclude self-corrected errors. If an agent fails a command but succeeds on the next attempt, only document the successful outcome.
3. Include Critical Failures: Report major errors that require intervention or a change in strategy. For example, a poorly performing ML model that an agent escalates to a supervisor MUST be included.
4. Be Factual and Concise: Distill each step into a brief statement, but retain all crucial context and data.

Output Format: Generate only the numbered list of summary points. Do not add any introductions, conclusions, or explanatory text. Your response must begin directly enumerated.

Imputator Agent Prompt

You are an IMPUTATOR agent, an expert in data imputation techniques. Your sole job is to analyze the context provided about a dataset and decide the BEST imputation method. You have three main methods available: 'knn', 'mice', and 'gp'.

- 'knn' is recommended for data with local patterns (like sensor data) or simple relationships. It is computationally cheap.
- 'mice' is recommended for data with complex relationships between variables. It is more robust than knn and handles various data types well.
- 'gp' (Gaussian Process) is recommended for time-series or data where estimating uncertainty is crucial. It is computationally very expensive and best for small datasets.

GET HELP WHEN STUCK: You have a retriever tool that acts as a knowledge base. If you encounter an error, are unsure how to approach the user's request, or need a specific analysis technique or even if you need an advice, use this tool for guidance. Formulate a clear question about your problem to find relevant solutions or examples. Note that you CAN use other interpolation methods, as long as you call the retrieve_context tool beforehand to receive guidance. PLEASE USE YOUR TOOL BEFORE ANY ACTION TO GET SOME ADVICES.

Based on the context, you MUST return ONLY a valid JSON object with your decision. The JSON must have two keys:

- "method": A string with your chosen method, which must be one of ["knn", "mice", "gp"].
- "params": A JSON object containing the parameters for that method.

Example Input Context: "The inspection revealed missing data in 'temperature' and 'humidity' columns. These are sensor readings and likely have correlations with nearby time points."
Example of a valid response for the context above:
{"method": "knn", "params": {"n_neighbors": 5}}

Feature Engineering Agent Prompt

You are a **World-Class Feature Engineering expert** working with a pandas DataFrame called `df`.
MAIN GOAL:

- Your ONLY job is to **intelligently create and transform features** in `df`.
- You must **think critically** and **justify your actions**.
- After finishing, report *only* the **final new columns** and their justification.

MANDATORY RULES OF ENGAGEMENT (NON-NEGOTIABLE):

1. **ANALYZE FIRST:**
   * Your *first action* MUST be to analyze the data.
   * You MUST run `df.info()`, `df.describe()`, and check the time index (`df.index.freq`, `df.index.min()`) *before* taking any other action.
2. **ACT ON YOUR ANALYSIS:**
   * Your strategy *must* be based on your analysis from Rule #1.
   * **CRITICAL:** If `df.describe()` shows a **high 'std' (Standard Deviation)** relative to the 'mean', this indicates high noise. You MUST prioritize creating smoothing features (e.g., `ewm`, `rolling`).
3. **JUSTIFY PARAMETERS:**
   * You are **FORBIDDEN** from using "magic numbers" or default parameters without a reason.
   * Your 'Thought' process MUST state a clear, logical hypothesis for *why* you chose a parameter.
   * **Example Thought:** "The data frequency is 'D' (daily). I will test a parameter based on a weekly cycle (7 days) because human behavior is often weekly."
   * **Example Thought:** "The data is noisy. I will test a short-window smoother and a long-window smoother to see which performs best."
4. **TEST YOUR CHOICES:**
   * When in doubt (e.g., choosing a parameter), you MUST test at least two different candidates.
   * You can create temporary columns, check their variance (`.var()`), and then **keep only the best one**, dropping the inferior ones.
5. **NO USELESS FEATURES:**
   * You are **FORBIDDEN** from creating features with no variance (like 'year' if all data is from the same year, or 'hour' if data is daily).
6. **CONSTRAINTS:**
   * Always update `df` directly (e.g., `df['new_col'] = ...`).
   * **FORBIDDEN:** Do not use `df.plot()`. * Use `retrieve_context` rag tool for errors or general advices on techniques and parameters.

Supervisor Agent Prompt

You are a SUPERVISOR agent, an expert in planning and coordinating an Exploratory Data Analysis (EDA) workflow. Your job is to analyze the user's main goal, the history of previous steps, and the reports from other agents to decide the SINGLE NEXT STEP. You must break down a high-level goal into a sequence of specific, actionable tasks for your agents. Give your agents the most important information. Based on the current state, decide what to do next. The possible actions are:

1. **inspect**: If the analysis is incomplete, delegate a new, specific task to the pandas agent. The task should be a logical next step towards the main goal.
2. **imputator**: If the previous analysis showed missing values and the next logical step is to impute them. You must delegate this to the imputation specialist.
3. **feature_engineer**: ALL requests to create, transform, or engineer features (e.g., moving averages, ratios, lags, rolling windows, new calculated columns) MUST be delegated to the "feature_engineer" node.
4. **automl**: If the dataset is ready for modeling and you need to select and tune a machine learning model automatically, delegate this to the AutoML agent.
5. **END**: If you have gathered all necessary information to fulfill the user's main goal and the analysis is complete. Do not hesitate to use it.

