Structured Agentic Workflows for Financial Time-Series Modeling with LLMs and Reflective Feedback

Yihao Ang[1,*] **Yifan Bao**[1,*] **Lei Jiang**[2,*] **Jiajie Tao**[2]
Anthony K. H. Tung[1,†] **Lukasz Szpruch**[3,†] **Hao Ni**[2,†]
[1]Department of Computer Science, National University of Singapore
[2]Department of Mathematics, University College London
[3]School of Mathematics, University of Edinburgh
{yihao_ang, yifan_bao, atung}@comp.nus.edu.sg
{lei.j, jiajie.tao.21, h.ni}@ucl.ac.uk, l.szpruch@ed.ac.uk

Abstract

Time-series data drives financial decision-making, yet building models that are simultaneously high-performing, interpretable, and auditable remains challenging. Automated Machine Learning (AutoML) streamlines development but often lacks domain adaptivity, while recent LLM-based agents enable end-to-end workflow automation. We introduce TS-Agent, a modular agentic framework designed to automate and enhance time-series modeling workflows for financial applications. The agent formalizes the pipeline as a structured, iterative decision process across three stages: model selection, code refinement, and fine-tuning, guided by contextual reasoning and experimental feedback. Central to our architecture is a planner agent equipped with structured knowledge banks, curated libraries of models and refinement strategies, which guide exploration, while improving interpretability and reducing error propagation. TS-Agent supports adaptive learning, robust debugging, and transparent auditing. Across financial forecasting and synthetic generation tasks, it consistently outperforms state-of-the-art AutoML and agentic baselines in accuracy, robustness, and decision traceability.

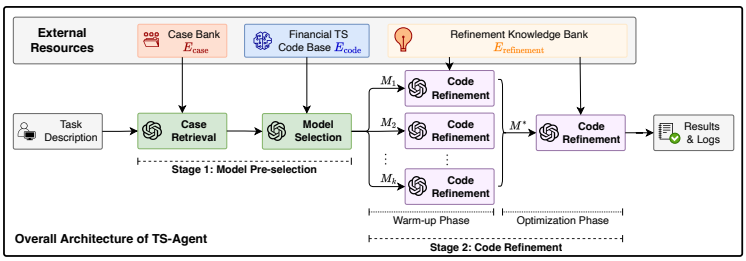
1 Introduction

Financial markets produce high-velocity, high-volume time series where timely, decision-relevant modeling is essential yet difficult. While AutoML systems accelerate pipeline construction, their static search strategies and optimization for generic statistical losses limit domain alignment and adaptability [7, 13]. Emerging LLM-based agentic systems couple language reasoning with code execution to automate end-to-end workflows [21, 10, 5, 2], but building agents that are *robust*, *auditable*, and *compliance-ready* for time-series modeling in finance remains open: beyond accuracy, practitioners require transparent processes that justify model choice, refinement, and deployment decisions and support human–AI collaboration [23, 10].

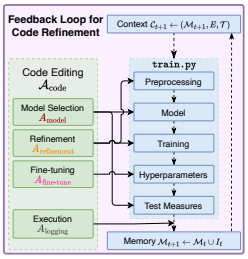
We propose TS-Agent, a modular agentic framework that automates and audits financial time-series workflows via iterative model selection, code refinement, and hyperparameter tuning guided by structured reasoning and execution feedback. TS-Agent integrates three read-only resources, a *Case Bank* for case-based reasoning, a *Financial Time-Series Code Base* of executable models/metrics for reuse, and a *Refinement Knowledge Bank* encoding expert heuristics, while logging every decision

[*]These authors contributed equally to this work.
[†]These authors are corresponding authors.

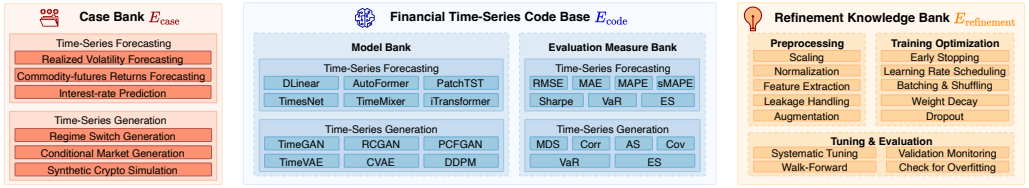
39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) Overall architecture of TS-Agent.



