When LLM Meets Time Series: Can LLMs Perform Multistep Time Series Reasoning and Inference

Anonymous Author(s)
Affiliation
Address
email

Abstract

The rapid advancement of Large Language Models (LLMs) has sparked growing interest in their application to time series analysis tasks. However, their ability to perform complex reasoning over temporal data application domains remains significantly underexplored. To achieve this goal, one first step is to establish a rigorous benchmark dataset for evaluation. In this work, we introduce TSAIA Benchmark, a first attempt to evaluate LLMs as a time series artificial intelligence assistant. To ensure both scientific rigor and practical relevance, we surveyed over 20 academic publications and identified 33 real world task formulations. The benchmark encompasses a broad spectrum of challenges, ranging from constraint aware forecasting to anomaly detection with threshold calibration, tasks that require compositional reasoning and multistep time series analysis. The question generator is designed to be dynamic and extensible, supporting continuous expansion as new datasets or task types are introduced. Given the heterogeneous nature of the tasks, we adopt task specific success criteria and tailored inference quality metrics to ensure meaningful evaluation for each task. We apply this benchmark to assess eight state of the art LLMs under a unified evaluation protocol. Our analysis reveals limitations in current models' ability to assemble complex time series analysis workflows, underscoring the need for specialized methodologies for adaptation toward domain specific applications. Our benchmark and code are publicly available online.

1 Introduction

Time series analysis is a core competency for data analysts and scientists across critical domains such as energy [1], finance [2], climate science [3], and healthcare [4]. Real-world time series workflows are inherently complex [5, 6]: they require multi-step reasoning [7], precise numerical computation [8], integration of domain knowledge [9], and adherence to operational constraints [10]. With the advent of powerful large language models (LLMs) demonstrating broad capabilities in language understanding [11], code generation [12], and scientific reasoning [13], a natural question arises: *Can these models act as time series "assistants" that follow natural language instructions and perform such complex workflows?* Answering this question requires rigorous benchmarks that capture the reasoning, computation, and decision-making challenges of time series analysis.

Existing time-series benchmarks fall into three categories, yet all miss essential ingredients for evaluating a general-purpose time-series AI assistant. Pure temporal reasoning benchmarks such as Test of Time [14] and TRAM [15] probe ordering, duration, and arithmetic but contain no time-series data (TS involved:✗), leaving numerical signal processing untested. Single-task static benchmarks like TSI-Bench [16], TSB-AD [17], GIFT-Eval [18], TFB [19], Time-MMD [20], CiK [21], and TGTSF [22] evaluate one narrowly defined task (e.g., imputation, anomaly detection,

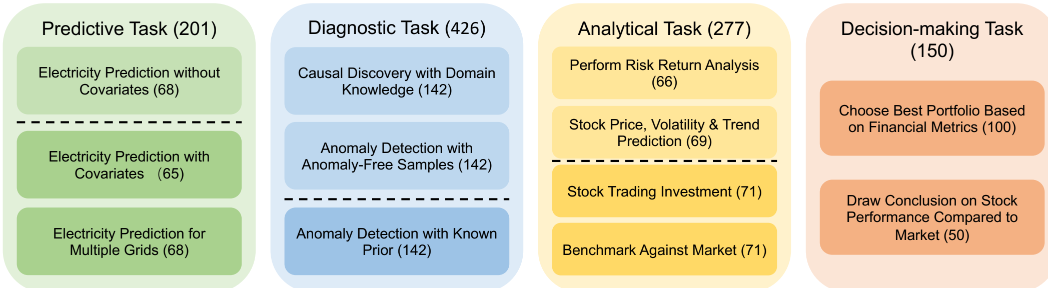

Figure 1: Categorization of Tasks in TSAIA. Lighter colors denote tasks with less difficulty and darker colors denote tasks with higher difficulty.

forecasting) under rigid settings (Dynamic:✗, Reasoning:✗, Tasks=1). More recent hybrid efforts like MTBench [23], CiK [21], and ChatTime [24] involve both time-series and text (TS involved:✓) and require reasoning (Reasoning:✓), but still adopt fixed settings (Dynamic:✗) and focus almost entirely on context-aided forecasting. Motivated by recent calls for building general time-series assistants [25], we propose a benchmark that goes beyond these limitations by supporting dynamic task settings, incorporating natural language instructions and auxiliary context, and covering diverse tasks beyond forecasting—laying the groundwork for evaluating models as adaptive, compositional reasoners over heterogeneous, real-world time-series scenarios.

In this work, we introduce the **T**ime **S**eries **A**rtificial **I**ntelligence **A**ssistant **TSAIA** Benchmark, designed for practical relevance, dynamic extensibility, and unified evaluation. It spans 33 task types distilled from existing literatures; and 1054 questions covering predictive, diagnostic, analytical, and decision making tasks. Solving TSAIA demands compositional and comparative reasoning, commonsense and decision oriented judgment, and numerical precision [26, 27]. We evaluate eight state of the art models: GPT-4o, Qwen2.5-Max, Llama-3.1 Instruct 70B, Claude-3.5 Sonnet, DeepSeek, Gemini-2.0, Codestral, and DeepSeek-R, using the CodeAct framework [28]. Agents generate executable Python code with iterative refinement, mitigating premature output[29] and numeric tokenization issues [30]. While some models excel on narrow tasks, none generalize across the full benchmark; common failures include imprecise numerics or trivial predictions and difficulty assembling complex pipelines. These results highlight the challenge of structured numerical reasoning in real world time series applications and position TSAIA as a critical benchmark for progress.

2 TSAIA Benchmark

To evaluate time series AI assistants effectively, we focus on tasks grounded in real-world use cases that data analyst in different time series application domains may face. By surveying existing literature on time series applications, we collected real world tasks that exihbit multi-step complexity with auxiliary context such as operational constraints or domain knowledge and requires precise numerical analysis and reasoning. Such problem definitions are then converted to natural language task instructions with task settings to be dynamically filled. As shown in Figure 1, tasks fall into four groups: **(1) Predictive:** forecasting with or without covariates under constraints (minimum or maximum limits [10, 31], ramp rates [32, 33], variability thresholds [6]); **(2) Diagnostic:** anomaly detection (with reference samples or known anomaly rate [5, 34, 35, 36]) and causal discovery with domain priors (e.g., partial causal ratios [37]); **(3) Analytical:** risk and return analysis and trading strategy [38, 39, 40, 41]; **(4) Decision Making:** multiple choice questions requiring comparison of portfolios or stock vs. market indices [42, 43], testing computation and reasoning [44]. TSAIA draws from public data repositories containing: grid load, solar or wind power with weather covariates [45][1], building energy usage [46][1], ERA5 climate variables [47][1], MIT-BIH ECG signals [48][†], and stock indices or prices [49][‡].

We use a modular, programmatic pipeline (Figure 2) to generate specific task instances: **(1) Task type**: select from a predefined library; **(2) Data**: sample a CSV time series dataset; **(3) Context**: ran-

[1]CC BY 4.0 licence; [†] Open Data Commons Attribution 1.0 licence. [‡] Yahoo Finance Terms of Service: personal, non commercial use only.

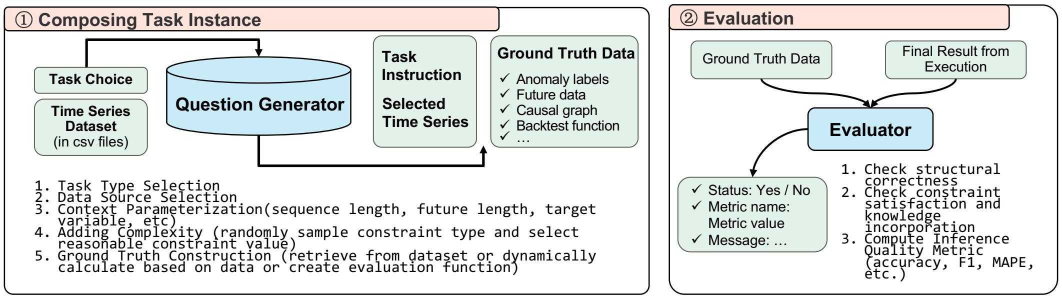

Figure 2: The proposed pipeline for multistep time series inference task instance generation and evaluation protocol.

domize parameters (input length, horizon, target, covariates) and fill a natural language template; **(4) Complexity**: inject domain constraints or auxiliary knowledge from the template; **(5) Ground truth**: retrieve from data, compute via formulas, or call task specific evaluators, ensuring an executable reference for automatic scoring. This task-generation pipeline enables TSAIA to grow dynamically: new instances can be synthesized from existing datasets by sampling different horizons or operational constraints, and additional datasets and task types can be incorporated to broaden coverage. As a result, the current release represents a cost-controlled snapshot rather than a fixed benchmark size.

