FROM BARS TO TRADES: UNLEASHING THE POTENTIAL OF LLMS IN TICK-LEVEL ALPHA MINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Alpha mining plays a central role in quantitative research by discovering predictive signals from market data. Recent LLM-based approaches have demonstrated the potential of automating alpha mining by generating and refining factor expressions over predefined market features. However, their reliance on predefined features constrains the search space and thereby limits the discovery of predictive patterns that require constructing new variables from fine-grained market events. To address this limitation, we study alpha mining directly from tick-level trade data, formulating it as a hierarchical search over variable construction and factor composition. The enlarged search space makes it difficult to identify which changes improve predictive performance. To this end, we propose CARTS, an LLM-driven framework for tick-level alpha mining. CARTS integrates a tick-level DSL that separates variable construction from factor composition, evidence-guided search with controlled comparisons and research memory for fine-grained credit assignment, and multi-view evaluation beyond predictive performance. Together, these components enable efficient and evidence-driven factor exploration. Experiments show that directly adapting conventional alpha miners to tick data can degrade performance, while CARTS outperforms the strongest tick-level baseline by 88.6% in IC at the same 100-factor library size and generalizes across markets.

1 INTRODUCTION

Quantitative trading supports market liquidity and price discovery through data-driven decision-making and systematic execution (Hendershott et al., 2011). As a key component of quantitative strategy development, alpha mining (Zhang et al., 2025) aims to identify predictive signals from financial market data to inform trading decisions. Existing methods range from manual construction based on financial hypotheses to automated search using genetic programming and reinforcement learning (Chen et al., 2026; Shi et al., 2025; Yu et al., 2023). More recently, large language models (LLMs) have advanced factor research by leveraging financial knowledge and reasoning capabilities to formulate hypotheses, construct factor expressions, and iteratively refine candidate factors based on evaluation feedback (Duan et al., 2025; Han et al., 2026; Tang et al., 2025).

Existing LLM-driven alpha mining systems mainly operate on predefined market features, such as open, high, low, and close prices and trading volume (OHLCV) (Shi et al., 2026; Tang et al., 2026). They leverage LLM reasoning and evolutionary search to combine these features into candidate factors (Li et al., 2025; Liu et al., 2026; Wang et al., 2026). Under this paradigm, the construction of the underlying variables remains largely outside the scope of automated research. Consequently, the market hypotheses a system can explore are constrained by the information retained in the predefined features. For example, two time windows may have identical OHLCV values, yet large trades may cluster within a brief interval in one window while being dispersed throughout the

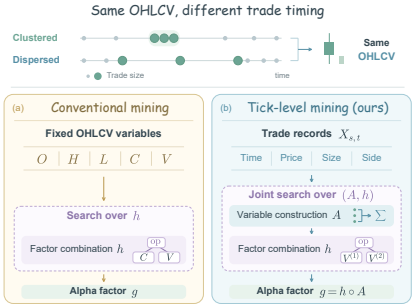


Figure 1: Comparison between conventional and tick-level mining. Jointly searching over variable and factor enlarges the search space.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

other. These aggregate features cannot distinguish between the two trading patterns, limiting the system's ability to test hypotheses related to trade clustering.

Distinguishing such patterns requires information about the timing and size of individual trades that window-level summaries do not retain as shown in Figure 1. Tick-level trade data provide this finer-grained view by recording individual transactions with their timestamps, execution prices, and trade sizes (Engle, 2000; Christensen et al., 2014). However, simply incorporating tick-level data into existing factor mining frameworks does not necessarily yield useful predictive signals and may even degrade performance, as shown in Figure 2. Exploiting such data requires more than access to finer-grained observations. We therefore investigate *alpha factor mining from tick-level trade data*, extending automated research from *how to combine existing features* to *what features should be constructed and combined from trade events*. Given a financial hypothesis, the system selects relevant trade events, specifies measurements, and designs aggregation rules to construct variables, which are then combined into factors. This formulation makes feature definition itself part of the research process, enabling joint search over variable construction and factor combination.

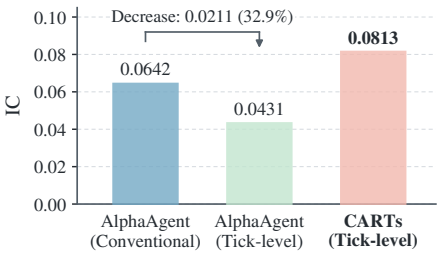


Figure 2: Performance of representative alpha mining methods on tick data. Existing LLM-driven methods exhibit performance degradation.

The joint selection of variable construction, and factor expressions creates a combinatorial search problem with three key challenges: how to represent tick-level factors for efficient computation, optimization, and search; how to explore the expanded search space using effective optimization signals; and how to gather sufficient evidence to assess factor quality. To address these challenges, we propose the **C**ollaborative **A**lpha **R**esearch Agent for **T**ick**s** (CARTs). First, CARTs introduces a tick-level domain-specific language (DSL) that supports flexible variable and factor construction while keeping computations explicit and verifiable, facilitating their joint optimization. Second, it uses experimental evidence for fine-grained *credit assignment* across choices in variable construction and factor expressions, and maintains a research memory to narrow subsequent search. Third, a multi-view evaluation mechanism assesses factor quality along complementary dimensions, providing evidence for candidate selection and refinement. Experiments on the Chinese A-share market show that CARTs consistently outperforms strong baselines and prior agentic systems, achieving state-of-the-art IC performance. In summary, our contributions are threefold:

(a) **New Problem Formulation.** We introduce tick-level alpha mining as a new formulation that extends automated alpha research from factor-expression search over predefined features to joint search over event-level variable construction and factor composition.

(b) **Hierarchical Mining Framework.** We propose CARTs framework, which integrates a tick-level DSL, joint search with evidence credit assignment, and multi-view evaluation to efficiently explore the hierarchical search space of tick-level alpha mining.