(b) Code refinement loop.



(c) Key components in the three external resources.

Figure 1: Overview of TS-Agent.

and rationale for reproducibility and fault localization [10]. A planner implements feedback-driven updates (beyond static AutoML and naive agents), isolates edits to refinement modules for debuggability, and aligns optimization with finance-aware metrics. Experiments across stock, exchange, and crypto tasks prove that it delivers superior accuracy, trading utility, robustness, and success consistency relative to AutoML and agentic baselines, with transparent, compliance-friendly traces.

2 Related Work

We review two primary lines of research related to TS-Agent: *Automated Machine Learning* (AutoML) and *agentic systems*. AutoML automates preprocessing, model selection, and hyperparameter tuning, commonly via Bayesian optimization and ensembling [24, 9, 13, 1]. Early systems jointly optimize pipelines [24, 9]; for time series, many reduce forecasting to regression with manual features, while AutoGluon extends to probabilistic forecasting and Optuna offers scalable hyperparameter optimization [7, 1]. LLM agents decompose goals, call tools, and self-correct [21, 29, 22, 25, 23] with domain variants support research ideation and data science via retrieval/-case adaptation [5, 10]. For high-stakes finance, they lack transparency, alignment with trading, risk, and compliance, and integration with existing codebases. TS-Agent targets these needs with finance-grounded planning, vetted toolchains, and human–AI collaboration.

3 Problem formulation

Given a financial time-series task $\mathcal{T} = (\text{desc}, \mathcal{D}, \mathcal{L})$ with data $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$ and an evaluation criterion $\mathcal{L}$, TS-Agent uses read-only external resources and an iterative editing loop to produce an executable `train.py` that minimizes $\mathcal{L}$ on $\mathcal{D}_{\text{test}}$ while logging all decisions. In this work, we mainly consider time series *forecasting* and *generation* tasks. For forecasting, let $X_t \in \mathbb{R}^d$ and define windowed pairs $(X_{t-p+1:t}, X_{t+1:t+q})t \in \mathfrak{T}$. A model $F_\theta : \mathbb{R}^{d\times p} \to \mathbb{R}^{d\times q}$ predicts future values; a common loss is MSE $:= \frac{1}{\mathfrak{T}} \sum_{t\in\mathfrak{T}} ||F_\theta(X_{t-p+1:t}) - X_{t+1,t+q}||^2$, often augmented with finance-aware metrics. For generation, given segments $(X_{t:t+q})_{t\in\mathfrak{T}}$, learn $G_\theta : \mathcal{Z} \to \mathbb{R}^{d\times q}$ mapping noise $\xi \sim \mathbb{P}$ to samples whose distribution matches that of $X_{t:t+q}$; quality is assessed via statistical distances and task-specific financial criteria.

Learning Framework. In practice, financial institutions typically maintain their own code pipelines for analyzing financial time-series data and have their domain knowledge bases. To reflect it, we assume that the agent has access to three read-only external resources. (1) **Case Bank** (E_{case}, text): a curated library of financial time-series forecasting and generation tasks with concise reports, retrieved to guide new tasks [10]. Sources include benchmarks, competitions, and peer-reviewed studies [16, 18, 17, 3]; data span equities, exchange, crypto, and synthetic; entries map tasks to

Table 1: Time series forecasting performance on three benchmark datasets. The best result for each LLM (per column) is bolded.