Ground truth data is either directly retrieved from the underlying dataset (future targets, anomaly labels), or computed from data (e.g., risk and return metrics), or defined by an evaluation routine (e.g., backtesting trading signals to calculate cumulative or annualized return and maximum drawdown [50, 51, 52]). For multiple-choice questions, the ground-truth option is determined by selecting the portfolio or market relation that optimizes the metric specified in the question (e.g., Sharpe ratio, VaR [53, 54]). Answer positions are uniformly randomized to prevent bias in random guess. To ensure meaningful assessment, we adopt task-specific evaluation criteria that go beyond surface-level correctness. Metrics are tailored for each task, and model outputs must satisfy the given constraints and incorporate the provided knowledge. Trivial or degenerate solutions (e.g., low-quality forecasts or all-zero anomaly labels) are considered failures even if they are well-formatted. The task-specific success criteria and quality metrics, chosen to reflect practical utility, are summarized in Table 2.

3 Experiments

Benchmark		GPT-4o	Qwen-Max	Llama3.1	Claude-3.5	DeepSeek	Gemini-2.0	Codestral	DeepSeek-R
Electricity Prediction with Covariates	Success Rate	0.55	0.78	0.51	0.72	0.86	0.14	0.43	0.96
	MAPE (std)	0.14 (0.17)	0.11 (0.11)	0.11 (0.10)	0.12 (0.14)	0.13 (0.16)	0.17 (0.11)	0.07 (0.06)	0.11 (0.11)
Electricity Prediction without Covariates	Success Rate	0.88	0.94	0.86	0.84	0.85	0.34	0.82	0.75
	MAPE (std)	0.18 (0.14)	0.15 (0.12)	0.17 (0.10)	0.19 (0.12)	0.16 (0.15)	0.26 (0.18)	0.17 (0.12)	0.22 (0.20)
Electricity Prediction for Multiple Grids	Success Rate	0.65	0.75	0.52	0.77	0.85	0.32	0.25	0.83
	MAPE (std)	0.16 (0.19)	0.19 (0.23)	0.61 (0.34)	0.17 (0.18)	0.16 (0.17)	0.56 (0.16)	0.15 (0.15)	0.21 (0.24)
Diagnostic Task w/ Reference Samples	Success Rate	0.37	0.20	0.39	0.65	0.38	0.12	0.41	0.49
	F1 (std)	0.88 (0.18)	0.86 (0.22)	0.87 (0.19)	0.83 (0.18)	0.87 (0.19)	0.88 (0.08)	0.86 (0.20)	0.86 (0.18)
Causal Discovery w/ Domain Knowledge	Success Rate	0.89	0.81	0.91	0.99	0.96	0.42	0.94	0.97
	MAPE (std)	0.69 (0.14)	0.67 (0.14)	0.69 (0.14)	0.74 (0.12)	0.69 (0.13)	0.71 (0.15)	0.69 (0.14)	0.73 (0.13)
Anomaly Detection w/ Multiple Sequences	Success Rate	0.87	0.42	0.40	0.99	0.99	0.30	0.40	0.98
	F1 (std)	0.31 (0.17)	0.38 (0.19)	0.41 (0.08)	0.60 (0.15)	0.61 (0.15)	0.42 (0.19)	0.24 (0.19)	0.56 (0.16)
Stock Prediction	Success Rate	0.89	0.72	0.54	0.65	0.80	0.17	0.48	0.80
	MAPE (std)	0.38 (0.18)	0.44 (0.17)	0.35 (0.18)	0.51 (0.14)	0.40 (0.19)	0.56 (0.33)	0.33 (0.19)	0.41 (0.15)
Stock Prediction Trend	Success Rate	0.43	0.30	0.57	0.43	0.26	0.04	0.52	0.35
	Accuracy (std)	0.90 (0.20)	0.86 (0.23)	0.88 (0.21)	0.85 (0.23)	1.00 (0.00)	1.00 (0.00)	0.96 (0.14)	0.81 (0.24)
Risk/Return Estimation	Success Rate	0.42	0.23	0.27	0.35	0.38	0.06	0.47	0.38
	Abs Error (std)	0.01 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.01)	0.01 (0.01)	0.01 (0.00)	0.01 (0.00)	0.01 (0.01)
Benchmark Against Market Analysis	Success Rate	0.44	0.20	0.06	0.51	0.73	0.18	0.01	0.77
	Abs Error (std)	0.00 (0.00)	0.01 (0.01)	0.03 (0.01)	0.00 (0.01)	0.00 (0.00)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)
Stock Trading Strategy	Success Rate	0.44	0.59	0.96	0.62	0.61	0.18	0.63	0.52
	Cumulative Return	0.13	0.10	0.00	0.09	0.09	0.05	0.06	0.07
	Annualized Return	2.43	4.56	0.05	4.58	1.69	0.36	3.87	1.41
	Maximum Drawdown	0.05	0.05	0.00	0.04	0.05	0.02	0.02	0.04

Table 1: Model Performance on TSAIA. Red indicates best result, Blue indicates second best.

We evaluated eight LLMs: GPT-4o [55], Qwen2.5-Max [56], Llama-3.1 Instruct 70B [57], Claude-3.5 Sonnet [58], DeepSeek [59], Gemini-2.0 [60], Codestral [61], and DeepSeek-R [62]. All models use the CodeAct framework [28] via AgentScope [63] to generate Python code, execute in controlled jupyter notebook environment, receive error feedback, and lastly revise. The maximum interaction

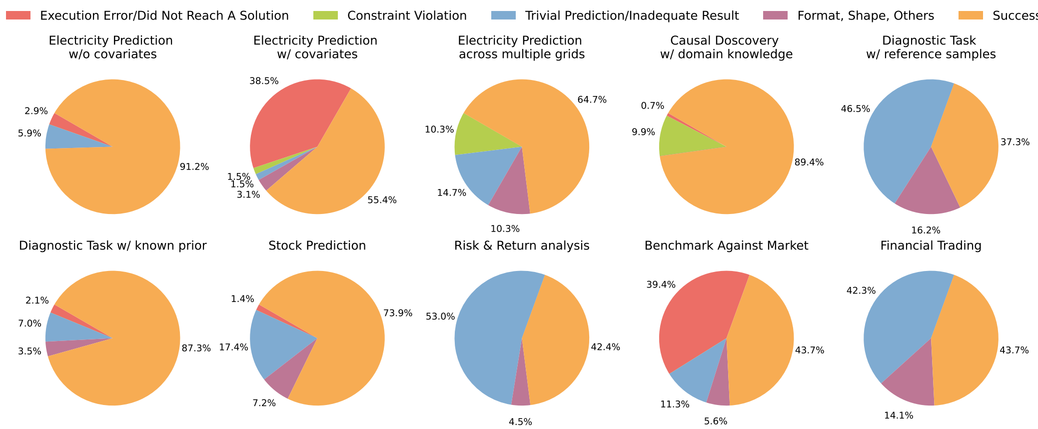

Figure 4: Case Study on GPT-4o Error Distribution across Tasks Grouped by Difficulty Level

turns is capped at five. We report success rate (the fraction of instances satisfying the predefined success criteria) and task-specific quality metrics (e.g., MAPE for forecasting, F1 for anomaly detection), computed only on successful executions by the evaluators in Figure 2.

Table 1 reports average success rates and quality metrics for the three task groups in TSAIA, with sub-task breakdowns in Appendix Tables 4,5, and 6. In predictive tasks, models achieve reasonable accuracy on single-sequence forecasting but show performance degradation on multi-grid settings, reflected by lower success rates and higher MAPE. Error decomposition (Figure 4) reveals that adding covariates and multi-series settings increases execution and constraint violation errors, suggesting difficulties in scaling reasoning to higher-dimensional inputs. In diagnostic tasks, models succeed when explicit domain knowledge or priors are available, but reference-sample calibration often collapses to trivial predictions which is consistent with the low success rates and high frequency of degenerate outputs. In analytical tasks, success rates are lowest: execution errors are frequent in market benchmarking, risk/return failures arise from unfamiliar metrics (table 6), and trading strategies are often suboptimal in backtests. Overall, reliability declines as tasks demand more multi-step reasoning, external context integration, and nuanced financial understanding, highlighting the challenge of complex multi-step time series workflows [64]. Notably, DeepSeek-R achieves the highest success rates across all groups, providing direct evidence that explicit reasoning improves performance on TSAIA tasks, though at the cost of increased token usage (Figure 3). By contrast, GPT-4o and DeepSeek-Chat achieve lower success rates but remain the most token-efficient. Additional error analysis and experimental results on multiple choice questions are shown in section B and D.

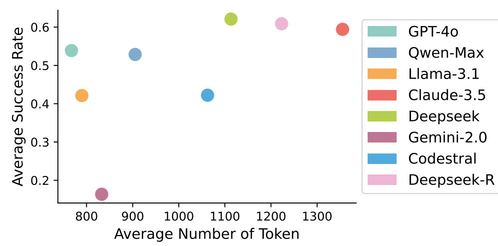

Figure 3: Average Success Rate of Models with respect to the Average Number of Tokens Used.

4 Conclusion

This paper introduces TSAIA, a first benchmark for evaluating LLMs as time series AI assistants. Covering diverse tasks, it emphasizes compositional reasoning, adherence to domain constraints, and integration of contextual knowledge in addition to the basic numerical precision demanded by traditional time series analysis. Evaluation of eight LLM agents reveals that current models are far from reliable time series assistants. They frequently fail under domain constraints, struggle with multistep workflows, and often produce trivial predictions under naive solution paths. Such gap underscores the need for hybrid approaches that combine symbolic reasoning, execution feedback, specialized tool integration, and domain alignment. TSAIA establishes a foundation for developing and systematically assessing next-generation time series inference agents.

