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

Yihao Ang*
National University of Singapore
yihao_ang@comp.nus.edu.sg

Yifan Bao*
National University of Singapore
yifan_bao@comp.nus.edu.sg

Lei Jiang*
University College London
lei.j@ucl.ac.uk

Jiajie Tao
University College London
jiajie.tao.21@ucl.ac.uk

Anthony K. H. Tung†
National University of Singapore
atung@comp.nus.edu.sg

Lukasz Szpruch†
University of Edinburgh
l.szpruch@ed.ac.uk

Hao Ni†
University College London
h.ni@ucl.ac.uk

ABSTRACT

Time-series data is central to decision-making in financial markets, yet building high-performing, interpretable, and auditable models remains a major challenge. While Automated Machine Learning (AutoML) frameworks streamline model development, they often lack adaptability and responsiveness to domain-specific needs and evolving objectives. Concurrently, Large Language Models (LLMs) have enabled agentic systems capable of reasoning, memory management, and dynamic code generation, offering a path toward more flexible workflow automation. In this paper, 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, key requirements for high-stakes environments such as financial services. Empirical evaluations on diverse financial forecasting and synthetic data generation tasks demonstrate that TS-Agent consistently outperforms state-of-the-art AutoML and agentic baselines, achieving superior accuracy, robustness, and decision traceability.

CCS CONCEPTS

• **Computing methodologies** → **Intelligent agents**; • **Mathematics of computing** → *Time series analysis.*

KEYWORDS

Financial time series; LLM; Agents

1 INTRODUCTION

The financial industry operates at an unprecedented velocity and scale, continuously generating massive streams of time-series data. Extracting actionable and timely insights from these large, dynamically evolving datasets remains a fundamental challenge. Automated machine learning (AutoML) systems such as AutoGluon [7] and H2O AutoML [14] aim to reduce barriers to entry by automating feature selection and hyperparameter tuning. These systems have accelerated model development cycles and empowered non-expert users. However, their reliance on static, rule-based model selection strategies—optimized predominantly for general-purpose statistical metrics—limits their adaptability. Recent advances in Large Language Models (LLMs) have enabled the emergence of agentic systems for data science automation. They aim to automate end-to-end workflows by coupling natural language reasoning with code generation and execution [2, 5, 10, 22]. For example, ResearchAgent [5] tackles general data science workflows, while DS-Agent [10] introduces a Case-Based Reasoning (CBR) paradigm that reuses historical solutions from online sources to guide LLM behavior. These frameworks reflect a growing trend toward *agentic automation*, where LLMs not only interface with structured data but also orchestrate complex reasoning and decision-making pipelines.

Despite this progress, building robust, adaptive, and auditable agents for time-series modeling remains a significant open problem. In regulated environments like financial services, performance alone is insufficient. Auditable and interpretable processes are essential for ensuring regulatory compliance, enabling human-AI collaboration, and building trust. It is not only important to generate high-performing models but also to *understand and justify how those models are conceived, selected, and refined.*

In this paper, we introduce TS-Agent, a modular agentic framework designed to automate, enhance, and audit financial time-series modeling workflows. Unlike traditional AutoML systems that treat modeling as a static optimization problem, our approach builds upon advances in structured reasoning systems [10, 24], extending them in two critical directions. First, we integrate domain-specific knowledge by incorporating curated external resources, mirroring the modularity and reuse common in real-world quantitative finance workflows. Second, we embed a reflective feedback mechanism, enabling the agent to iteratively update decisions based on empirical results and code execution logs, thereby improving adaptability, robustness, and transparency.

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

Key Contributions. TS-Agent formalizes the workflow as a dynamic, multi-stage decision process, consisting of *model selection*, *code refinement*, and *hyperparameter fine-tuning*, executed iteratively via a planner agent. At each stage, decisions are informed by structured reasoning, contextual memory, and performance feedback. Our core contributions include:

- **Structured Knowledge Banks for Context-Aware Decision-Making**: TS-Agent draws on three external resources to inform its decisions: (1) a *Case Bank* of past financial modeling tasks and solutions for case-based reasoning; (2) a *Financial Time-Series Code Base* with executable models and metrics for direct reuse; and (3) a *Refinement Knowledge Bank* encoding expert heuristics and diagnostic strategies to guide context-aware, iterative model refinement grounded in financial best practices.
- **Feedback-Driven Online Learning**: The planner agent continually updates its policy based on feedback from experimental outcomes. This enables adaptive refinement loops that surpass the limitations of static AutoML pipelines and naive LLM-based agents, while offering a consistent interface for introspection, debugging, and improvement.
- **Auditable and Debuggable by Design**: Our modular architecture isolates code modifications to specific refinement modules, while logging each decision and its rationale. This design facilitates reproducibility, fault localization, and compliance auditing—key requirements for high-stakes AI deployment in financial environments.
- **Empirical Validation on Financial Tasks:** We evaluate TS-Agent across diverse real-world financial tasks—including stock price forecasting, cryptocurrency prediction, and synthetic time-series generation. Our results show that TS-Agent consistently outperforms AutoML pipelines and LLM-based agents in predictive accuracy, trading utility, and robustness while offering greater interpretability and success consistency.