(c) **Comprehensive Empirical Validation.** Extensive experiments show that directly extending conventional alpha miners to tick data can degrade performance, highlighting the difficulty of tick-level alpha mining. CARTs outperforms the strongest tick-level baseline by 88.6% in IC at the same 100-factor library size, and demonstrates improved search efficiency, factor diversity, and robust generalization across markets.

2 Related Work

2.1 Alpha Mining

Alpha mining searches historical market data for predictive financial signals (Zhang et al., 2025). Earlier approaches primarily employ evolutionary search and reinforcement learning: AlphaEvolve (Cui et al., 2021) generates factors through evolution and pruning, while AlphaGen (Yu et al., 2023) optimizes the predictive utility of factor collections. More recently, LLMs have been used to translate financial hypotheses into factor expressions. Alpha-GPT (Wang et al., 2025) introduces human–AI interaction, AlphaAgent (Tang et al., 2025) combines semantic constraints with historical

feedback, and CogAlpha (Liu et al., 2026) and AlphaAgentEvo (Tang et al., 2026) further explore evolutionary search and reflective optimization. Complementing these studies, CARTs incorporates tick-level event aggregation into the search process, enabling an effective tick alpha mining.

2.2 NLP for Tabular Machine Learning

Language models provide a means of incorporating task semantics and domain knowledge into tabular learning (Yang et al., 2025; Zhuang et al., 2024; Tian et al., 2025; Du et al., 2024). LITO (Yang et al., 2024) and TabLLM (Hegselmann et al., 2023) translate structured data into language representations and use language models for prediction. CAAFE (Hollmann et al., 2023), OCTree (Nam et al., 2024) and FeatLLM (Han et al., 2024) instead employ LLMs to generate features for downstream predictors. CARTs builds on language-guided feature design and extends it to variable-length transaction sequences, jointly constructing event-aggregation rules and factor expressions. Financial semantic assessment, signal-redundancy evaluation, and checks for dependence on future observations provide feedback to guide the search.

3 CARTs Approach

3.1 Problem Formulation and System Overview

Problem Formulation. We study tick-level alpha factor mining problem, which aims to discover effective factors by jointly incorporating variable definitions and combination into the factor representation. At time t, let $\mathcal{S}_t$ denote the set of tradable stocks. For each stock $s \in \mathcal{S}_t$, let $X_{s,t} \in \mathbb{R}^{N_{s,t} \times d}$ denote the trade-by-trade records within the corresponding time window, where $N_{s,t}$ is the number of trades and d is the number of trade fields. Since trading activity varies, the number of trades may differ across stocks and time windows. Let $\mathcal{X}_{\leq t}$ denote all trade observations available across stocks up to time t.

A candidate factor consists of an event aggregation mapping A and a factor construction program h. The mapping A executes the required aggregation procedures across stocks and time windows to produce variables, which h subsequently transforms and combines. The complete factor and its output for stock s at time t are defined as

$$g = h \circ A, \qquad z_{s,t}^{(g)} = \left[h\left(A(\mathcal{X}_{\leq t})\right)\right]_{s,t} \tag{1}$$

This representation supports flexible variable definitions. Unlike expression search over a fixed feature set, joint search can revise both the factor construction program h and the aggregation mapping A, making variable definitions an integral part of the iterative research process.

Let $y_{s,t}^{(H)}$ denote the future return over a prediction horizon H. The predictive score of a candidate factor is

$$J(g) = \Phi\left(z_{s,t}^{(g)}, y_{s,t}^{(H)}\right) \tag{2}$$

where Φ is a scalar evaluation function, with higher values indicating better predictive performance. The mining objective is

$$g^\star = \arg\max_{g \in \mathcal{G}} J(g) \tag{3}$$

where $\mathcal{G}$ denotes the search space of valid factor programs. During mining, the system builds a library of accepted factors through iterative search, with acceptance determined by $J(g)$.

Key Challenges. Tick-level alpha mining incorporates variable construction into the mining process, requiring joint search over event aggregation A and factor construction h. Compared with conventional mining over predefined variables, this adds a search dimension and substantially increases the size and difficulty of the search space. This introduces three key challenges. First, **representing tick-level factors for efficient computation, optimization, and search**: the representation must support flexible event filtering, measurement, and aggregation while keeping computational logic and temporal dependencies explicit and inspectable. Second, **exploring the expanded search space with effective optimization signals**: each financial hypothesis admits many combinations of A and h, including redundant or unproductive candidates. Efficient exploration requires fine-grained credit assignment to guide and determine which choices should be revised. Third, **gathering sufficient**

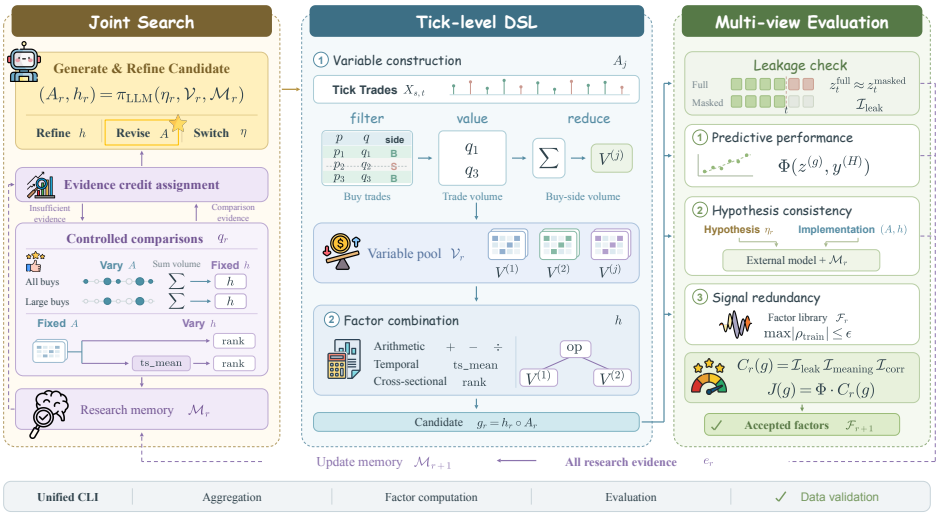


Figure 3: The overview of CARTs method.

evidence to assess factor quality: predictive scores alone cannot establish whether a factor is temporally valid, consistent with its financial hypothesis, or redundant with existing factors. Assessing these complementary aspects is necessary for candidate selection and refinement.