References

[1] Francisco Martinez Alvarez, Alicia Troncoso, Jose C Riquelme, and Jesus S Aguilar Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2010.

[2] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[3] Manfred Mudelsee. Trend analysis of climate time series: A review of methods. *Earth-science reviews*, 190:310–322, 2019.

[4] Niels K Rathlev, John Chessare, Jonathan Olshaker, Dan Obendorfer, Supriya D Mehta, Todd Rothenhaus, Steven Crespo, Brendan Magauran, Kathy Davidson, Richard Shemin, et al. Time series analysis of variables associated with daily mean emergency department length of stay. *Annals of emergency medicine*, 49(3):265–271, 2007.

[5] Xudong Yan, Huaidong Zhang, Xuemiao Xu, Xiaowei Hu, and Pheng-Ann Heng. Learning semantic context from normal samples for unsupervised anomaly detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3110–3118, 2021.

[6] Kyo Beom Han, Jaesung Jung, and Byung O Kang. Real-time load variability control using energy storage system for demand-side management in south korea. *Energies*, 14(19):6292, 2021.

[7] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

[8] Kostadin Cvejoski, Ramsés J Sánchez, and César Ojeda. The future is different: Large pre-trained language models fail in prediction tasks. *arXiv preprint arXiv:2211.00384*, 2022.

[9] Zhiyi Xue, Liangguo Li, Senyue Tian, Xiaohong Chen, Pingping Li, Liangyu Chen, Tingting Jiang, and Min Zhang. Domain knowledge is all you need: A field deployment of llm-powered test case generation in fintech domain. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pages 314–315, 2024.

[10] Ke-qiu WANG, Si-guang SUN, Hong-yi WANG, Chang-xu JIANG, and Zhao-xia JING. 220kv city power grid maximum loadability determination with static security-constraints. *Power, Energy Engineering and Management (PEEM2016)*, page 1, 2016.

[11] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32, 2019.

[12] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[13] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

[14] Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating llms on temporal reasoning. *arXiv preprint arXiv:2406.09170*, 2024.

[15] Yuqing Wang and Yun Zhao. Tram: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6389–6415, 2024.

[16] Wenjie Du, Jun Wang, Linglong Qian, Yiyuan Yang, Zina Ibrahim, Fanxing Liu, Zepu Wang, Haoxin Liu, Zhiyuan Zhao, Yingjie Zhou, et al. Tsi-bench: Benchmarking time series imputation. *arXiv preprint arXiv:2406.12747*, 2024.

[17] Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[18] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

[19] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

[20] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash. Time-MMD: Multi-domain multimodal dataset for time series analysis. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[21] Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, et al. Context is key: A benchmark for forecasting with essential textual information. *arXiv preprint arXiv:2410.18959*, 2024.

[22] Zhijian Xu, Yuxuan Bian, Jianyuan Zhong, Xiangyu Wen, and Qiang Xu. Beyond trend and periodicity: Guiding time series forecasting with textual cues. *arXiv preprint arXiv:2405.13522*, 2024.

[23] Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*, 2025.

[24] Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In *AAAI Conference on Artificial Intelligence*, 2025.

[25] Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position paper: What can large language models tell us about time series analysis. *arXiv preprint arXiv:2402.02713*, 2024.

[26] Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. Understanding and patching compositional reasoning in llms. *arXiv preprint arXiv:2402.14328*, 2024.

[27] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[28] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

[29] Seoha Song, Junhyun Lee, and Hyeonmok Ko. Hansel: Output length controlling framework for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25146–25154, 2025.

[30] Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models. *Journal of the American Medical Informatics Association*, 31(9):2151–2158, 2024.

[31] Claudia Greif, Raymond B Johnson, Chao an Li, Alva J Svoboda, and K Andrijeski Uemura. Short-term scheduling of electric power systems under minimum load conditions. *IEEE transactions on power systems*, 14(1):280–286, 1999.

[32] Jonas Schaible, Bijan Nouri, Lars Höpken, Tim Kotzab, Matthias Loevenich, Niklas Blum, Annette Hammer, Jonas Stührenberg, Klaus Jäger, Christiane Becker, et al. Application of nowcasting to reduce the impact of irradiance ramps on pv power plants. *EPJ Photovoltaics*, 15:15, 2024.

[33] Spyros I Gkavanoudis, Kyriaki-Nefeli D Malamaki, Eleftherios O Kontis, Charis S Demoulias, Aditya Shekhar, Umer Mushtaq, and Sagar Bandi Venu. Provision of ramp-rate limitation as ancillary service from distribution to transmission system: Definitions and methodologies for control and sizing of central battery energy storage system. *Journal of Modern Power Systems and Clean Energy*, 11(5):1507–1518, 2023.

[34] Yi Song, Sennan Kuang, Junling Huang, and Da Zhang. Unsupervised anomaly detection of industrial building energy consumption. *Energy and Built Environment*, 2024.

[35] Zijian Niu, Ke Yu, and Xiaofei Wu. Lstm-based vae-gan for time-series anomaly detection. *Sensors*, 20(13):3738, 2020.

[36] Shikha Verma, Kuldeep Srivastava, Akhilesh Tiwari, and Shekhar Verma. Deep learning techniques in extreme weather events: A review. *arXiv preprint arXiv:2308.10995*, 2023.

[37] Raquel Aoki and Martin Ester. Parkca: Causal inference with partially known causes. In *BIOCOMPUTING 2021: Proceedings of the Pacific Symposium*, pages 196–207. World Scientific, 2020.

[38] Timothy Riley and Qing Yan. Maximum drawdown as predictor of mutual fund performance and flows. *Financial Analysts Journal*, 78(4):59–76, 2022.

[39] Cynthia Miglietti, Zdenka Kubosova, and Nicole Skulanova. Bitcoin, litecoin, and the euro: an annualized volatility analysis. *Studies in Economics and Finance*, 37(2):229–242, 2020.

[40] Talwar Shalini, Pranav Shah, and Shah Utkarsh. Picking buy-sell signals: A practitioner's perspective on key technical indicators for selected indian firms. *Studies in Business and Economics*, 14(3):205–219, 2019.

[41] Rommy Pramudya. Technical analysis to determine buying and selling signal in stock trade. *International Journal of Finance & Banking Studies (2147-4486)*, 9(1):58–67, 2020.

[42] Oliver Steinki and Ziad Mohammad. Common metrics for performance evaluation: Overview of popular performance measurement ratios. *Available at SSRN 2662054*, 2015.

[43] Francis Gupta, Robertus Prajogi, and Eric Stubbs. The information ratio and performance. *Journal of Portfolio Management*, 26(1):33, 1999.

[44] Paolo Legrenzi, Vittorio Girotto, and Philip N Johnson-Laird. Focussing in reasoning and decision making. *Cognition*, 49(1-2):37–66, 1993.

[45] Xiangtian Zheng, Nan Xu, Loc Trinh, Dongqi Wu, Tong Huang, S Sivaranjani, Yan Liu, and Le Xie. Psml: a multi-scale time-series dataset for machine learning in decarbonized energy grids. *arXiv preprint arXiv:2110.06324*, 2021.

[46] Large-scale energy anomaly detection (lead). `https://www.kaggle.com/competitions/energy-anomaly-detection/data`, 2022. Kaggle Competition.

[47] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly journal of the royal meteorological society*, 146(730):1999–2049, 2020.

[48] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[49] Yahoo Finance. Yahoo Finance, n.d. Accessed: 2025-05-14.

[50] Roger Clarke, Harindra De Silva, and Steven Thorley. Minimum-variance portfolio composition. *Journal of Portfolio Management*, 37(2):31, 2011.

[51] David G Booth and Eugene F Fama. Diversification returns and asset contributions. *Financial Analysts Journal*, 48(3):26–32, 1992.

[52] Malik Magdon-Ismail and Amir F Atiya. Maximum drawdown. *Risk Magazine*, 17(10):99–102, 2004.

[53] William F Sharpe. The sharpe ratio. *Journal of portfolio management*, 21(1):49–58, 1994.

[54] Darrell Duffie and Jun Pan. An overview of value at risk. *Journal of derivatives*, 4(3):7–49, 1997.

[55] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[56] Qwen Team. Qwen2.5-max: Exploring the intelligence of large-scale moe model, January 2025. Accessed: 2025-05-14.

[57] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[58] Anthropic. Claude 3.5 sonnet, June 2024. Accessed: 2025-05-14.

[59] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[60] Google DeepMind. Introducing gemini 2.0: Our new ai model for the agentic era, December 2024. Accessed: 2025-05-14.

[61] Mistral AI. Codestral, May 2024. Accessed: 2025-05-14.

[62] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[63] Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Ze Yu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Agentscope: A flexible yet robust multi-agent platform. *CoRR*, abs/2402.14034, 2024.

[64] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332, 2023.

[65] Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *EMNLP (Findings)*, 2024.

[66] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*, 2023.