Dataset	Model	RMSE ↓				MAE ↓				MAPE (%) ↓				sMAPE (%) ↓				Success Rate (%) ↑			
		GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova
Crypto	TS-Agent	**0.277**	**0.206**	**0.254**	**0.249**	**0.0076**	**0.0051**	**0.0066**	**0.0063**	**1.692**	**1.531**	**1.643**	**1.655**	**1.693**	**1.529**	**1.644**	**1.658**	**100**	**100**	**100**	**100**
	DS-Agent	0.369	0.297	0.320	0.307	0.0082	0.0070	0.0074	0.0073	2.065	1.883	1.905	1.912	2.063	1.881	1.907	1.914	60	60	60	60
	ResearchAgent	0.392	0.341	0.355	3.477	0.0094	0.0083	0.0088	0.0085	2.344	2.040	2.210	2.198	2.347	2.040	2.230	2.200	40	60	60	60
	AutoGluon	0.223				0.0055				1.664				1.662				100			
Exchange	TS-Agent	**0.0073**	**0.0068**	**0.0069**	**0.0077**	**0.0041**	**0.0036**	**0.0040**	**0.0041**	**0.564**	**0.474**	**0.499**	**0.562**	**0.563**	**0.474**	**0.498**	**0.560**	**100**	**100**	**100**	**80**
	DS-Agent	0.0095	0.0088	0.0090	0.0096	0.0066	0.0057	0.0062	0.0068	0.801	0.782	0.792	0.811	0.800	0.781	0.792	0.810	80	**100**	60	60
	ResearchAgent	0.0099	0.0096	0.0095	0.0098	0.0066	0.0056	0.0054	0.0059	0.812	0.793	0.804	0.817	0.811	0.79	0.804	0.817	60	40	80	60
	AutoGluon	0.0089				0.0052				0.701				0.699				100			
Stock	TS-Agent	8.727	**8.017**	**7.982**	**8.590**	**5.117**	**4.912**	**4.905**	**5.047**	2.336	**2.046**	**2.076**	**2.123**	**2.001**	**1.770**	**1.765**	**1.850**	**80**	**100**	**100**	**100**
	DS-Agent	**8.644**	8.557	8.559	8.732	5.248	5.193	5.150	5.207	**2.308**	2.137	2.177	2.244	2.142	1.969	2.055	2.099	60	80	**100**	60
	ResearchAgent	9.890	9.410	9.570	9.791	6.041	5.677	5.738	5.910	2.878	2.498	2.420	2.590	2.414	2.053	2.238	2.331	60	80	80	80
	AutoGluon	8.430				5.258				2.174				1.890				100			

Table 2: Trading performance on time series forecasting task for Crypto dataset.

Dataset	Model	Sharpe Ratio Difference ↓				VaR Difference ↓				ES Difference ↓			
		GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova
Crypto	TS-Agent	**0.414**	**0.378**	**0.403**	**0.407**	**0.0088**	**0.0069**	**0.0077**	**0.0080**	**0.0082**	**0.0066**	0.0067	**0.0069**
	DS-Agent	0.484	0.46	0.458	0.462	0.0130	0.0089	0.0092	0.0098	0.0121	0.0084	**0.0061**	0.0089
	ResearchAgent	0.479	0.471	0.472	0.475	0.0117	0.0086	0.0094	0.0090	0.0111	0.0081	0.0087	0.0088
	AutoGluon	0.402				0.0076				0.0072			

effective model families. (2) **Refinement Knowledge Bank** ($E_{\text{refinement}}$, text): best practices for *preprocessing*, *training*, and *tuning/evaluation*, linked to logged outcomes. (3) **Code Base** (E_{code}): read-only repository with (i) a *Model Bank* covering forecasting and generation implementations, and (ii) an *Evaluation Measure Bank* including standard statistical scores and finance-specific tests.

4 TS-Agent Framework

TS-Agent automates financial time-series modeling by iteratively editing a modular `train.py` using three read-only resources (case bank, refinement knowledge, code base) and prior run logs. The system follows two stages: (1) model pre-selection and (2) code refinement, and produces reproducible scripts with auditable traces, as depicted in Figure 1(a).

The action space comprises $\mathcal{A}_{\text{model}}$ (choose models/measures), $\mathcal{A}_{\text{refine}}$ (insert training strategies), $\mathcal{A}_{\text{tune}}$ (tune hyperparameters), and $\mathcal{A}_{\text{log}}$ (execute and record). $\mathcal{A}_{\text{refine}}$ might introduce bugs; the agent applies iterate–fix–rerun debugging [10]. At step t, memory $\mathcal{M}_t = (I_v, \mathcal{S}_v)_{v \le t}$ stores logs I_v and code states $\mathcal{S}_v$, and context $\mathcal{C}_t = \mathcal{M}_t \cup E \cup \mathcal{T}$ conditions decisions. The chain-of-code-edits follow $\pi(\mathcal{A}_{\text{code}}|\mathcal{C}_t) = \pi(A_{\text{model}}|\mathcal{C}_t) \cdot \pi(A_{\text{refinement}}|A_{\text{model}}, \mathcal{C}_t) \cdot \pi(A_{\text{fine-tune}}|A_{\text{model}}, A_{\text{refinement}}, \mathcal{C}_t)$. We illustrate the feedback loop in Figure 1(b) and Algorithm 1.

