

Granger-Guided Autonomous Prompt Optimization for Agentic Financial Feature Engineering

Alperen Cengiz Ozturk^{*†}, David Forino[†], Ismail Ari^{*}

^{*}Özyeğin University, Istanbul, Türkiye

[†]Quanted Technologies Ltd., London, United Kingdom

Abstract—Feature Engineering remains a major bottleneck in financial machine learning pipelines, because useful predictors often require both domain knowledge and repeated empirical validation. Large Language Model (LLM) agents can generate context-aware financial features from dataset metadata, documentation, and research context, but their outputs depend strongly on prompt design and may produce weak or spurious signals without measurable feedback. We present a closed-loop agentic feature-engineering framework in which an LLM-based pipeline generates candidate financial time-series features, a fixed Granger-based evaluation framework scores their rolling-window predictive signals, and an autonomous prompt-optimization loop edits the feature-generation prompt using this statistical feedback. Across 5 alternative-data datasets with 10 tickers each, 56 attempted iterations produced 7 best-so-far prompt improvements. The internal composite score increased from 0.0631 to 0.0754, a relative improvement of 19.5%. These results indicate stronger predictive relevance of signals under the fixed evaluator, without implying trading profitability or structural causality.

Index Terms—Agentic AI, automated feature engineering, prompt optimization, Granger causality, financial time series, statistical feedback.

I. INTRODUCTION

Feature Engineering (FE) remains a major bottleneck in financial engineering, because useful predictors often encode domain-specific hypotheses about which information signals may help predict future returns. Designing such features manually requires knowledge of the data source, the market setting, and the statistical behavior of the target variable. It is also iterative: practitioners (*a.k.a* quants) propose a feature, implement it, align it with the prediction target, evaluate its signal, and revise the hypothesis when the evidence is weak.

Automated FE reduces part of this burden by generating and evaluating candidate transformations at scale. However, many existing approaches search over predefined mathematical transformations, relational aggregations, or model-driven feature selection criteria. These methods can improve scalability, but they may remain weakly connected to the semantic meaning of the dataset. In finance, this limitation matters because a useful feature is not only a transformation of columns, but also a testable hypothesis about how an information channel may reflect future returns.

LLM agents provide a complementary direction. An agentic workflow can inspect the metadata of the dataset, summarize vendor-provided feature documentation, retrieve relevant external research context, propose feature definitions, and generate executable code. Tool-using LLM agents have shown

that language models can combine reasoning with external actions such as search, retrieval, and code execution [1], while retrieval-augmented generation (RAG) enables external knowledge to be incorporated into the model context [2]. For FE, this creates the possibility of context-aware feature generation: the system can use both dataset structure and domain knowledge to propose compact, interpretable feature hypotheses.

However, LLM-generated features still require disciplined validation. A generated feature may sound plausible but have little predictive relevance. In this work, we use Granger causality [3] as a predictive-relevance diagnostic: a feature is useful when its lagged values help explain future returns beyond the target’s own lagged values under the specified test design. We interpret this as evidence of temporal predictive relevance, not as proof of structural or economic causality.

A second challenge is prompt quality. In an LLM-based feature generation pipeline, the prompt strongly shapes which signal families the agent considers, how it uses retrieved context, and how it balances interpretability against statistical testability. Manual prompt tuning is subjective and expensive because prompt quality can only be observed after running the downstream generation, implementation, and evaluation steps. This creates a closed-loop optimization problem: if generated features can be scored statistically, then the prompt that generates those features can also be optimized using that score as feedback.

We present an autonomous prompt-optimization framework for agentic financial FE. The framework combines three components: an agentic pipeline that generates executable financial time-series features, a fixed Granger-based evaluator that scores rolling-window predictive-relevance signals, and an autoresearch loop that edits only the feature-generation prompt. The dataset suite, feature execution logic, evaluator, and scoring rule remain fixed during optimization, so accepted improvements are attributable to changes in the prompt rather than changes in the evaluation procedure.

The paper makes three contributions:

- 1) An agentic FE pipeline that combines dataset profiling, research-context retrieval, feature proposal, and code generation for financial time-series data (Sec.3.A);
- 2) A fixed rolling-window Granger-based feedback framework that evaluates generated features through predictive-relevance signals (Sec.3.B, Sec.4.B), and

- 3) An autonomous prompt-optimization loop that improves the feature-generation prompt using the Granger composite score (Sec.3.C, Sec.4.C).

II. RELATED WORK

A. Automated Feature Engineering

Automated FE aims to reduce the manual effort required to construct useful input representations for ML models. Deep Feature Synthesis introduced an influential approach based on relational aggregation and transformation primitives [4]. Other systems, such as AutoFeat and OpenFE, expand this direction by searching over mathematical transformations or candidate feature operators and selecting useful features according to the downstream predictive performance [5], [6]. More broadly, automated FE is often treated as part of the AutoML pipeline, where feature construction, model selection, and hyperparameter optimization are automated to reduce human trial-and-error [7].