A Extended Benchmark Details

A.1 Question Generator: How to generate specific task instances?

I have historical Temperature, Relative Humidity, Wind Speed data and the corresponding load_power data for the past 117 minutes. I need to ensure that the maximum allowable system load does not exceed 1.0689227278350713 MW. Think about how Temperature, Relative Humidity, Wind Speed influence load_power. Please give me a forecast for the next 12 minutes for load_power. Your goal is to make the most accurate forecast as possible, refine prediction result based on the constraint previously described, and …

(a) Task Instruction

Time	Tempera-ture	Relative Humidity	Wind Speed	Load power
2020-09-13 09:44	24.58	89.41	1.4	0.923
2020-09-13 09:45	24.60	89.31	1.4	0.924
…	…	…	…	…
2020-09-13 11:39	25.40	81.87	1.4	1.003
2020-09-13 11:40	25.40	81.87	1.4	1.004

(b) Serialized Dataset

1.0051
1.0057
1.0062
1.0068
1.0073
1.0079
1.0084
1.0090
1.0095
1.0101
1.0106
1.0112

(c) Ground Truth

Figure 5: Example Task Instance containing the task instruction, accompanied serialized dataset, and ground truth.

As shown in figure 5, each task instance contains a natural language instruction paired with structured time series inputs and corresponding ground truth data. By design, the benchmark framework is extensible and dynamic. New task instances can be generated automatically by applying the same pipeline to additional time series data sources when accompanied by its designated task template, supporting ongoing evaluation and adaptation to new domains. This supports long-term benchmarking efforts and enables ongoing expansion across domains.

A.2 Evaluation: How to perform evaluation on heterogeneous task instances?

Task Type	Success Criterion	Metrics
Constrained Forecasting	Prediction is of correct shape and satisfies the specified operational constraint and the prediction is non-trivial (MAPE<1)	MAPE
Anomaly Detection w/ reference samples	A binary sequence with correct length is obtained and the prediction is non-trivial (F1-score>0)	F1-score
Causal Inference w/ domain knowledge	A binary causal matrix with correct shape is returned. The provided domain knowledge is incorporated	Accuracy
Financial Analytics	A scalar value is returned and the prediction is non-trivial (absolute error<0.05)	Absolute Error
Financial Trading	An investment signal of correct length is returned and there is no loss in investment	CR, AR, MDD

Table 2: Task-specific success criteria and inference quality metrics. CR denotes Cumulative Return, AR denotes Annualized Return, MDD denotes Maximum Drawdown.

The evaluation process follows a three-stage protocol. In the first stage, outputs are validated for structural correctness, shape conformity. The second stage checks against specified constraint and domain knowledge incorporation. Lastly, the inference quality metric is computed relative to ground truth data. Results are returned in a structured format, including the success status, diagnostic messages, and detailed metric scores. Failures are categorized into execution errors (model output fails to run or parse), constraint violations (outputs violate injected domain-specific rules), and low quality outputs (predictions meet format expectations but fall short on metric thresholds). Multiple choice questions have a simple evaluation procedure of checking against ground truth letter option.

A.3 Comparison with Existing Benchmarks

As shown in table 3, existing time-series benchmarks fall into three main groups, each lacking one or more component for evaluating time series AI assistant: First, datasets such as Test of Time [14] and TRAM [15] present pure QA tasks (ordering, duration, arithmetic) with no time-series included (TS

Benchmark	Dynamic	TS involved	Reasoning	#Tasks	Task Type
Test of Time [14]	✗	✗	✓	1	QA
TRAM [15]	✗	✗	✓	1	QA
TSI-Bench [16]	✗	✓	✗	1	TS Analysis
TSB-AD [17]	✗	✓	✗	1	TS Analysis
GIFT-Eval [18]	✗	✓	✗	1	TS Analysis
TFB [19]	✗	✓	✗	1	TS Analysis
Time-MMD [20]	✗	✓	✗	1	TS Analysis
CiK [21]	✗	✓	✗	1	TS Analysis
TGTSF [22]	✗	✓	✗	1	TS Analysis
LLM TS Struggle [65]	✗	✓	✓	2	QA, TS Analysis
MTBench [23]	✗	✓	✓	3	QA, TS Analysis
ChatTime [24]	✗	✓	✓	3	QA, TS Analysis
TSAIA(Ours)	✓	✓	✓	**4**	QA, TS Analysis

Table 3: Comparison of TSAIA and existing temporal-related benchmarks. Dynamic indicates whether new task instances can be continuously generated.

involved:✗). While they evaluate logical reasoning, they cannot test how models process numerical signals. Secondly, benchmarks like TSI-Bench [16], TSB-AD [17], GIFT-Eval [18], TFB [19], Time-MMD [20], CiK [21], and TGTSF [22] focus on a single static time-series analysis including imputation, anomaly detection, and forecasting over fixed datasets with pre-defined sliding window size for evaluation (Dynamic:✗, Reasoning:✗, Tasks=1). Lastly, recent efforts on hybrid QA and analysis such as MTBench [23], ChatTime [24] combine time-series and text inputs (TS involved:✓) and include reasoning components (Reasoning:✓), yet remain fixed setting (Dynamic:✗) and only covers context-aided forecasting in time series analysis component part. TSAIA arises as first of its kind time series inference benchmark with practical relevance, task diversity, and supports continuous expansion.

B Additional Result

Models are evaluated with hyperparameter $top\_p = 1$ and $temperature = 0$ to minimize sampling randomness. In predictive tasks (table 4), models handle simple constraints (maximum or minimum load) better than temporal smoothness (ramp or variability). In diagnostic tasks (table 5), models struggle more with leveraging reference samples to calibration anomaly threshold. In analytical tasks (table 6), price or volatility prediction is moderate to strong; trend prediction lags. In risk and return analysis, models favor familiar or simpler metrics (e.g., volatility, Sharpe). In decision making tasks, most models hover near chance (Figure 6); DeepSeek-R achieves consistent above random accuracy but with higher token usage (Figure 3). Overall, results underline the value of domain specialization [66]. Harder tasks elicit more interaction turns under CodeAct (Figure 7).

C Dataset Statistics

Table 7 summarizes the dataset statistics for the raw time series datasets used in TSAIA. The climate data is obtained from ERA5 dataset [2]. Energy data with covariates is obtained from[3]. The ECG signal data is obtained from PhysioNet[4][5]. The building energy usage data is obtained from Kaggle[6]. Notably, the daily stock data, hourly stock data, and energy data with geolocation were manually scraped and preprocessed. The energy data with geolocation was obtained from official energy grid operator

[2] https://climatelearn.readthedocs.io/en/latest/user-guide/tasks_and_datasets.html#era5-dataset

[3] https://github.com/tamu-engineering-research/Open-source-power-dataset

[4] https://physionet.org/content/nsrdb/1.0.0/

[5] https://physionet.org/content/ltdb/1.0.0/

[6] https://www.kaggle.com/competitions/energy-anomaly-detection/data

Metric		GPT-4o	Qwen-Max	Llama3.1	Claude-3.5	DeepSeek	Gemini-2.0	Codestral	DeepSeek-R
Electricity Prediction with Covariates									
Max Load	Success Rate	0.50	0.75	0.56	0.88	1.00	0.19	0.31	0.94
	MAPE (std)	0.09 (0.12)	0.07 (0.06)	0.10 (0.11)	0.11 (0.12)	0.10 (0.12)	0.52 (0.41)	0.03 (0.03)	0.10 (0.11)
Min Load	Success Rate	0.76	0.82	0.65	0.82	0.88	0.18	0.59	1.00
	MAPE (std)	0.11 (0.11)	0.09 (0.11)	0.10 (0.11)	0.09 (0.11)	0.12 (0.18)	0.09 (0.04)	0.09 (0.10)	0.09 (0.11)
Load Ramp Rate	Success Rate	0.46	0.80	0.53	0.93	0.80	0.13	0.47	0.93
	MAPE (std)	0.18 (0.14)	0.14 (0.12)	0.15 (0.07)	0.19 (0.19)	0.11 (0.08)	0.04 (0.01)	0.12 (0.08)	0.11 (0.07)
Load Variability	Success Rate	0.47	0.76	0.29	0.29	0.76	0.06	0.35	0.94
	MAPE (std)	0.20 (0.31)	0.13 (0.16)	0.09 (0.12)	0.09 (0.12)	0.19 (0.27)	0.05 (0.00)	0.04 (0.03)	0.11 (0.14)
Electricity Prediction without Covariates									
Max Load	Success Rate	1.00	0.94	0.94	1.00	0.94	0.41	1.00	0.71
	MAPE (std)	0.18 (0.16)	0.10 (0.07)	0.16 (0.10)	0.15 (0.13)	0.15 (0.12)	0.10 (0.02)	0.12 (0.07)	0.23 (0.26)
Min Load	Success Rate	0.94	0.94	0.94	0.94	0.88	0.29	0.71	0.88
	MAPE (std)	0.14 (0.08)	0.14 (0.08)	0.17 (0.08)	0.17 (0.09)	0.13 (0.09)	0.12 (0.03)	0.14 (0.05)	0.17 (0.16)
Load Ramp Rate	Success Rate	0.76	1.00	0.71	0.76	0.82	0.24	0.88	0.76
	MAPE (std)	0.24 (0.19)	0.23 (0.22)	0.21 (0.11)	0.28 (0.16)	0.19 (0.20)	0.42 (0.28)	0.29 (0.30)	0.22 (0.13)
Load Variability	Success Rate	0.82	0.88	0.82	0.65	0.76	0.41	0.71	0.65
	MAPE (std)	0.17 (0.12)	0.13 (0.09)	0.15 (0.09)	0.16 (0.12)	0.19 (0.17)	0.39 (0.39)	0.13 (0.07)	0.24 (0.24)
Electricity Prediction for Multiple Grids									
Max Load	Success Rate	0.76	0.88	0.47	0.88	0.94	0.47	0.12	0.88
	MAPE (std)	0.21 (0.27)	0.21 (0.24)	0.64 (0.31)	0.18 (0.20)	0.16 (0.21)	0.34 (0.39)	0.10 (0.03)	0.23 (0.27)
Min Load	Success Rate	0.76	0.88	0.24	0.94	0.94	0.18	0.29	0.94
	MAPE (std)	0.10 (0.12)	0.18 (0.29)	0.46 (0.37)	0.13 (0.20)	0.08 (0.11)	0.01 (0.01)	0.23 (0.37)	0.16 (0.23)
Load Ramp Rate	Success Rate	0.65	0.65	0.88	0.88	0.94	0.29	0.29	1.00
	MAPE (std)	0.19 (0.24)	0.18 (0.18)	0.73 (0.33)	0.27 (0.21)	0.21 (0.21)	1.00 (0.00)	0.10 (0.05)	0.19 (0.19)
Load Variability	Success Rate	0.41	0.59	0.53	0.41	0.59	0.35	0.29	0.53
	MAPE (std)	0.15 (0.13)	0.18 (0.23)	0.61 (0.36)	0.11 (0.13)	0.18 (0.14)	0.90 (0.23)	0.19 (0.13)	0.24 (0.25)