Algorithm 1: Feedback Loop for Code Refinement.

```
Initialize contextual information C_1;
for t ← 1 to T_max do
    Conduct Model Selection, Refinement, Fine-tuning;
    Conduct action A_logging and record the log I_t;
    Update the agent's memory M_{t+1} ← M_t ∪ I_t;
    Update the context C_{t+1} ← (M_{t+1}, E, T);
end
```

Each iteration retrieves promising models via case-based reasoning from case bank [10], refines training using the refinement knowledge and recent logs, tunes hyperparameters, then executes and logs outcomes that extend $\mathcal{M}_t$. Stage 1 instantiates the top-k candidates in `train.py` by Model Selection, while Stage 2 runs a two-phase round-robin search [20]: a short, parallel warm-up to pick an incumbent and conducts Code Refinement, which iteratively applies Refinement and Fine-tuning, followed by optimization cycles that accept edits only if the loss improves (else revert), yielding decision-aligned performance with complete audit trails.

5 Experiments

Setup. We evaluate TS-Agent on forecasting and generation tasks using three financial datasets Crypto (hourly, 20 USDT pairs, 2024) [4], Exchange (daily FX, 1990–2010) [12], and Stock (daily U.S. equities, 2020–2024) [30]. Baselines include DS-Agent [10], ResearchAgent [5], and AutoML tools (AutoGluon for forecasting [7], Optuna for generation [1]). All agents are instantiated with GPT-3.5, GPT-4o, Claude Sonnet 4, and Nova Pro.

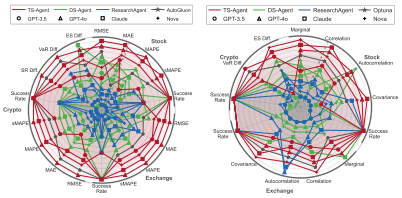
Table 3: Time series generation performance on Exchange and Stock datasets. — indicates that no successful runs were recorded. The best result for each LLM (per column) is bolded.

Dataset	Model	Marginal ↓				Correlation ↓				Autocorrelation ↓				Covariance ↓				Success Rate (%) ↑			
		GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova
Exchange	TS-Agent	**0.360**	**0.356**	0.374	**0.366**	**4.249**	**2.796**	**2.803**	**1.874**	**0.00426**	0.00216	0.00226	**0.00120**	**0.00301**	**0.00179**	**0.00151**	**0.00120**	**100**	**100**	**100**	**100**
	DS-Agent	—	0.692	**0.329**	0.401	—	14.755	3.373	14.290	—	0.0605	0.00475	0.0347	—	0.00706	0.00287	0.00600	0	40	**100**	20
	ResearchAgent	—	0.737	0.530	—	—	9.591	6.369	—	—	**0.000884**	**0.00105**	—	—	0.00595	0.00403	—	0	80	**100**	0
	Optuna	0.377				2.114				0.00198				0.00172				100			
Stock	TS-Agent	**0.309**	**0.269**	**0.266**	**0.259**	**3.468**	**1.228**	**1.194**	**1.413**	**0.227**	**0.152**	**0.153**	0.161	$\mathbf{6.99 \times 10^{-5}}$	$\mathbf{4.43 \times 10^{-5}}$	$\mathbf{4.35 \times 10^{-5}}$	$\mathbf{4.50 \times 10^{-5}}$	**100**	**100**	**100**	**100**
	DS-Agent	—	0.299	0.331	0.277	—	6.930	5.282	11.991	—	0.231	0.286	**0.147**	—	5.58×10^{-5}	5.81×10^{-5}	5.63×10^{-5}	0	80	**100**	40
	ResearchAgent	—	0.575	0.397	0.904	—	7.175	7.343	12.879	—	0.490	0.329	1.244	—	5.75×10^{-5}	4.74×10^{-5}	5.84×10^{-5}	0	80	**100**	40
	Optuna	0.270				1.530				0.162				4.83×10^{-5}				100			

Table 4: Time series generation performance on Crypto dataset.