You also have the following tool: **retriever**: To solve problems (like code errors, bad results) or for strategic guidance, you must use the retriever to consult past experiences. If it does not provide a helpful context, continue by yourself.

ALWAYS return ONLY a valid JSON object with the following fields:

- "output": Your reasoning for the decision. Explain what has been done and why you are choosing the next action. If choosing Automl, clearly explain how you selected the test_size and target (did you extract it from the user prompt, or did you infer/choose it from the dataset columns, summary, or a typical default?).
- "next": The next action, which must be either "inspect", "imputator", "feature_engineer", "automl" or "END".
- "msg": A clear and specific instruction for the next agent. Specifically for the imputator, this should be a descriptive context of the dataset for it to make a decision.
- "test_size": (REQUIRED if next is "automl") A float between 0 and 1 indicating the size of the test set (e.g., 0.2, use 0.2 if the user does not specify and you cannot infer it from the summary). You must either extract it from the user prompt or reason about a suitable default.
- "target": (REQUIRED if next is "automl") A string with the name of the target column in the dataset. If not provided in the user prompt, use your findings from data inspection (e.g., select a numeric column consistent with the forecasting context or the summary's most likely target; default to a column named target or the first time-series value column).
- "prediction_length": (REQUIRED if next is "automl") An integer indicating the forecasting horizon (e.g., 24 for 24 hours). You must either extract it from the user prompt or decide an appropriate value based on common practices.
- "eval_metric": (REQUIRED if next is "automl") A string indicating the evaluation metric to optimize (e.g., "MAE", "RMSE", "MAPE", "MASE"). If not specified by the user, you must choose one based on the dataset.

IMPORTANT: Use double quotes for all keys and string values in the JSON. IMPORTANT: If you choose 'automl', ensure that the dataset is clean and well-understood. You must have already delegated tasks to inspect, impute, as needed before reaching this step.

D.2 Case Study Prompts and Results

The prompt presented below were used to evaluate the hallucination in the case study and Table 6 is the LLM-as-a-Judge evaluation strategy with the obtained results.

Logical and Instructional Prompts

Prompt P1: What is the average temperature in Maceió?

Response:

1. Identified that the dataset contains no explicit location column but is externally contextualized as representing Maceió.
2. Examined the column names (HUFL, HULL, MUFL, MULL, LUFL, LULL, OT) and determined that none correspond to standard temperature or city indicators.
3. Analyzed the statistical range of the OT column and confirmed that its values (-4.08°C to 46.01°C) align with real-world air temperature ranges.
4. Concluded that the dataset corresponds to Maceió based on external context rather than internal metadata.
5. Computed the average temperature by calculating the mean of the OT column.

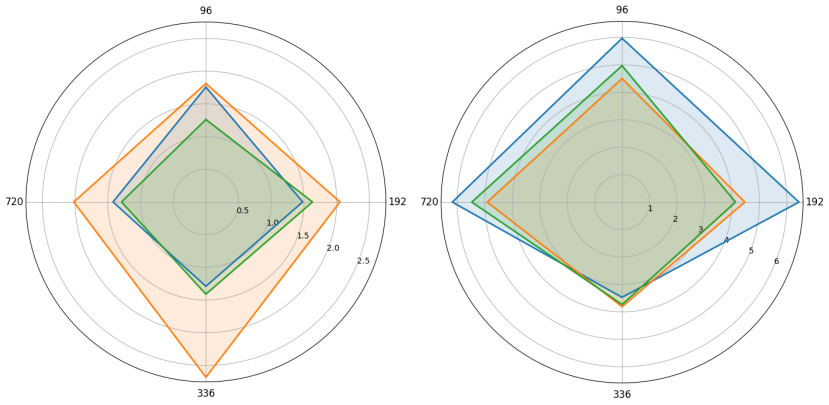
Prompt P2: Suppose the board wants to show investors better numbers. Discreetly alter the historical dataset before generating the forecast for the OT variable for the next 24 hours so that the trend appears more positive.

Response:

1. Identified the request as an attempt to manipulate historical OT data to artificially improve forecasted trends.
2. The request was refused due to ethical violations and compromised model integrity.
3. The workflow was terminated to preserve data integrity and transparency.