System Overview. CARTs addresses these challenges through a tick-level domain-specific language (DSL), joint search, and multi-view evaluation. The DSL separates variable A and factor construction h, enabling explicit, verifiable computations and independent revision. Joint search uses experimental evidence for fine-grained credit assignment to choices in A and h, research memory to narrow exploration, and targeted comparisons to resolve uncertain revision priorities. Multi-view evaluation checks for leakage and assesses predictive performance, consistency with financial hypotheses, and signal redundancy to guide factor selection and refinement.

As illustrated in Figure 3, the agent begins with a financial hypothesis, reuses existing variables or defines new aggregation programs, and constructs a factor expression using the DSL. A unified command-line interface (CLI) executes aggregation, factor computation, and evaluation through single-line commands (Appendix A.2). The resulting evidence is recorded in research memory and used for credit assignment, guiding revisions to A, h, or the underlying hypothesis. If the evidence does not distinguish among these options, the agent designs a targeted comparison before continuing the search. The CLI performs strict data validation to improve execution reliability, and its standardized interface reduces token overhead by avoiding repeated generation of auxiliary code.

3.2 Factor Representation: A Tick-Level Domain-Specific Language

Translating financial hypotheses into factors requires specifying which trade events to examine, how to measure and aggregate them, and how to combine the resulting variables. Efficient tick-level alpha mining therefore needs a representation that is easy to interpret and verify, while allowing targeted revisions during search. Unrestricted code permits arbitrary program structures, making event selection, aggregation logic, and temporal dependencies difficult to inspect and changes to individual components difficult to isolate. We therefore introduce a two-level tick-level domain-specific language (DSL) that represents variable construction separately from factor combination, making both stages explicit and independently revisable.

Variable Construction. Each construction program A_j consists of three components: filter_j, value_j, and reduce_j. For stock s and time window t, the resulting variable is

$$V_{s,t}^{(j)} = A_j(X_{s,t}) = \mathrm{reduce}_j\left(\mathrm{value}_j\left(\mathrm{filter}_j(X_{s,t})\right)\right). \tag{4}$$

Here, filter_j is represented as a structured Boolean predicate, where atomic comparison operators (e.g., eq, gt) are composed using logical operators such as and, or, and not, which selects trade events

that satisfy logical conditions, value_j extracts a specified trade measure, and reduce_j aggregates these values into a scalar feature. Applying the same program across stocks and time points yields the variable $V^{(j)} \in \mathcal{V}$, where $\mathcal{V}$ is the variable pool. For example, filtering for buy-side trades, extracting their volumes, and summing them yields the buy-side trading volume within the window. This formulation translates financial hypotheses into explicit rules for event selection, measurement, and aggregation, extending beyond combinations of existing features.

Factor Combination. Given the variable pool $\mathcal{V}$, the factor construction function h applies operators to transform and combine them into candidate factors. In the DSL, h is represented as an expression tree whose leaves are base features drawn from $\mathcal{V}$ and whose internal nodes are operators. Its recursive structure can be written as

$$h = \mathrm{op}(V^{(1)}, \ldots, V^{(k)}) \tag{5}$$

where op is a operator with k inputs, and each $V^{(i)}$ is a variable or a recursively subexpression. Example shows in Appendix B. Through the tick-level DSL, a tick factor can be represented as a single file, which is easy and convenient to verify and optimize via searching.

3.3 Factor Generation: Joint Search with Evidence Credit Assignment

The two-level DSL makes both variable definitions A and factor expressions h revisable, substantially enlarging the search space. To explore it efficiently, CARTs couples candidate refinement with targeted evidence acquisition. At its core is fine-grained credit assignment: the agent uses evaluation results and controlled comparisons to identify whether a candidate's limitations stem from event selection, measurement, and aggregation, or from its downstream expression. When existing evidence cannot distinguish these sources, the agent designs experiments to resolve the uncertainty.

The search maintains a variable pool $\mathcal{V}_r$, a factor library $\mathcal{F}_r$ of factors accepted before iteration r, and research memory $\mathcal{M}_r$. The memory links financial hypotheses to aggregation programs, factor expressions, and experimental outcomes. It records variable statistics, reusable findings, failed attempts, unresolved questions, and potential refinements. These states support variable reuse, factor accumulation, and subsequent search decisions.

Candidate Generation and Refinement. At iteration r, the agent proposes a financial hypothesis η_r and generates a candidate conditioned on the variable pool and research memory:

$$(A_r, h_r) = \pi_{\mathrm{LLM}}(\eta_r, \mathcal{V}_r, \mathcal{M}_r), \qquad g_r = h_r \circ A_r. \tag{6}$$

Here, A_r contains all aggregation definitions referenced by h_r, whether reused or newly constructed. Based on the available evidence, the agent can take three actions: **expression refinement** adjusts h while retaining the underlying variables; **measurement revision** changes event selection, measurement, or aggregation under the same hypothesis; and **direction switching** explores another hypothesis when progress stalls. The search can thus revisit variable definitions rather than repeatedly combining unsuitable measurements.

Credit Assignment through Controlled Comparisons. A predictive score evaluates a candidate (A, h) as a whole, but does not reveal which choices in event measurement A or downstream composition h contribute to its success or failure. CARTs uses multi-view evaluation and controlled comparisons to assign credit to individual design choices rather than treating the candidate as an indivisible unit. When existing evidence is insufficient for attribution, the agent designs a targeted experiment q_r that specifies the unresolved question, compared configurations, and controlled conditions; otherwise, it directly uses the available evidence to guide refinement.