Model	VaR Difference ↓				ES Difference ↓				Success Rate (%) ↑			
	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova	GPT-3.5	GPT-4o	Claude	Nova
TS-Agent	**0.00205**	**0.00326**	**0.00235**	**0.00245**	**0.000714**	0.00299	**0.000629**	**0.000566**	**100**	**100**	**100**	**100**
DS-Agent	0.00564	0.00378	0.00427	—	0.00617	**0.000729**	0.00115	—	20	**100**	**100**	0
ResearchAgent	0.0107	0.0105	0.00557	0.0126	0.00227	0.00149	0.00147	0.00113	40	**100**	**100**	60
Optuna	0.00220				0.000832				100			

Time Series Forecasting (TSF). We evaluate five representative forecasting models (i.e., Autoformer [28], PatchTST [19], TimesNet [27], DLinear [32], and TimeMixer [26]) and report averaged RMSE, MAE, MAPE, sMAPE, for Crypto, differences in Sharpe, VaR, and ES, alongside a 5-run Success Rate [17]. Table 1 shows that TS-Agent with its best LLM variant (typically GPT-4o or Claude) outperforms baselines, reducing RMSE by $> 20\%$ on Exchange and $\sim 8\%$ on Crypto vs. AutoGluon, and by up to 30% vs. DS-Agent, while maintaining a 100% success rate. On risk fidelity (Table 2), it achieves 20% lower Sharpe/-VaR differences than the strongest competing agent with competitive ES, indicating preservation of risk-sensitive structure in volatile regimes. Ranking profiles (Figure 2(a)) place TS-Agent with the best average rank and the outermost contour. Although GPT-4o is generally strongest, margins are smaller for TS-Agent, consistent with backbone-agnostic resilience from refining vetted Financial TS Code Base implementations rather than synthesizing models from scratch.



(a) TSF tasks. (b) TSG tasks.

Figure 2: Rankings on two time series tasks.

Time Series Generation (TSG). We benchmark five representative models (i.e., TimeGAN [31], PCFGAN [14], RCGAN [8], TimeVAE [6], and DDPM [11]) and assess fidelity on Exchange/Stock via Marginal, Correlation, Autocorrelation, and Covariance scores [15, 30], and on Crypto via tail-risk fit (VaR, ES) owing to heavy-tailed returns [17]. Tables 3 and 4 demonstrate that, across datasets and LLM backbones, TS-Agent matches or exceeds Optuna and consistently outperforms agentic baselines, achieving a 100% success rate with markedly lower error dispersion. Rankings (Figure 2(b)) further place TS-Agent on the outermost contour (best overall), Optuna second, and the generic agents trailing, evidence that domain-informed, code-grounded workflows yield higher-quality, more robust synthetic series than pure AutoML or generic agents.

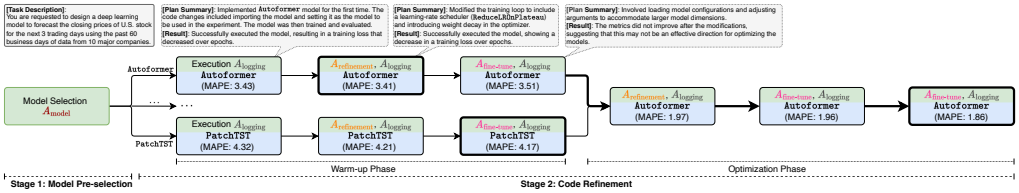


Figure 3: An illustration of TS-Agent's two-stage workflow on a financial TS forecasting task.

Case Study. On a forecasting task (Figure 3) predicting the next three trading-day closes for ten U.S. stocks from a 60-day window (metric: average MAPE), TS-Agent begins with a scaffolded `train.py`, performs Stage 1 case-based model pre-selection [10] to shortlist `Autoformer` and `PatchTST` [28, 19], then executes a Stage 2 two-phase round-robin refine–tune–execute loop with auditable logs; warm-up retains the best per model (`Autoformer` 3.41 vs. `PatchTST` 4.17 MAPE), and optimization further improves `Autoformer` to 1.86 MAPE by accepting only loss-reducing edits, demonstrating efficient, transparent end-to-end automation.

6 Conclusion

This paper introduces TS-Agent, a modular, structured agentic framework designed to automate financial time-series workflows. By formalizing the modeling pipeline as a multi-stage decision process, TS-Agent integrates domain-specific knowledge and feedback-driven reasoning to deliver interpretable, adaptive, and high-performing solutions across forecasting and generation tasks.