These methods show that feature construction can be effectively automated, but they usually operate over predefined transformation spaces. As a result, they may have limited access to the semantic meaning of a dataset or to external domain knowledge about why a particular signal should be predictive. Our work is complementary: instead of only searching over a fixed operator library, we use an LLM-based agent to propose domain-aware financial feature hypotheses and then evaluate those hypotheses with a fixed statistical feedback mechanism for fairness.

B. LLM Agents and Retrieval-Augmented Workflows

Tool-using LLM agents have shown that language models can combine reasoning with external actions such as search, retrieval, and code execution [1]. Retrieval-augmented generation (RAG) similarly improves knowledge-intensive generation by conditioning model outputs on retrieved documents rather than relying only on parametric memory [2]. These ideas are relevant to FE because useful features may require understanding both the dataset structure and the external context of the financial signal.

In our framework, the agentic component follows this direction by combining dataset profiling, research-context retrieval, feature proposal, and code generation. The central focus of this paper, however, is not only feature generation with an LLM agent. It is the closed-loop setting in which the feature-generation prompt is improved using statistical feedback from the generated features.

C. Statistical Validation for Time-Series Features

Financial FE requires careful validation because apparent relationships in time-series may be unstable, regime-dependent, or driven by spurious correlations. Granger causality provides a classical framework for testing whether lagged values of one time-series improve prediction of another beyond the target’s own lagged history [3]. In this paper, Granger causality is used as a predictive-relevance diagnostic rather than as evidence of structural causality.

Prior work on time-varying Granger causality emphasizes that temporal relationships can change across regimes, motivating rolling or windowed evaluation rather than a single static test [8]. Bootstrap-based causality tests further address robustness concerns that arise in finite-sample financial settings [9]. Our evaluator follows the same general motivation for time-series caution, but uses a fixed rolling-window Granger score as the feedback signal for comparing prompt variants.

D. Prompt Optimization and Autonomous Research Loops

Prompt design strongly affects the behavior of LLM systems, but prompts are often tuned manually through repeated trial and error. Recent work has begun to treat prompt design itself as an optimization problem. In Optimization by PROMpting (OPRO), an LLM proposes new candidate prompts based on previous prompts and their scores [10]. This establishes a general pattern in which natural-language instructions can be improved through iterative feedback. Open-source systems such as Karpathy’s *autoresearch* repository illustrate a related keep-or-revert experimental loop in which an AI agent proposes code changes, evaluates them with a quantitative metric, and retains only improvements [11].

Our work applies this general pattern to agentic financial feature engineering. The optimization objective is not benchmark task accuracy, but a Granger-based composite score computed from generated time-series features. The mutation surface of AI agent is also deliberately restricted: the feature-generation prompt may change, while the dataset suite, evaluator, and scoring logic remain fixed. This connects automated feature engineering, LLM agents, time-series validation, and prompt optimization into a single closed-loop framework.

III. CLOSED-LOOP AGENTIC FE FRAMEWORK

The proposed framework is designed as a closed-loop system in which feature generation, statistical evaluation, and prompt improvement are separated into distinct components. This separation is central to the design. The feature-generation prompt is allowed to change, but the dataset suite, feature execution logic, evaluator, and scoring rule remain fixed during optimization. As a result, accepted improvements can be attributed to changes in the prompt rather than to changes in the evaluation procedure. Figure 1 summarizes the overall architecture.

A. Agentic Feature Generation

The first component is an LLM-based feature-generation pipeline that transforms dataset context into executable feature functions. For each dataset, the pipeline begins by profiling the available columns, time coverage, missing-value patterns, data types, and basic statistical properties. This gives the feature-generation agent a structured view of the dataset before it proposes new predictors.

The system then incorporates external context from documentation or research material. This step is intended to make feature generation more domain-aware than a purely syntactic