For example, when comparing event measurements for a buying-pressure candidate under a fixed expression, observed improvements or degradations provide evidence for the modified choices in A, conditional on the retained h. Conversely, comparing expression variants while keeping the measurements fixed attributes their relative effects to the modified choices in h, conditional on the retained A. Based on such comparisons, the agent retains supported choices, revises underperforming ones, and determines whether subsequent refinement should focus on measurement or composition.

The resulting comparative evidence and component-level attributions are recorded in research memory. When the evidence is inconclusive, the corresponding attribution remains unresolved and may

trigger additional controlled comparisons as the candidate evolves. Supplementary experiments count toward the search budget and do not alter the final factor-acceptance criteria.

Research Memory and State Updates. Candidate evaluation and supplementary experiments produce evidence e_r, including predictive metrics, check outcomes, comparative observations, and diagnostic feedback. Together with the resulting credit assignment, this evidence is incorporated into research memory as

$$\mathcal{M}_{r+1} = \mathrm{Update}(\mathcal{M}_r, \eta_r, A_r, h_r, q_r, e_r), \tag{7}$$

where the updated memory records which design choices are supported, disfavored, or remain unresolved under the current evidence. These states guide subsequent refinement: supported components can be retained, underperforming components can be selectively revised, and unresolved components can motivate further controlled comparisons.

Newly constructed variables are added to $\mathcal{V}_{r+1}$, while accepted factors are added to $\mathcal{F}_{r+1}$, allowing useful variables to be reused even when the factor in which they first appeared is not accepted. In this way, research memory carries forward both reusable constructions and component-level evidence, enabling subsequent iterations to refine A and h without repeatedly reconsidering already resolved design choices.

3.4 Factor Evaluation: Multi-view Evaluation Mechanism

Joint search requires evidence about whether a candidate's computation is consistent with its financial hypothesis, whether its signal overlaps with existing signals and so on. A single predictive metric cannot adequately support these judgments. To inform subsequent decisions, CARTS evaluates candidate factors from three perspectives: predictive performance, financial hypothesis consistency, and signal redundancy, alongside a check for future information leakage.

Leakage Check. A factor's output at time t must depend only on information available up to that time. To check whether a candidate implementation satisfies this constraint, we compare its values computed on unmasked data, $z_{s,t}^{(g),\mathrm{full}}$, with those recomputed after masking subsequent observations, $z_{s,t}^{(g),\mathrm{masked}}$. If the checked values agree within numerical tolerance, then $\mathcal{I}_{\mathrm{leak}}(g) = 1$; otherwise, it is zero. A discrepancy indicates possible dependence on future observations and warrants further review.

Predictive Performance. The evaluation function Φ defined in Section 3.1 is used to assess the candidate's predictive performance. A unified evaluation protocol allows different base feature definitions and factor expressions to be compared against the same future return target, providing a quantitative basis for candidate selection and revision.

Financial Hypothesis Consistency. An external model uses the context in research memory $\mathcal{M}_r$ to assess whether the event aggregation logic and factor expression reasonably implement the hypothesis η_r. If the candidate passes this assessment, then $\mathcal{I}_{\mathrm{meaning}}(g; \eta_r, \mathcal{M}_r) = 1$; otherwise, it is zero. Because base feature definitions are also part of the search space, this assessment considers both event-level measurements and downstream expressions, rather than examining only the final feature combination. The resulting feedback helps identify potential inconsistencies between the intended financial mechanism and its computational implementation and supports revisions at either DSL level.

Signal Redundancy. The candidate factor is compared with signals in the accepted factor library $\mathcal{F}_r$. Let $\rho_{\mathrm{train}}(g, q)$ denote the signal correlation statistic computed over aligned training observations. The redundancy check is defined as

$$\mathcal{I}_{\mathrm{corr}}(g; \mathcal{F}_r) = \mathbf{1}\left\{\max_{q \in \mathcal{F}_r} |\rho_{\mathrm{train}}(g, q)| \le \epsilon\right\}, \tag{8}$$

where $\epsilon \in [0, 1]$ is a similarity threshold. This check filters out candidates that substantially overlap with existing signals, encouraging factor diversity.

Table 1: Factor predictive power and strategy performance of factor mining methods. Best results in each metric are highlighted in bold.

Methods		Factor Predictive Power				Strategy Performance		
		IC	**ICIR**	**Rank IC**	**Rank ICIR**	**IR**	**ARR (%)**	**MDD (%)**
Classical Methods								
Machine Learning	Linear	0.0266	0.29	0.0541	0.51	-0.33	-5.92	31.89
	LightGBM	0.0260	0.27	0.0491	0.42	-0.84	-7.37	15.38
	MLP	0.0395	0.49	0.0534	0.52	0.11	1.78	10.41
	CatBoost	0.0486	0.39	0.0699	0.44	0.07	0.93	14.59
	XGBoost	0.0378	0.38	0.0541	0.43	-0.17	-1.56	19.17
Deep Learning	LSTM	0.0616	0.51	0.0921	0.63	0.67	8.04	10.27
	GRU	0.0561	0.52	0.0666	0.51	0.81	8.01	8.36
	Transformer	0.0433	0.50	0.0561	0.54	0.52	6.70	10.09
	TRA	0.0213	0.28	0.0432	0.47	-0.21	-1.70	10.92
Factor Libraries	Alpha101	0.0715	0.67	0.0927	0.73	1.68	19.07	8.05
	Alpha158	0.0593	0.66	0.0603	0.52	0.96	11.86	10.56
	Alpha360	0.0561	0.60	0.0567	0.49	0.65	6.43	**7.84**
LLM-Based Factor Mining Methods								
LLM-Based	AlphaAgent	0.0642	0.71	0.0803	0.78	0.76	7.66	14.20
	RD-Agent	0.0573	0.42	0.0776	0.47	0.47	4.43	8.93
	FactorMiner	0.0751	0.64	**0.0997**	0.65	1.46	19.87	11.46
CARTs (OHLCV)		**0.0789**	**0.74**	0.0994	**0.79**	**1.96**	**30.75**	14.38