Table 4: Model Performance on Predictive Task. Red indicates best result, Blue indicates second best.

Benchmark		GPT-4o	Qwen-Max	Llama3.1	Claude-3.5	DeepSeek	Gemini-2.0	Codestral	DeepSeek-R
Diagnostic Task w/ Reference Samples									
Extreme Weather Detection w/ Reference Samples	Success Rate	0.24	0.23	0.23	0.62	0.23	0.14	0.23	0.34
	F1 (std)	0.91 (0.23)	0.90 (0.24)	0.90 (0.24)	0.90 (0.18)	0.90 (0.24)	0.96 (0.06)	0.91 (0.23)	0.91 (0.20)
ECG Signal Anomaly w/ Reference Samples	Success Rate	0.51	0.17	0.55	0.68	0.54	0.10	0.59	0.63
	F1 (std)	0.55 (0.35)	0.70 (0.29)	0.43 (0.36)	0.65 (0.32)	0.54 (0.34)	0.01 (0.00)	0.58 (0.34)	0.61 (0.32)
Causal Discovery w/ Domain Knowledge									
Causal Discovery w/ Quantitative Knowledge	Success Rate	0.94	0.92	0.99	1.00	0.97	0.39	0.94	0.96
	Accuracy (std)	0.69 (0.09)	0.77 (0.11)	0.78 (0.12)	0.77 (0.09)	0.71 (0.11)	0.42 (0.18)	0.72 (0.11)	0.77 (0.10)
Causal Discovery w/ Qualitative Knowledge	Success Rate	0.85	0.70	0.83	0.97	0.96	0.45	0.93	0.99
	Accuracy (std)	0.87 (0.17)	0.79 (0.17)	0.77 (0.18)	0.89 (0.16)	0.89 (0.14)	0.72 (0.20)	0.88 (0.15)	0.90 (0.12)
Anomaly Detection across Multiple Sequences									
Extreme Weather Detection w/ Known Anomaly Rate	Success Rate	0.87	0.31	0.03	0.97	0.97	0.37	0.23	1.00
	F1 (std)	0.53 (0.25)	0.62 (0.19)	0.68 (0.05)	0.73 (0.12)	0.72 (0.11)	0.65 (0.18)	0.42 (0.31)	0.72 (0.10)
Energy Usage Anomaly w/ Known Anomaly Rate	Success Rate	0.87	0.52	0.77	1.00	1.00	0.23	0.58	0.96
	F1 (std)	0.08 (0.09)	0.14 (0.20)	0.15 (0.11)	0.48 (0.18)	0.50 (0.19)	0.19 (0.21)	0.06 (0.06)	0.40 (0.23)

Table 5: Model Performance on Diagnostic Task. Red indicates best result, Blue indicates second best.

websites[7][8][9], and the associated geolocation was inferred as the largest city within the operational zone delineated by each provider's published grid map[10][11][12]. Stock price data was scraped using the pyfinance[13] package, with data pulled up to date as of 2024-09-17. The stock market indices data are pulled from various sources on the web. The causal discovery dataset is synthetically generated to reflect controlled causal structures. The prompt used to obtain causal discovery dataset is shown in section G.

[7] https://www.nyiso.com/load-data
[8] https://www.ercot.com/gridinfo/load/load_hist
[9] https://www.misoenergy.org/markets-and-operations/real-time--market-data/market-reports
[10] https://www.nyiso.com/documents/20142/1397960/nyca_zonemaps.pdf
[11] https://www.ercot.com/news/mediakit/maps
[12] https://www.misostates.org/images/stories/meetings/Cost_Allocation_Principles_Committee/2021/Website_Presentations.pdf
[13] https://pypi.org/project/pyfinance/

Benchmark		GPT-4o	Qwen-Max	Llama3.1	Claude-3.5	DeepSeek	Gemini-2.0	Codestral	DeepSeek-R
Stock Prediction									
Future Price	Success Rate	0.96	1.00	0.70	0.74	0.87	0.17	0.39	1.00
	MAPE (std)	0.06 (0.08)	0.05 (0.07)	0.06 (0.08)	0.12 (0.16)	0.05 (0.07)	0.28 (0.42)	0.05 (0.05)	0.05 (0.07)
Future Volatility	Success Rate	0.83	0.43	0.39	0.74	0.57	0.17	0.57	0.61
	MAPE (std)	0.70 (0.28)	0.83 (0.26)	0.64 (0.29)	0.75 (0.31)	0.90 (0.13)	0.84 (0.24)	0.61 (0.32)	0.77 (0.24)
Future Trend	Success Rate	0.43	0.30	0.57	0.26	0.43	0.04	0.52	0.35
	Accuracy (std)	0.90 (0.20)	0.86 (0.23)	0.88 (0.21)	1.00 (0.00)	0.85 (0.23)	1.00 (0.00)	0.96 (0.14)	0.81 (0.24)
Risk/Return Estimation									
Annualized Return	Success Rate	0.45	–	0.09	0.27	0.36	–	0.18	0.55
	Abs Error (std)	0.02 (0.02)	–	0.01 (0.00)	0.01 (0.01)	0.02 (0.01)	–	0.03 (0.02)	0.02 (0.02)
Annualized Volatility	Success Rate	0.91	0.82	1.00	1.00	1.00	0.09	1.00	0.91
	Abs Error (std)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.02 (0.00)	0.00 (0.00)	0.00 (0.00)
Maximum Drawdown	Success Rate	0.18	0.09	0.27	0.18	0.27	0.09	–	0.45
	Abs Error (std)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	–	0.01 (0.01)
Calmar Ratio	Success Rate	0.18	0.18	–	0.27	0.27	–	0.82	0.18
	Abs Error (std)	0.01 (0.01)	0.01 (0.01)	–	0.02 (0.01)	0.02 (0.01)	–	0.01(0.01)	0.01 (0.01)
Sortino Ratio	Success Rate	0.09	0.09	–	–	0.18	–	0.09	–
	Abs Error (std)	0.01 (0.00)	0.01 (0.00)	–	–	0.00 (0.00)	–	0.00 (0.00)	–
Sharpe Ratio	Success Rate	0.73	0.18	0.27	0.36	0.18	0.18	0.73	0.18
	Abs Error (std)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.01)	0.02 (0.02)	0.02 (0.02)	0.00 (0.00)	0.02 (0.01)
Benchmark Against Market Analysis									
Information Ratio	Success Rate	0.44	0.20	0.06	0.51	0.73	0.18	0.01	0.77
	Abs Error (std)	0.00 (0.00)	0.01 (0.01)	0.03 (0.01)	0.00 (0.01)	0.00 (0.00)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)
Stock Trading Strategy									
Trading Strategy	Success Rate	0.44	0.59	0.96	0.62	0.61	0.18	0.63	0.52
	Cumulative Return	0.13	0.10	0.00	0.09	0.09	0.05	0.06	0.07
	Annualized Return	2.43	4.56	0.05	4.58	1.69	0.36	3.87	1.41
	Maximum Drawdown	0.05	0.05	0.00	0.04	0.05	0.02	0.02	0.04

Table 6: Model Performance on Analytical Tasks. Red indicates the best result, Blue indicates the second-best. A dash (–) denotes that no successful cases were recorded.