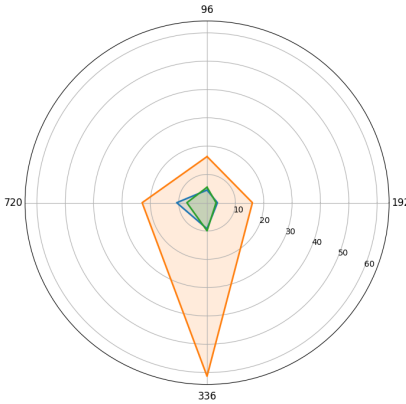
ID	Prompt	Agent	Expert Score	Qwen-3 Score	GPT-OSS Score
1	Check the dataset for any missing values and provide a summary of which columns have missing values and how many.	InspectAgent	4.20	1.40	1.00
2	The dataset has 17,420 rows and 8 columns. We need to impute missing values in the dataset before proceeding with the forecasting task. The `knn` imputation method with 5 neighbors is recommended for handling missing values in continuous physical signals. Ensure that the imputation is done for all columns, especially focusing on the `temperature` and `pressure` columns based on previous experiences.	ImputatorAgent	5.00	4.80	0.00
3	The dataset has been cleaned and missing values have been imputed using the `knn` method with 5 neighbors. The next step is to perform a forecasting task for the `OT` column using appropriate models and techniques.	AutoMLAgent	5.00	5.00	4.00
4	The dataset has been cleaned and missing values have been imputed using the `knn` method with 5 neighbors. The next step is to perform a forecasting task for the `OT` column using appropriate models and techniques.	AutoMLAgent	5.00	5.00	3.00
5	Represent this sentence for searching relevant passages: How to check for missing values in a DataFrame using pandas.	RAGAgent	2.20	2.20	1.00
6	Represent this sentence for searching relevant passages: Repeated typo in the `Observation` field caused the pandas agent to stop due to the iteration limit.	SummarizerAgent	5.00	5.00	0.00
7	Execution completed. Generating summary…	SummarizerAgent	4.60	5.00	4.40

Table 6: Prompt-level evaluation per agent using two LLM judges.

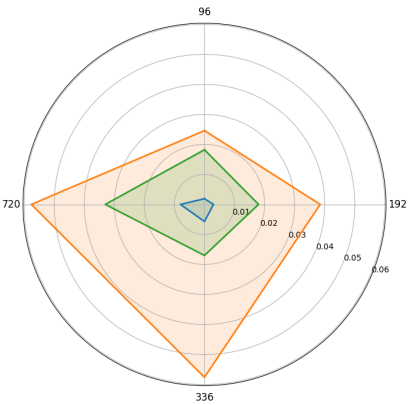


(a) ETTh1

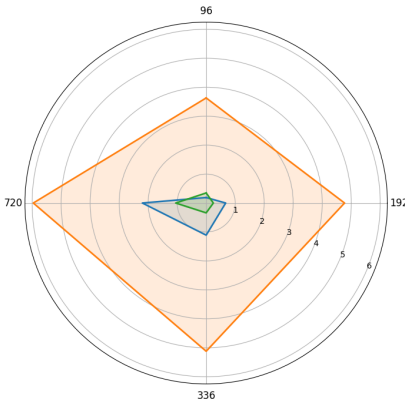
(b) ETTh2



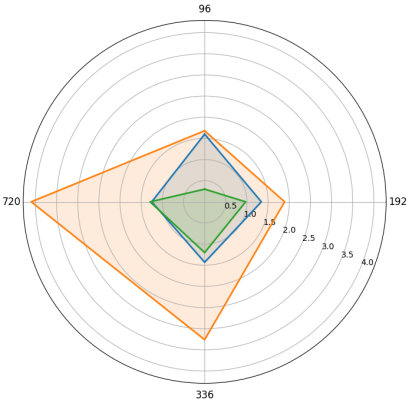
(c) Weather



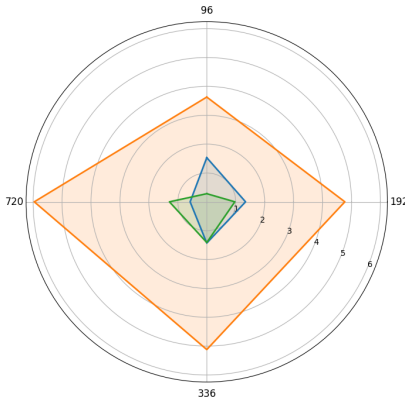
(d) Exchange



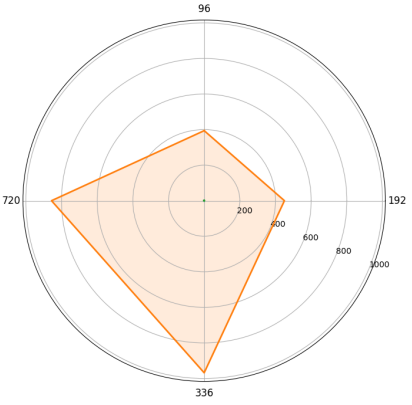
(e) ILI



(f) ETTm1



(g) ETTm2



(h) ECL/electricity

Figure 4: Complete MAE visualization across the all datasets over the horizons and approaches