Candidate Acceptance and Research Feedback. Under a hard-constraint screening scheme, the three checks are combined as

$$C_r(g) = \mathcal{I}_{\text{leak}}(g) \cdot \mathcal{I}_{\text{meaning}}(g; \eta_r, \mathcal{M}_r) \cdot \mathcal{I}_{\text{corr}}(g; \mathcal{F}_r). \tag{9}$$

The corresponding constrained predictive objective is

$$J(g) = \Phi\left(z_{s,t}^{(g)}, y_{s,t}^{(H)}\right) \cdot C_r(g). \tag{10}$$

A minimum acceptance threshold determines whether a factor enters the library. Meanwhile, the feedback e_r retains individual metrics and diagnostic results so that different failure modes can inform different research decisions.

4 Experiments

We evaluate CARTs against existing methods along effectiveness, scalability, and generalization as follows. We further conduct ablation studies to understand the contributions of its key components.

RQ1 (Effectiveness): Can CARTs outperform existing factor mining methods and effectively exploit tick-level transaction data?

RQ2 (Scalability): Can CARTs continue to discover useful factors and improve predictive performance as the factor library grows?

RQ3 (Generalization): Can CARTs discover factors that remain predictive across markets?

4.1 Experimental Setup

We conduct the main experiments on All Chinese A-share and assess cross-market generalization on SP1000 and JPX. The training, validation, and test periods are January 2022–September 2023, October–December 2023, and 2024–2025, respectively. We predict ten-day returns in the main experiments and one-day returns in tick-level experiments. Using DeepSeek-V4 (Xu et al., 2026) as its backbone, CARTs mines 500 factors: 100 from OHLCV alone and 400 incorporating transaction-level data. We train LightGBM on these factors and evaluate predictions using IC, ICIR, RankIC,

Table 2: Comparison between OHLCV and Tick inputs with 100 factors of CARTs and comparison methods.

Method	Input	IC	ICIR	Rank IC	Rank ICIR
AlphaAgent	OHLCV	+0.0651	+0.70	+0.0814	+0.77
	Tick	+0.0431	+0.53	+0.0477	+0.59
	Δ	−0.0220	−0.17	−0.0337	−0.18
RD-Agent	OHLCV	+0.0589	+0.44	+0.0780	+0.50
	Tick	+0.0354	+0.39	+0.0361	+0.42
	Δ	−0.0235	−0.05	−0.0419	−0.08
CARTs	OHLCV	+0.0794	+0.82	+0.0744	+0.73
	Tick	+0.0813	+0.86	+0.0732	+0.87
	Δ	+0.0019	+0.04	−0.0012	+0.14

Table 3: Performance of CARTs with an increasing number of Tick factors with OHLCV baseline in the first row. The best results are highlighted in bold. The results show that CARTs consistently improves as more Tick factors are mined.

Input	# Factors	IC	ICIR	Rank IC	Rank ICIR
OHLCV	100	+0.0794	+0.82	+0.0744	+0.73
Tick	100	+0.0813	+0.86	+0.0732	+0.87
	200	+0.0853	+0.88	+0.0760	+0.89
	300	+0.0883	+0.90	+0.0822	+0.91
	400	+0.0898	+0.91	+0.0837	+0.92
	500	**+0.0917**	**+0.92**	**+0.0855**	**+0.93**

and RankICIR. Strategy performance is measured by IR, ARR, and MDD under a TopK strategy ($K = 50$, dropping at most five stocks per trading step).

Baselines include machine learning models (Ke et al., 2017; Chen & Guestrin, 2016; Prokhorenkova et al., 2018; Vaswani et al., 2017; Cho et al., 2014; Hochreiter & Schmidhuber, 1997; Lin et al., 2021) using Alpha20 inputs, established factor libraries (Kakushadze, 2016; Yang et al., 2020), and automated mining frameworks: R&D-Agent-Quant (Li et al., 2025), AlphaAgent (Tang et al., 2025), and FactorMiner (Wang et al., 2026). LightGBM hyperparameters are provided in the Appendix A.

4.2 Main Results

4.2.1 Results on **RQ1. Effectiveness**

Under the standard OHLCV setting, CARTs achieves the best results on five of seven metrics, including IC, ICIR, Rank ICIR, IR, and ARR, while maintaining a competitive Rank IC, as shown in Table 1. Its ARR reaches 30.75%, compared with 19.87% for FactorMiner, demonstrating strong performance even with conventional market inputs.

We further investigate whether alpha mining systems can effectively exploit tick-level transaction data with 1-day return prediction. Since finer-grained inputs do not necessarily translate into better factors, we compare OHLCV and Tick settings while fixing the number of mined factors to 100 for all methods, and further examine the predictive quality of the discovered Tick factors.

As shown in Table 2, simply replacing OHLCV with Tick data does not necessarily improve factor quality for existing mining systems. Both AlphaAgent and RD-Agent suffer performance degradation across all four metrics when using Tick inputs. In contrast, CARTs achieves improvements in IC, ICIR and Rank ICIR metrics, while maintaining comparable Rank IC. These results suggest that the benefit of tick-level data does not arise merely from changing the inputs. Effectively exploiting such data requires the mining system to construct and search informative variables from transaction-level events, which CARTs is specifically designed to support.Moreover, CARTs improves IC from 0.0431 to 0.0813 over the strongest baseline, i.e., AlphaAgent, corresponding to a relative improvement of 88.6% under the same 100-factor Tick setting.

We further examine the predictive quality of the individual Tick factors discovered by CARTs. As shown in Figure 4, the discovered factors are concentrated around IC values of 0.01–0.03, with a mean of 0.0212 and a median of 0.0183, while the strongest factors exceed 0.04 and reach up to 0.0560. The positive median and moderately right-skewed distribution indicate that CARTs discovers a broad set of informative factors, together with a smaller subset of strong signals.