Together, these contributions demonstrate the feasibility and advantages of agentic, interpretable, and auditable automation for time-series workflows in high-stakes domains.

2 RELATED WORK

We review two primary lines of research related to TS-Agent: *automated machine learning* and *agentic systems*. The former focuses on automating the model development cycle, while the latter emphasizes autonomous reasoning and task execution.

2.1 Automated Machine Learning

Automated machine learning (AutoML) systems automate end-to-end machine learning workflows, covering data preprocessing, feature engineering, model selection, and hyperparameter optimization [1, 9, 14, 25]. Typical pipelines involve (1) selecting preprocessors, (2) searching over model families, and (3) tuning hyperparameters. Recent frameworks also incorporate meta-learning and ensembling for improved generalization [7]. Early systems such as Auto-WEKA [25] and Auto-sklearn [9] formulate the model selection and hyperparameter tuning problem as a joint optimization task using Bayesian optimization. In time-series domains, AutoML systems typically treat forecasting as generic regression, requiring manual feature engineering for lags and trends. AutoGluon [7] mitigate this by integrating diverse models for probabilistic forecasting, while Optuna [1] is an efficient and scalable hyperparameter optimization framework for generative modeling tasks.

Despite progress, existing AutoML systems face critical gaps in financial time-series tasks. First, they lack domain-specific reasoning capabilities. Second, most operate as black-box search tools with limited interpretability. Third, they usually optimize for statistical error metrics, often ignoring financial performance indicators. TS-Agent addresses these gaps by embedding domain-specific diagnostics and financial context reasoning directly into the automation loop, moving beyond static AutoML approach, aligning model selection with financial decision-making needs.

2.2 Agentic Systems

Agentic systems enable autonomous planning and execution over multi-step workflows by LLMs. Unlike static AutoML pipelines, they typically perform task decomposition, invoke external tools, and iteratively refine outputs based on feedback [23, 24, 26, 31]. General-purpose frameworks like AutoGPT [22] support autonomous goal decomposition and tool integration. The ReAct framework [31] interleaves reasoning traces with task-specific actions, allowing agents to dynamically plan and update their behaviour. The Reflexion [23] introduced *verbal reinforcement learning*, where feedback from previous episodes is transformed into natural language self-reflections, effectively encoding an interpretable memory that supports iterative learning and credit assignment. While general-purpose agentic systems are flexible, they often lack domain specialization. Recent works have explored agentic systems tailored for data science and time-series tasks. For example, ResearchAgent [5] tackled the problem of scientific ideation by constructing an LLM agent that proposes research ideas, retrieves literature, and iteratively refines its outputs through self-critique and multi-agent peer review. DS-Agent [10] introduced a case-based reasoning framework for automated data science. It retrieves relevant cases from a human insight database, adapts them to new tasks, and iteratively revises the solution based on execution feedback.

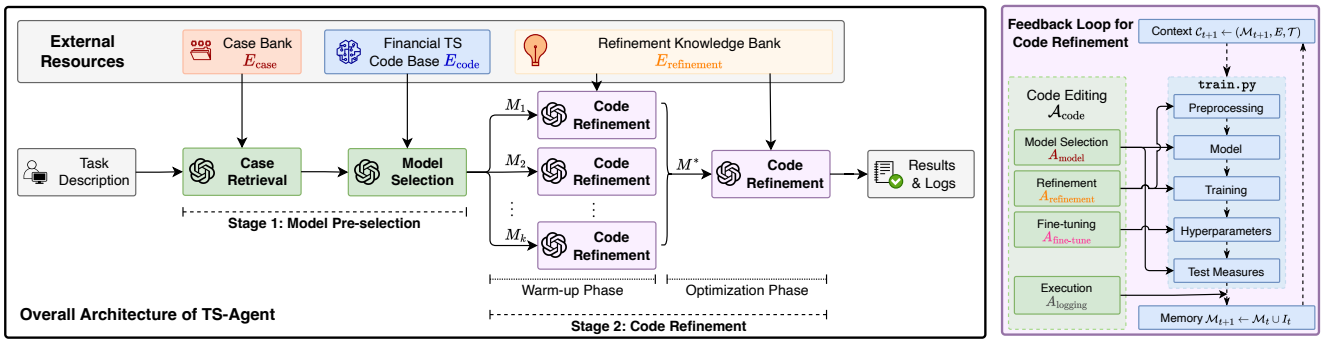
Current agentic systems, however, are not tailored for high stakes decision making. The often lack transparency, adaptability, and effective human–AI teaming, making it ill-suited for deployment in high-stakes financial services. Furthermore, for these system to be adapted then need to integrate with already existing code bases and align with organizations specific know-hows.

3 PROBLEM FORMULATION

The main objective of the TS-Agent is to automatically solve any given financial time-series tasks accurately and effectively by leveraging the external library information (including the contextual information and code base) and the flexible agent workflow.