Acknowledgments

HN and LJ are supported by the EPSRC [grant number EP/S026347/1]. HN is also supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1. Moreover, HN and LJ gratefully acknowledge the support of the Impact Accelerator program at the UCL Centre for Digital Innovation, powered by Amazon Web Services. This project is partially supported by the AWS computing resources. They especially thank Tomasz Grzybowski (AWS) and Igor Tseyzer (UCL) for their valuable insights on the implementation of the LLM agent workflow. Besides, HN and LJ thank Hang Lou for useful discussions on the evaluation bank design. In addition, AT, YA, and YB are supported by the Ministry of Education, Singapore, under its MOE AcRF TIER 1 Grant (T1 251RES2517). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. Moreover, LS acknowledges the support of the UKRI Prosperity Partnership Schemes: EP/V056883/1 - FAIR Framework for responsible adoption of Artificial Intelligence in the financial services industry and APP43592: AI² – Assurance and Insurance for Artificial Intelligence, which supported this work.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, pages 2623–2631, 2019.

[2] Yihao Ang, Yifan Bao, Qiang Huang, Anthony KH Tung, and Zhiyong Huang. Tsgassist: An interactive assistant harnessing llms and rag for time series generation recommendations and benchmarking. *Proceedings of the VLDB Endowment*, 17(12):4309–4312, 2024.

[3] Yihao Ang, Qiang Huang, Yifan Bao, Anthony KH Tung, and Zhiyong Huang. Tsgbench: Time series generation benchmark. *Proc. VLDB Endow.*, 17(3):305–318, 2023.

[4] Yihao Ang, Qiang Wang, Qiang Huang, Yifan Bao, Xinyu Xi, Anthony KH Tung, Chen Jin, and Zhiyong Huang. Ctbench: Cryptocurrency time series generation benchmark. *arXiv preprint arXiv:2508.02758*, 2025.

[5] Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *NAACL*, pages 6709–6738, 2025.

[6] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

[7] David Eriksson and et al. Autogluon-timeseries: Robust automl for time series forecasting. `https://github.com/autogluon/autogluon`, 2023. Accessed: 2025-07-30.

[8] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[9] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *NeurIPS*, volume 28, 2015.

[10] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*, 2024.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, pages 6840–6851, 2020.

[12] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, pages 95–104, 2018.

[13] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*, 2020.

[14] Hang Lou, Siran Li, and Hao Ni. Pcf-gan: generating sequential data via the characteristic function of measures on the path space. In *NeurIPS*, pages 39755–39781, 2023.

[15] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein gans for time series generation. In *ICAIF*, pages 1–8, 2021.

[16] Hao Ni, Lukasz Szpruch, and Jiajie Tao. Regime-switching Financial Time-Series Generation. `https://github.com/tjj0502/hackathon_starting_kit`, 2023. ICAIF 2023 Hackathon.

[17] Hao Ni, Lukasz Szpruch, Jiajie Tao, and Yang Long. Crypto Market Simulation for Risk Estimation. `https://hackathon.deepintomlf.ai/competitions/40`, 2024. Antalpha ICAIF 2024 Hackathon.

[18] Hao Ni and Jiajie Tao. Market Scenario Generator Hackathon: From Stability to Storms. `https://github.com/DeepIntoStreams/Market_Scenario_Generator_Hackathon_starting_kit`, 2023. ICAIF 2023 Hackathon.

[19] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

[20] Rasmus V Rasmussen and Michael A Trick. Round robin scheduling–a survey. *European Journal of Operational Research*, 188(3):617–636, 2008.

[21] Toran Bruce Richards. Autogpt: An autonomous gpt-4 experiment. `https://github.com/Torantulino/Auto-GPT`, 2023.

[22] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, volume 36, pages 8634–8652, 2023.

[23] Amanpreet Singh and et al. Large language models orchestrating structured reasoning achieve kaggle grandmaster level. *arXiv preprint arXiv:2307.05068*, 2023.

[24] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, pages 847–855, 2013.

[25] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *TMLR*.

[26] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *ICLR*, 2024.

[27] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, 2023.

[28] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, volume 34, pages 22419–22430, 2021.

[29] Shinn Yao, Jiong Zhao, Dian Yu, and et al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[30] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *NeurIPS*, pages 5509–5519, 2019.

[31] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *NeurIPS*, pages 5509–5519, 2019.

[32] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, volume 37, pages 11121–11128, 2023.