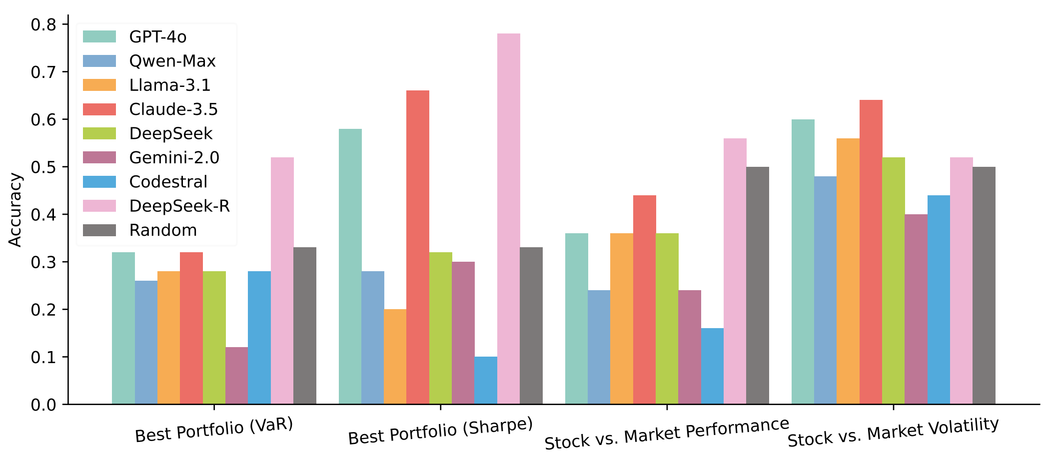

Figure 6: Model Performance on Decision-Making and Analysis-Interpretation Tasks in Multiple Choice Format

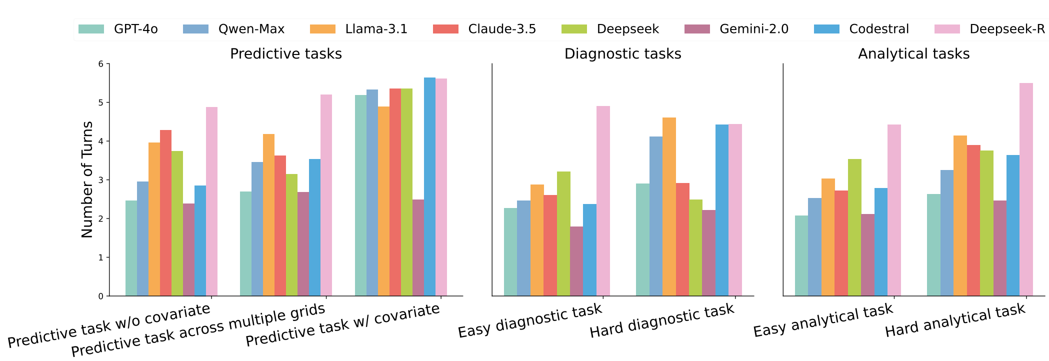

Figure 7: Average Number of Turns Models Take to Reach a Solution Grouped by Difficulty Level

Dataset	Number of Data Files	Avg Total Timestamps	Number of Variables
Climate Data	624	526	2048
Energy Data w/ geolocation	22	8760	1-3
Energy Data w/ Covariates	66	872601	11
Building Energy Usage Data	398	5019	1
Causal Data	8	529	3–6
Daily Stock Data	6780	3785	7
Hourly Stock Data	5540	35	7
Stock Market Indices Data	6	3388	4
ECG Signal Data	24	10804352	2

Table 7: Dataset Statistics of the constructed dataset. The exact number of time series are not calculated because it depends on randomly sampled sequence length when generating task instances.

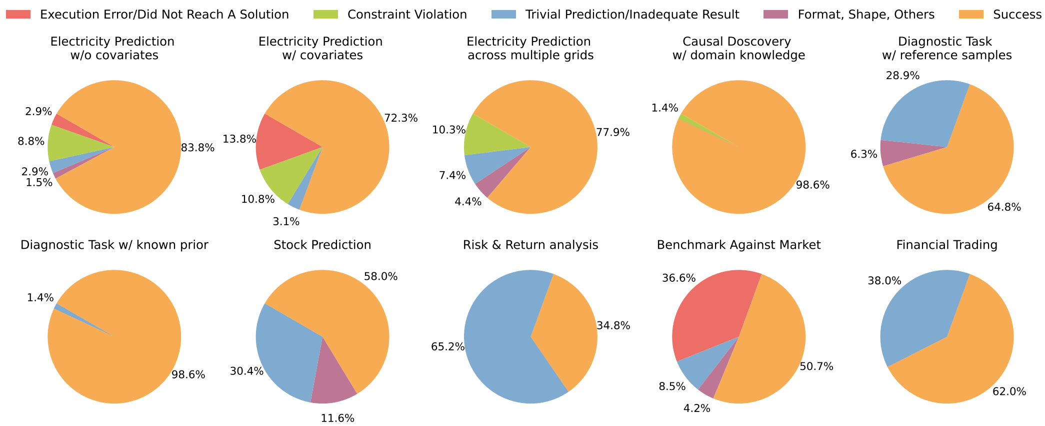

Figure 8: Case Study on Claude 3.5 Error Distribution across Tasks Grouped by Difficulty Level

D Additional Error Analysis

Beyond GPT-4o, we extend our error analysis to other representative models, whose detailed error distributions are visualized in Figures 8–14. While Claude-3.5 generally performs well, it exhibits a noticeable proportion of constraint violation errors in electricity prediction tasks, suggesting challenges in handling numerical constraints embedded within the input. Despite the complexity of financial trading tasks, Llama-3.1 performs competitively relative to other models—particularly notable given its open-source nature. In contrast, Gemini-2.0 and Codestral show a high incidence

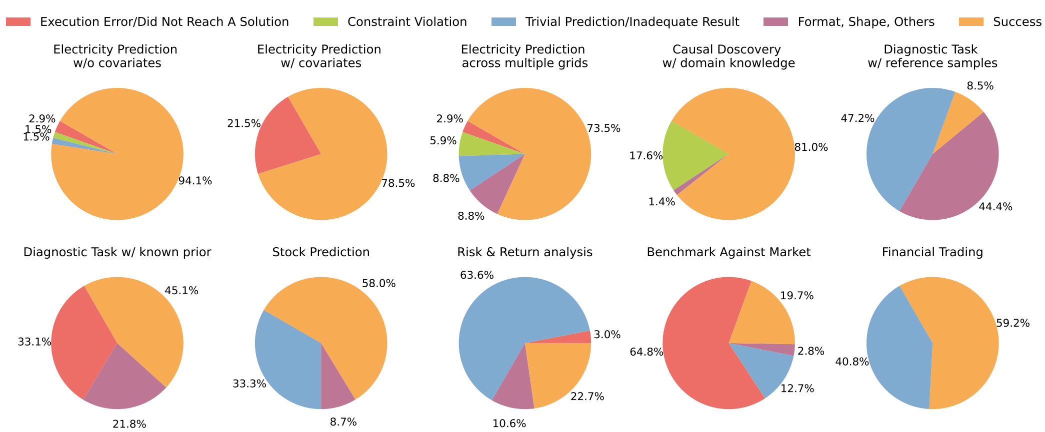

Figure 9: Case Study on Qwen Error Distribution across Tasks Grouped by Difficulty Level

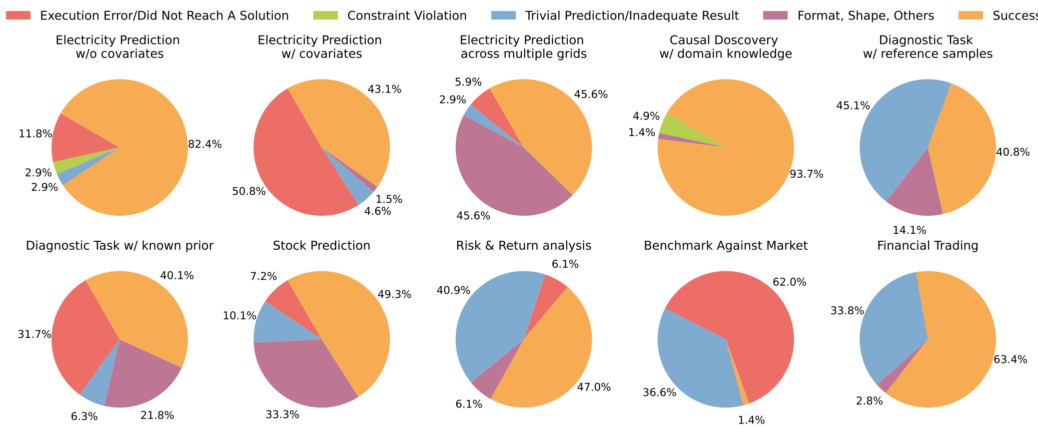

Figure 10: Case Study on Codestral Error Distribution across Tasks Grouped by Difficulty Level

of execution errors across nearly all task categories, indicating limited suitability for structured, multi-step time series reasoning. DeepSeek-R consistently avoids execution failures and maintains a relatively high success rate across tasks.