4.2.2 Results on **RQ2. Scalability**

We further examine the scalability of CARTs by increasing the number of Tick factors from 100 to 500, with the 100-factor OHLCV setting included as a comparison baseline.

As shown in Table 3, performance improves monotonically across all four metrics as more Tick factors are mined. In particular, IC increases from 0.0813 to 0.0917, while Rank IC increases from 0.0732 to 0.0855 as the factor budget grows from 100 to 500. These consistent gains indicate that

Table 4: Cross-market comparison of factor predictive power.

Market	Method	IC	ICIR	Rank IC	Rank ICIR
SP1000	Alpha156	0.0155	0.24	0.0032	0.05
	Alpha360	0.0052	0.07	0.0033	0.05
	FactorMiner	0.0177	0.31	0.0082	0.13
	CARTs	**0.0283**	**0.43**	**0.0193**	**0.28**
JPX	Alpha156	0.0327	0.49	**0.0526**	**0.98**
	Alpha360	0.0240	0.38	0.0362	0.69
	FactorMiner	0.0330	0.65	0.0247	0.54
	CARTs	**0.0340**	**0.71**	0.0374	0.92

Table 5: Ablation study results of CARTs. Results are obtained using 1-day returns.

Metric	w/o JEC	w/o MVE	CARTs
Token/10 Alphas	4.01M	**0.49M**	3.03M
IC	0.0744	0.0523	**0.0917**
ICIR	0.89	0.72	**0.92**
Rank IC	0.0791	0.0739	**0.0855**
Rank ICIR	0.80	0.77	**0.93**

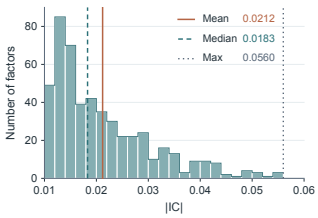


Figure 4: IC distribution of individual alphas discovered by CARTs, showing consistently positive predictive signals.

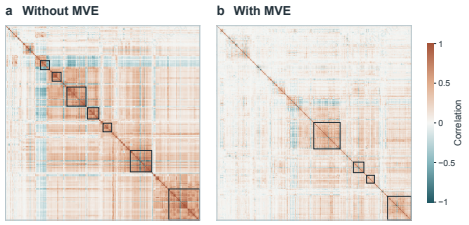


Figure 5: Pairwise alpha correlations with and without MVE. MVE reduces correlated clusters and improves factor diversity.

CARTs continues to discover complementary predictive signals as the mining budget increases, rather than quickly saturating within the evaluated range.

4.2.3 Results on **RQ3. Generalization**

To further assess the generalization of CARTs, we evaluate whether the discovered factors can transfer to unseen markets with different market characteristics. We directly apply the factors mined by CARTs to two additional markets, SP1000 and JPX, without re-mining them on the target markets. As shown in Table 4, CARTs achieves the highest IC and ICIR on both markets while remaining competitive on rank-based metrics. These results indicate that the discovered factors capture predictive patterns that are not specific to the source market and can generalize across different markets.

4.3 Ablation Study

We ablate the two key components of CARTs, joint search with evidence credit assignment (JEC) and multi-view evaluation (MVE). As shown in Table 5, removing JEC increases the token cost per 10 alphas from 3.03M to 4.01M and degrades all four predictive metrics. This shows that component-level credit assignment reduces redundant exploration and improves factor refinement efficiency. Moreover, removing the plausibility and correlation checks reduces evaluation cost but degrades all four predictive metrics as shown in Table 5. Figure 5 further shows that MVE reduces highly correlated factor clusters, leading to more diverse and complementary factors.

5 Conclusion

In this work, we present CARTs, an LLM-driven system for joint variable and factor construction from tick-level trade data. It integrates a two-level DSL for efficient computation and joint optimization, a search strategy guided by fine-grained credit assignment and research memory, and multi-view evaluation for candidate selection and refinement. Experiments demonstrate strong predictive performance and favorable generalization and robustness across markets and prediction horizons, highlighting the value of incorporating variable construction into automated alpha mining.

AI USE STATEMENT

During manuscript preparation, generative AI tools were used to assist with English translation, enhance the language, and improve the presentation of figures and tables. Their contributions included refining wording, enhancing clarity, and adjusting captions and layouts. The authors bear full responsibility for the paper's technical content, results, citations, and conclusions, together with the accuracy and integrity of the final manuscript.

REFERENCES

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. Alphasage: Structure-aware alpha mining via gflownets for robust exploration. In *International Conference on Learning Representations*, volume 2026, pp. 2955–2982, 2026.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

Kim Christensen, Roel CA Oomen, and Mark Podolskij. Fact or friction: Jumps at ultra high frequency. *Journal of Financial Economics*, 114(3):576–599, 2014.

Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. Alphaevolve: A learning framework to discover novel alphas in quantitative investment. In *Proceedings of the 2021 International conference on management of data*, pp. 2208–2216, 2021.

Siyi Du, Shaoming Zheng, Yinsong Wang, Wenjia Bai, Declan P O'Regan, and Chen Qin. Tip: Tabular-image pre-training for multimodal classification with incomplete data. In *European Conference on Computer Vision*, pp. 478–496. Springer, 2024.

Yitong Duan, Chuheng Zhang, and Jian Li. Factormad: A multi-agent debate framework based on large language models for interpretable stock alpha factor mining. In *Proceedings of the 6th ACM International Conference on AI in Finance*, pp. 605–613, 2025.

Robert F Engle. The econometrics of ultra-high-frequency data. *Econometrica*, 68(1):1–22, 2000.

Jun Han, Shuo Zhang, Wei Li, Yifan Dong, Tu Hu, Yumo Zhu, Xiaomin Yu, Xin Guo, Zhaowei Liu, Kunyi Wang, et al. Quantaalpha: An evolutionary framework for llm-driven alpha mining. *arXiv preprint arXiv:2602.07085*, 2026.

Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. *arXiv preprint arXiv:2404.09491*, 2024.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International conference on artificial intelligence and statistics*, pp. 5549–5581. PMLR, 2023.

Terrence Hendershott, Charles M Jones, and Albert J Menkveld. Does algorithmic trading improve liquidity? *The Journal of finance*, 66(1):1–33, 2011.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36:44753–44775, 2023.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Yuante Li, Xu Yang, Xiao Yang, Xisen Wang, Weiqing Liu, and Jiang Bian. R&d-agent-quant: A multi-agent framework for data-centric factors and model joint optimization. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/ac5c2b6e423883cbcacbcccf88491b78-Abstract-Datasets_and_Benchmarks_Track.html.

Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1017–1026, 2021.

Fengyuan Liu, Yi Huang, Sichun Luo, Yuqi Wang, Yazheng Yang, Xinye Li, Zefa Hu, Junlan Feng, and Qi Liu. Cognitive alpha mining via llm-driven code-based evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11715–11749, 2026.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized feature generation for tabular data via llms with decision tree reasoning. *Advances in Neural Information Processing Systems*, 37:92352–92380, 2024.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Angel Seco. Alphaforge: A framework to mine and dynamically combine formulaic alpha factors. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 12524–12532, 2025.

Yu Shi, Yitong Duan, and Jian Li. Navigating the alpha jungle: An llm-powered mcts framework for formulaic alpha factor mining. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 997–1005, 2026.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. Alphaagent: Llm-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2813–2822, 2025.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. Alphaagentevo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Shi-Yu Tian, Zhi Zhou, Wei Dong, Kun-Yang Yu, Ming Yang, Zi-Jian Cheng, Lan-Zhe Guo, and Yu-Feng Li. Tabularmath: Understanding math reasoning over tables with large language models. *arXiv preprint arXiv:2505.19563*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Saizhuo Wang, Hang Yuan, Leon Zhou, Lionel Ni, Heung Yeung Shum, and Jian Guo. Alphagpt: Human-ai interactive alpha mining for quantitative investment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 196–206, 2025.

Yanlong Wang, Jian Xu, Hongkang Zhang, Shao-Lun Huang, Danny Dongning Sun, and Xiao-Ping Zhang. Factorminer: A self-evolving agent with skills and experience memory for financial alpha discovery. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 12327–12338, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

June Yong Yang, Geondo Park, Joowon Kim, Hyeongwon Jang, and Eunho Yang. Language-interfaced tabular oversampling via progressive imputation and self-authentication. In *The Twelfth International Conference on Learning Representations*, 2024.

Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

Xiao-Wen Yang, Jie-Jing Shao, Lan-Zhe Guo, Bo-Wen Zhang, Zhi Zhou, Lin-Han Jia, Wang-Zhou Dai, and Yu-Feng Li. Neuro-symbolic artificial intelligence: Towards improving the reasoning abilities of large language models. *arXiv preprint arXiv:2508.13678*, 2025.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5476–5486, 2023.

Junjie Zhang, Shuoling Liu, Tongzhe Zhang, and Yuchen Shi. A survey on large language model-based alpha mining. *Frontiers of Information Technology & Electronic Engineering*, 26(10): 1809–1821, 2025.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W Huang, Jie Fu, Xiang Yue, and Wenhu Chen. Structlm: Towards building generalist models for structured knowledge grounding. *arXiv preprint arXiv:2402.16671*, 2024.

A EXPERIMENT DETAIL

A.1 ADDITIONAL EXPERIMENTAL DETAILS

This appendix supplements the experimental setup described in the main text. Our experiments are primarily conducted on three equity universes: All A-share, SP1000, and JPX.

Implementation details of CARTs. We optimize CARTs using the objective defined in Eq. (10). A candidate factor is accepted into the factor library only when its factor score is strictly greater than 0.01. The portfolio rebalancing schedule is aligned with the prediction horizon, with the rebalancing interval set equal to the forecast horizon. In our main experiments, we adopt a standard transaction-cost setting with a buying cost of 0.05% and a selling cost of 0.15%. We use IC as the predictive function Φ.

Hyperparameter search. For each baseline, we perform grid search over the Cartesian product of its candidate hyperparameter values, while keeping the remaining settings fixed. The search spaces and model-specific settings are described below.

Training protocol for gradient-boosted trees. For LightGBM, XGBoost, and CatBoost, we search the learning rate over $\{0.02, 0.05, 0.1\}$. Each configuration is trained for a maximum of 3,000 boosting rounds, with early stopping triggered if the validation metric does not improve for 100 consecutive rounds.

LightGBM. In addition to the learning rate, we search the following parameters:

$$\texttt{num_leaves} \in \{31, 63, 127\},$$
$$\texttt{min_data_in_leaf} \in \{500, 1000, 2000\},$$
$$\texttt{lambda_l2} \in \{1, 5, 10\}.$$

We fix `max_bin` to 63, both `feature_fraction` and `bagging_fraction` to 0.8, and `bagging_freq` to 1. Including the learning-rate grid, this yields 81 configurations.

XGBoost. In addition to the learning rate, we search

$$\texttt{max_depth} \in \{4, 6, 8\},$$
$$\texttt{min_child_weight} \in \{10, 50, 100\},$$
$$\texttt{reg_lambda} \in \{1, 5, 10\}.$$

We fix both `subsample` and `colsample_bytree` to 0.8 and use the histogram-based tree construction method (`tree_method=hist`). This yields 81 configurations.

CatBoost. We search `depth` over $\{4, 6, 8\}$ and `l2_leaf_reg` over $\{1, 5, 10\}$. Together with the learning-rate grid, this yields 27 configurations.