3.1 Financial Time-Series Tasks

A machine learning task typically consists of a task description ($\mathcal{T}$), the dataset ($\mathcal{D}$), and an evaluation criterion for model performance ($\mathcal{L}$). Here, the dataset $\mathcal{D} := (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$ contains the information



(a) Overall architecture of TS-Agent.

(b) Feedback loop for code refinement.

Figure 1: Overview of TS-Agent.

about the training and test split. Model training is performed exclusively on the training dataset, and once training is complete, the evaluation criterion $\mathcal{L}$ is used to assess the model's performance on the test dataset. In this paper, we primarily focus on two categories of financial time-series tasks, described below. The evaluation criteria may include general-purpose metrics for each task type or be tailored to specific financial applications, depending on the use case. We offer concrete examples of evaluation measures in **§5**.

Time-Series Forecasting Tasks. The forecasting task refers to the prediction of the future time-series given the past ones. Let $X := (X_t)_{t\in\mathbb{N}}$ denote a d-dimensional time-series. It is the regression task where the input and output pairs are $(X_{t-p:t}, X_{t+1,t+q})_{t\in\mathfrak{T}}$, where $\mathfrak{T}$ is the set of available time stamps. The objective of this task is to predict the conditional expectation of $\mathbb{E}[X_{t+1:t+q}|X_{t-p+1:t}]$ given the past time-series information $X_{t-p:t}$ as accurately as possible. Let $F_\theta : \mathbb{R}^{d\times p} \to \mathbb{R}^{d\times q}$. One commonly used evaluation metric is Mean Squared Error (MSE), which is defined as

$$\text{MSE} := \frac{1}{\mathfrak{T}} \sum_{t\in\mathfrak{T}} ||F_\theta(X_{t-p+1:t}) - X_{t+1,t+q}||^2.$$

In financial applications, the additional task-specific metrics (e.g., profit-and-loss measures or risk-adjusted scores) are often used.

Time-Series Generation Tasks. It aims to construct a generative model that produces synthetic time-series whose distribution closely matches that of the true time-series based on the observations. Suppose that we aim to learn the distribution of $X_{\text{seg}} := X_{t:t+q}$, which is assumed to be independent of t. The dataset in this case consists of samples $(X_{t:t+q})_{t\in\mathfrak{T}}$. The generative model is composed of the pre-specified noise distribution $\xi \in \mathcal{P}(E)$ and the generator, denoted by G_θ, is a function mapping from $E \to \mathbb{R}^{d\times q}$. Then, it induces the fake distribution $G_\theta(\xi)$. The task is to select an appropriate noise distribution and design and train a generator G_θ so that $G_\theta(\xi)$ produces high-fidelity samples of $X_{t:t+q}$. Evaluation metrics for this task may include statistical distance measures (e.g., Wasserstein distance, maximum mean discrepancy) or task-specific financial criteria.

3.2 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, in our work, we assume that the agent has access to three read-only external resources:

(1) **Case Bank** (denoted by E_{case}, textual data) as a collection of past tasks and reports summarizing successful methodologies.

(2) **Refinement Knowledge Bank** (denoted by $E_{\text{refinement}}$, textual data) as common practices in preprocessing, training optimization, and hyperparameter tuning and evaluation.

(3) **Code Base** (denoted by E_{code}), a repository containing model implementations of various models (model bank) and evaluation metrics (evaluation measures bank) (in `.py` format).

Given a new task T (see Example 1 as illustration) and at $t = 0$, the agent is given a Python template train.py (denoted by $\mathcal{S}_0$, see Example 2 as illustration) along with the above external resources. Using this information, the agent's objective is to complete the submodules of $\mathcal{S}_0$ so that it becomes executable, successfully trains the model, and produces valid model predictions with the goal of minimizing the corresponding loss function on the test dataset. For clarity, we use color coding to highlight information associated with key actions, which we defer to define in **§4**. Below, we will provide the details of each external source. An illustrative overview of these external sources is provided in Figure 2.

Case Bank. Similar to existing agentic systems [10], TS-Agent possesses a *Case Bank* $C = \{c_1, \ldots, c_{|C|}\}$ as a library consisting of the financial time-series tasks and the reports of successful methodologies. These cases could be derived from industry benchmarks, top-performing competition entries, and peer-reviewed academic papers on relevant financial data analysis (e.g., [3, 17–19]). It comprises two types of cases: time-series forecasting and time-series generation. Its tasks involve diverse financial datasets, such as historical stock prices, exchange rates, cryptocurrencies, and synthetic data. For instance, one task in our case bank is a task of generative time-series modelling on the cryptocurrency markets, where the recommended solution is a VAE-based model. Figure 2(a) shows an illustration of the case bank.

Refinement Knowledge Bank. A library of best practices and tips for improving the training of models. As illustrated in Figure 2(c), it contains guidance on selecting preprocessing methods, training strategies, optimization techniques, and evaluation metrics based on the logged training results. For example, a guidance entry under Learning Rate Scheduling might be: *Employ a learning rate scheduler*