E CodeAct System Prompt Template

> You are a helpful assistant that gives helpful, detailed, and polite answers to the user's questions. The code written by assistant should be enclosed using <execute> tag, for example: <execute> print('Hello World!') </execute>. You should provide the solution in a single <execute> block instead of taking many turns. You'll receive feedback from your code execution. You should always import packages and define variables before starting to use them. You should stop <execute> and provide an answer when they have already obtained the answer from the execution result. Whenever possible, execute the code for the user using <execute> instead of providing it. Your response should be concise, but do express their thoughts. Always write the code in <execute> block to execute them. You should not ask for the user's input unless necessary. Solve the task on your own and leave no unanswered questions behind. You should do every thing by your self. You are not allowed to install any new packages or overwrite available variables provided to you in the question. Additionally, you are provided with the following variables available: {variable names} The above variables is already available in your interactive Python (Jupyter Notebook) environment, allowing you to directly use them without needing to re-declare them.

F Refinement Solution Path

Box F illustrates the solution refinement trajectory of Deepseek-R. Execution feedback from the CodeAct Python interpreter enabled the model to revise its output twice—first to address a syntax error due to a missing closing parenthesis, and second to resolve an import error. As shown in Figure 7, Deepseek-R consistently requires more interaction turns than other models. A closer examination reveals that this behavior stems from the lack of a proper stopping mechanism: although a correct solution was reached by turn 3, the model continued executing redundant steps in turns 4 and 5. Notably, Deepseek-R also incorporates explicit inline comments such as `# Total pairs: 5*4=20, top 20% is 4 pairs` to document its intermediate reasoning steps, contributing to its overall performance strength.

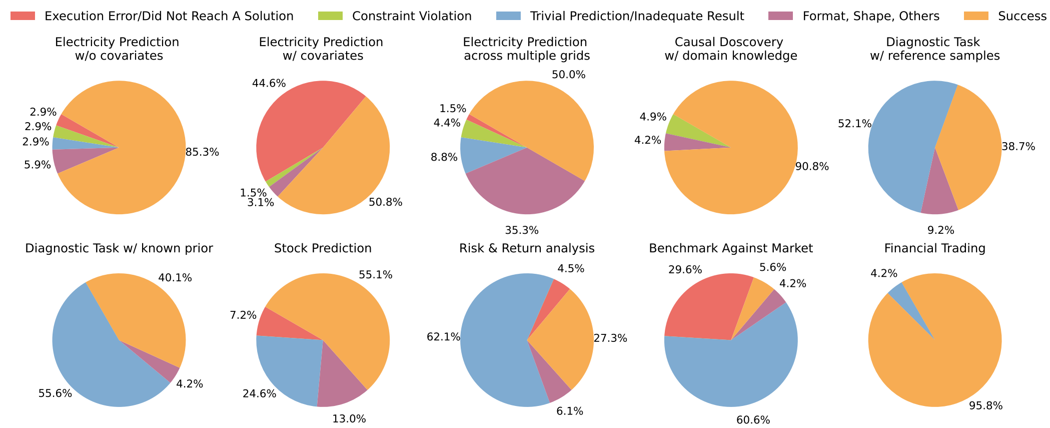

Figure 11: Case Study on Llama Error Distribution across Tasks Grouped by Difficulty Level

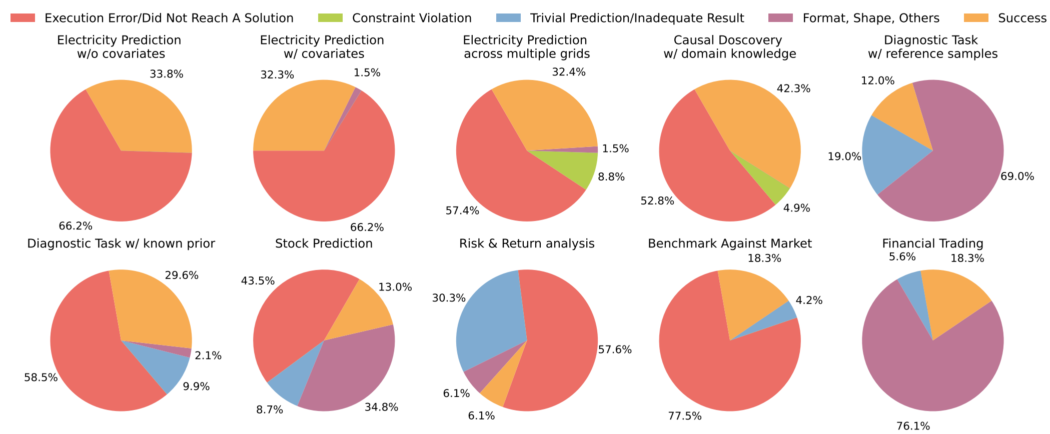

Figure 12: Case Study on Gemini Error Distribution across Tasks Grouped by Difficulty Level

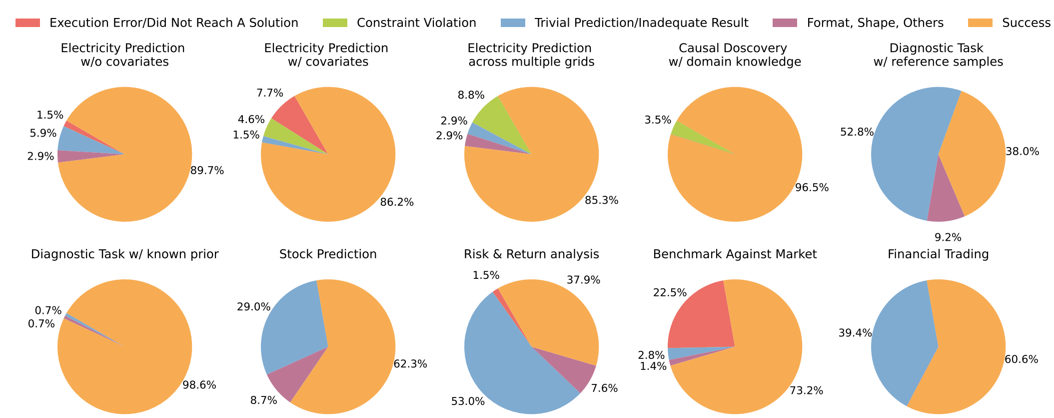

Figure 13: Case Study on Deepseek-chat Error Distribution across Tasks Grouped by Difficulty Level

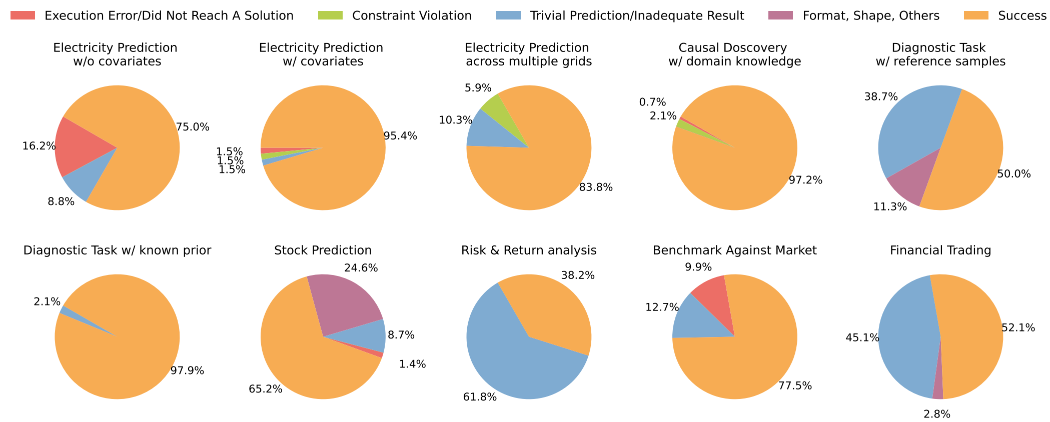

Figure 14: Case Study on Deepseek-R Error Distribution across Tasks Grouped by Difficulty Level

Deepseek-R Solution Path

User:
I have historical Temperature (A), Humidity (B), Air Quality Index (C), Wind Speed (D), Precipitation (E) data and want to get the causal relationship between each pair of the variables. I know that 20.0% of the variable pairs have relationship. Self-causalation is not considered. Consider the potential influence of each variable on the others in this variable list: [’Temperature (A)’, ’Humidity (B)’, ’Air Quality Index (C)’, ’Wind Speed (D)’, ’Precipitation (E)’]. Please provide 2d numpy matrix with binary values to indicate whether each pair of variables has a relationship. The data for all variables are stored in variable VAL.
Requirements:
- Store your output in the variable called ‘predictions‘, make sure to consider predictions to have the right shape according to the question’s output requirements if it’s supposed to be an array, there is no need to consider shapes if you only need to return a numerical value.
- Do not customly define/generate/overwrite the available variables, assume that the variables are already defined and available.
data note: VAL values are usually stored in either a dataframe or numpy values.
You should enclose your python code in <execute> </execute> tag and do not overwrite available variables that store the data. Do not use any other tags like “‘python“‘.