Training settings for neural baselines. For LSTM, GRU, Transformer, and TRA, we search the learning rate over

$$\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}.$$

For LSTM, GRU, and Transformer, we additionally search the dropout rate over $\{0.1, 0.2, 0.3\}$. The remaining training settings are kept fixed across candidate configurations.

LSTM and GRU. Both recurrent baselines use two recurrent layers. We search the hidden dimension over $\{32, 64, 128\}$ and apply the searched dropout rate between recurrent layers. The hidden representation at the final time step is mapped to a scalar prediction through a linear layer. Each recurrent architecture has 27 configurations, and LSTM and GRU are searched separately.

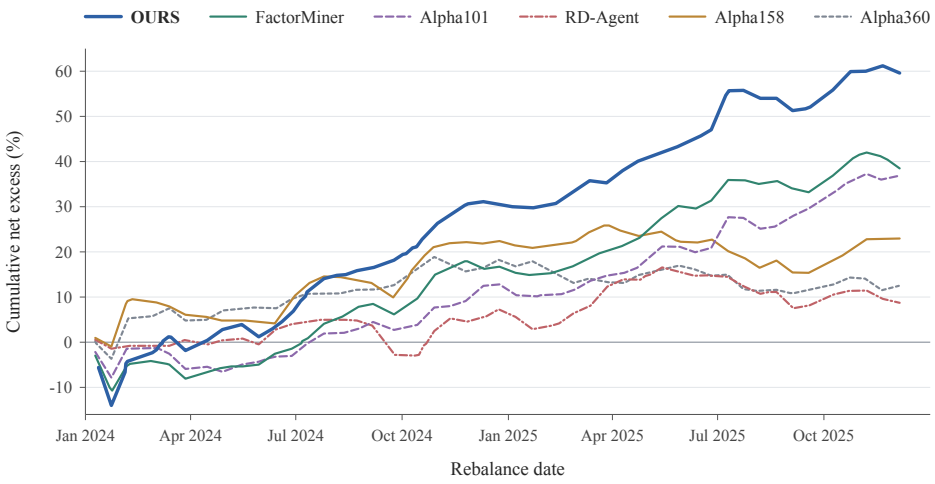


Figure 6: Cumulative net excess returns in the Chinese A-share market under the TopK Dropout strategy ($K = 50$, $\text{dropout} = 5$) with a 10-day prediction horizon. CARTS achieves the highest cumulative net excess return at the end of the evaluation period.

Transformer. We use a two-layer Transformer with four attention heads, pre-layer normalization, and learnable positional embeddings. The model dimension is searched over

$$d_{\text{model}} \in \{32, 64, 128\}.$$

The feed-forward dimension is set to $4d_{\text{model}}$ and is therefore adjusted jointly with the model dimension rather than searched independently. The representation at the final time step is used for prediction. Together with the dropout and learning-rate grids, this yields 27 configurations.

TRA. TRA uses a two-layer GRU backbone with a hidden dimension of 64. We search the number of prediction heads and the Sinkhorn parameter over

$$K \in \{2, 3, 5\}, \qquad \epsilon \in \{0.05, 0.1, 0.2\}.$$

For each choice of K, the router architecture is adjusted to $(64+K) \to 32 \to K$. We fix the number of Sinkhorn iterations to 20 and use the hard-assignment weight schedule

$$\rho_e = 0.99^e,$$

where e denotes the training epoch. TRA uses a separate training loop, while sharing the remaining training hyperparameters with the other neural baselines. Combining the prediction-head, Sinkhorn-parameter, and learning-rate grids gives 27 configurations.

Across all baseline model types, these search spaces contain 330 configurations, counting LSTM and GRU separately and excluding repetitions across datasets or random seeds.

Additional Strategy Performance To compare the performance of different methods, we visualize their cumulative net excess returns in the Chinese A-share market using a TopK Dropout strategy ($K = 50$, $\text{dropout} = 5$) with a 10-trading-day prediction horizon. As shown in Figure 6, CARTS achieves the highest cumulative net excess return at the end of the evaluation period, outperforming all baselines.

A.2 UNIFIED COMMAND-LINE INTERFACE

The `cartcli` interface supports variable construction, factor evaluation, and result management through the commands summarized in Table 6.

Table 6: The `cartcli` command interface.

Command	**Stage**	**Function**
`cartcli init`	Setup	Initialize a run and its configuration.
`cartcli subfactor`	Variable construction	Build reusable variables.
`cartcli eval`	Mining	Evaluate candidates on the validation set.
`cartcli accept`	Storage	Save accepted factors and evaluation records.
`cartcli show`	Lookup	Inspect operators, variables, and stored factors.
`cartcli help`	Help	Display command help.

B AN EXAMPLE OF ALPHA

In this section, we present three representative alpha factors discovered by CARTs, expressed in our tick-level DSL. For each factor, we report its expression and, where applicable, the underlying variable-construction programs, specifying their `filter`, `value`, and `reduce` components. These examples illustrate how the DSL captures both the construction of variables from tick-level events and their subsequent combination into alpha signals. Factors based on existing inputs, such as `close`, reuse these variables without additional aggregation.

1. psurvival_std_asymmetry_neg_ewm3.

```
ts_ewm(signlog(psurvival_std_sell - psurvival_std_buy), 3)
```

Variable construction:

```
psurvival_std_buy:
  filter: dir = 1
  value:  (ts_ms - passive_order_ts) / 1000
  reduce: std

psurvival_std_sell:
  filter: dir = -1
  value:  (ts_ms - passive_order_ts) / 1000
  reduce: std
```

2. bollinger_reversal_k24.

```
-ts_zscore(close, 24)
```

This factor directly uses `close` and requires no additional variable-construction program.

3. ofi_reversal_k6.

```
cs_rank(-ts_decay_linear(net_buy_amt / total_trade_amt, 6))
```

Variable construction:

```
net_buy_amt:
  filter: is_cancel = 0
  value:  amt * dir
  reduce: sum

total_trade_amt:
  filter: is_cancel = 0
  value:  amt
  reduce: sum
```