Assisstant:

```
<execute>
import numpy as np

# Handle VAL (dataframe or numpy array)
if isinstance(VAL, pd.DataFrame):
    data = VAL.values
else:
    data = VAL

n_vars = 5  # Number of variables
predictions = np.zeros((n_vars, n_vars), dtype=int)

# Calculate time-lagged correlations for directed pairs
correlations = []
for i in range(n_vars):
    for j in range(n_vars):
        if i != j:
            if data.shape[0] > 1:
                x = data[:-1, i]
```

```
                y = data[1:, j]
                corr = np.corrcoef(x, y)[0, 1] if len(x) > 1 else 0
                correlations.append((i, j, abs(corr if not np.isnan(corr)
                else 0))

# Sort by correlation strength and select top 20% (4 pairs)
sorted_pairs = sorted(correlations, key=lambda x: -x[2])[:4]

# Mark selected pairs in predictions matrix
for i, j, _ in sorted_pairs:
    predictions[i, j] = 1

# Ensure correct variable name (user specified 'predictions')
predictions = predictions
</execute>
```

User:
Execution Failed: Execution Output: An error occurred while executing the following cell: sorted_pairs = sorted(correlations, key=lambda x: -x[2])[:4] SyntaxError: invalid syntax
Side note: Remember to enclose your code in <execute> </execute> tag and do not overwrite any available variables provided to you in the question, especially that they contain the data.

Assistant:

```
<execute>
import numpy as np

# Handle VAL (dataframe or numpy array)
if isinstance(VAL, pd.DataFrame):
    data = VAL.values
else:
    data = VAL

n_vars = 5  # Number of variables
predictions = np.zeros((n_vars, n_vars), dtype=int)

# Calculate time-lagged correlations for directed pairs
correlations = []
for i in range(n_vars):
    for j in range(n_vars):
        if i != j:
            if data.shape[0] > 1:
                x = data[:-1, i]
                y = data[1:, j]
                if len(x) > 1:
                    corr = np.corrcoef(x, y)[0, 1]
                    corr = 0 if np.isnan(corr) else corr
                else:
                    corr = 0
                correlations.append((i, j, abs(corr)))

# Sort by correlation strength and select top 20% (4 pairs)
sorted_pairs = sorted(correlations, key=lambda x: -x[2])[:4]

# Mark selected pairs in predictions matrix
for i, j, _ in sorted_pairs:
    predictions[i, j] = 1
</execute>
```

User:

Execution Failed: Execution Output: An error occurred while executing the following cell: if isinstance(VAL, pd.DataFrame): NameError: name ’pd’ is not defined.
Side note: Remember to enclose your code in <execute> </execute> tag and do not overwrite any available variables provided to you in the question, especially that they contain the data.
Assistant:

```
<execute>
import numpy as np
import pandas as pd

# Handle VAL (dataframe or numpy array)
if isinstance(VAL, pd.DataFrame):
    data = VAL.values
else:
    data = VAL

n_vars = 5  # Number of variables
predictions = np.zeros((n_vars, n_vars), dtype=int)

# Calculate time-lagged correlations for directed pairs
correlations = []
for i in range(n_vars):
    for j in range(n_vars):
        if i != j:
            if data.shape[0] > 1:
                x = data[:-1, i]
                y = data[1:, j]
                if len(x) > 1:
                    corr = np.corrcoef(x, y)[0, 1]
                    corr = 0 if np.isnan(corr) else corr
                else:
                    corr = 0
                correlations.append((i, j, abs(corr)))

# Total pairs: 5*4=20, top 20% is 4 pairs
sorted_pairs = sorted(correlations, key=lambda x: -x[2])[:4]

# Update predictions matrix
for pair in sorted_pairs:
    i, j, _ = pair
    predictions[i, j] = 1
</execute>
```

User:
Execution Successful: Execution Output: []

Assistant:
The code now correctly imports necessary packages and handles both DataFrame and numpy array inputs for ‘VAL‘. It calculates time-lagged correlations between variables, selects the top 20% strongest relationships (4 pairs), and constructs the binary matrix ‘predictions‘ with 1s indicating causal directions. The final matrix respects the problem constraints and data structure requirements.

```
<execute>
import numpy as np
import pandas as pd

# Handle VAL (dataframe or numpy array)
if isinstance(VAL, pd.DataFrame):
    data = VAL.values
else:
```

```
    data = VAL

n_vars = 5  # Number of variables
predictions = np.zeros((n_vars, n_vars), dtype=int)

# Calculate time-lagged correlations for directed pairs
correlations = []
for i in range(n_vars):
    for j in range(n_vars):
        if i != j:
            if data.shape[0] > 1:
                x = data[:-1, i]
                y = data[1:, j]
                if len(x) > 1:
                    corr = np.corrcoef(x, y)[0, 1]
                    corr = 0 if np.isnan(corr) else corr
                else:
                    corr = 0
                correlations.append((i, j, abs(corr)))

# Sort by absolute correlation and select top 4 pairs (20%)
sorted_pairs = sorted(correlations, key=lambda x: -x[2])[:4]

# Update predictions matrix
for i, j, _ in sorted_pairs:
    predictions[i, j] = 1

predictions  # Final output matrix
</execute>
```

User:
Execution Successful: Execution Output: ['array([[0, 0, 1, 1, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[1, 0, 1, 0, 0],[0, 0, 0, 0, 0]])']
Assistant:
The code correctly identifies the top 20% of variable pairs with the strongest time-lagged correlations, resulting in a 5x5 binary matrix ‘predictions‘ where 1 indicates a causal relationship. The matrix shape and data type adhere to the problem's requirements.

```
<execute>
import numpy as np
import pandas as pd

# Handle VAL (dataframe or numpy array)
if isinstance(VAL, pd.DataFrame):
    data = VAL.values
else:
    data = VAL

n_vars = 5  # Number of variables
predictions = np.zeros((n_vars, n_vars), dtype=int)

# Calculate time-lagged correlations
correlations = []
for i in range(n_vars):
    for j in range(n_vars):
        if i != j and data.shape[0] > 1:
            x = data[:-1, i]
            y = data[1:, j]
            if len(x) >= 2:  # Ensure valid correlation calculation
```

```
                corr = np.corrcoef(x, y)[0, 1]
                corr = 0 if np.isnan(corr) else abs(corr)
                correlations.append((i, j, corr))

# Select top 20% pairs (4 out of 20 possible directed pairs)
top_pairs = sorted(correlations, key=lambda x: -x[2])[:4]
for i, j, _ in top_pairs:
    predictions[i, j] = 1

predictions
</execute>
```

User:
Execution Successful: Execution Output: [’array([[0, 0, 1, 1, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[1, 0, 1, 0, 0],[0, 0, 0, 0, 0]])’]
Assistant: I have reached the maximum number of executions. Can you assist me or ask me another question?
Evaluation: final value <class ’numpy.ndarray’> (5, 5) [[0 0 1 1 0] [0 0 0 0 0] [0 0 0 0 0] [1 0 1 0 0] [0 0 0 0 0]] evaluation result: {’status’: 1, ’accuracy’: 0.84}

G Casual Discovery Data Generation Prompt

Now you are a Time series data scientist, please help me to write the code to generate some synthetic data in real world Time series domain, you should save the data into "*/data.csv":

Now suggesting you should construct a series data based on a relation matrix and the correlation ratio for different influence factor, you should notice the following points,for time step I want you to generate 500 time steps:

1. data correlation: the multi variable should be correlated, sample: which A first influence B, then B have influence on C or D, there should be some time delay, as the influence on other staff needs time.

2. data trend: there should be some trend in the data, like the data is increasing or decreasing.

3. data: seasonality there should be some seasonality in the data, like the data is periodic.

4. data noise: the noise should be added to the data, as the real world data is not perfect.

5. data background: the data should have some real world background, you should first think about different real world data, and provide a description for the variable and time series data, then generate the data using the code. CoT Sample: Q: Approximate Relation Ratio: 0.5 Relation Matrix:

	A	B	C	D
A	1	1	0	1
B	0	1	0	1
C	0	1	1	1
D	0	0	0	1

- A influences B and D, and itself.
- B influences D, and itself.
- C influences B and D, and itself.
- D influences only itself.

variable size: 4 A: Scenario: Sales Data of a Chain of Stores Over Time Let’s assume we are generating synthetic data,the variable size for the data is 4. for the daily sales of multiple stores across a chain, the sales numbers are influenced by:

1. Advertising (A): The level of advertising spend directly impacts the sales of each store. After a delay, this starts influencing sales. 2. Sales (B): The sales numbers for each store are influenced by both the advertising and local seasonal events. 3. Economic Factors (C): Broader economic trends, like GDP growth or unemployment rates, also impact sales. These factors show a delayed and more

subtle influence over time. 4. Customer Sentiment (D): Customer sentiment affects the sales of specific products in each store and is influenced by both advertising and broader economic factors.

Seasonality: Sales experience periodic seasonal trends, with peaks around the holidays and lower numbers during off-seasons.

Trend: There is a general increasing trend in sales as the chain expands.

Noise: Random noise is added to mimic real-world data fluctuations.