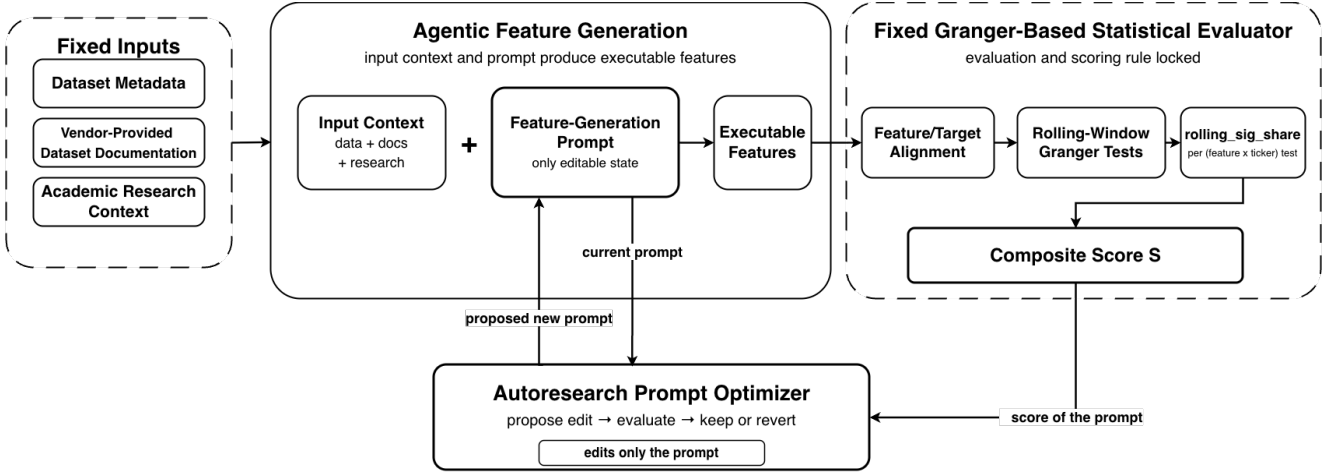


Fig. 1. Closed-loop agentic feature-engineering framework. The agentic pipeline generates executable candidate features, the fixed Granger evaluator converts them into rolling-window predictive-relevance scores, and the autoresearch loop updates only the feature-generation prompt when the composite score improves.

transformation search. Instead of only applying generic operations to columns, the agent can use contextual information to form feature hypotheses related to financial mechanisms such as abnormal activity, sentiment changes, news intensity, order-flow behavior, or earnings-expectation revisions.

Using the dataset profile and retrieved context, the agent proposes candidate feature specifications. Each specification describes the intended signal, the financial intuition and the computation needed to construct the feature from the raw data. In actual runs, generated examples included a market-microstructure feature that detects sudden increases in participant-volume concentration, a retail-demand feature that aggregates recent negative or positive foot-traffic surprises relative to a rolling baseline, and an earnings-attention feature that tracks changes in the number of crowd-sourced EPS forecasts before earnings events. These specifications are then converted into executable Python functions with a consistent interface. The generated functions are treated as the output of the agentic generation stage and as the input to the statistical evaluation stage.

This design keeps the feature-generation process both context-aware and auditable. The generated features are accompanied by descriptions and formulas that can be inspected by a human reviewer, while the executable functions allow the same evaluation procedure to be repeated across datasets, tickers, horizons, and prompt variants.

B. Fixed Granger-Based Statistical Feedback

The second component is a fixed statistical feedback layer based on Granger predictive relevance. For each generated feature, the evaluator aligns the feature series with forward-return targets and evaluates whether lagged feature values improve prediction beyond the target’s own lagged history. The detailed target construction, lag selection, rolling-window settings, and composite-score aggregation are defined in Section IV.

The key output of this stage is a rolling-window predictive-relevance signal. Rather than relying only on a single full-

sample test, the evaluator measures how often a generated feature shows Granger-predictive structure across rolling market windows. This is important in financial time-series, because relationships may be regime-dependent: a feature can be useful during some periods and weak during others.

The evaluator is deliberately fixed throughout prompt optimization. The fixed evaluator therefore acts as a stable feedback source for comparing different feature-generation prompts.

C. Autoresearch Prompt Optimization

The third component is an autonomous prompt-optimization loop, referred to as autoresearch. The loop treats the feature-generation prompt as the object of search. This is motivated by the fact that prompt wording can strongly affect which signal families the agent considers, how it uses retrieved context, and whether it produces generic transformations or statistically testable financial hypotheses.

Each iteration follows a propose–evaluate–keep-or-revert structure. First, the optimizer proposes one edit to the current feature-generation prompt. The edited prompt is then used to regenerate candidate feature functions. These features are evaluated by the fixed Granger-based scorer across the dataset suite, and the resulting outputs are aggregated into the composite score. If the candidate prompt produces a strictly higher composite score than the current best prompt, the edit is accepted. Otherwise, it is rejected and the system reverts to the previous best prompt.

The AI agent’s mutation surface is intentionally narrow. The autoresearch loop may edit only the feature-generation prompt. It may not modify the dataset loaders, generated-feature execution framework, target construction, evaluator, score definition, or dataset suite. This restriction makes the optimization result easier to interpret: a score improvement reflects a prompt that generates better feature candidates under the same statistical test design.

TABLE I
EVALUATION DATASET SUITE

Dataset	Signal Domain	Tickers
Swiss market data	Market microstructure / order flow	10
PassBy	Retail foot traffic	10
ContextAnalytics_X	Social sentiment	10
MTNewsires	News flow	10
Estimize	Earnings forecasts	10

The loop is implemented with persistent file-based state and stateless per-iteration execution. The current best prompt, best score, and iteration history are stored outside the LLM session and reloaded at the start of each iteration. This makes the optimization process restartable and auditable, while avoiding dependence on expensive conversational memory for critical experiment state.

IV. EXPERIMENTAL DESIGN

This section defines the dataset suite, the Granger-based feature-evaluation procedure, the composite score, and the optimization protocol used to test the proposed framework. The experiment is designed to measure whether autonomous prompt optimization improves the statistical predictive-relevance signals of generated features under a fixed evaluation procedure.

A. Dataset Suite

The evaluation suite contains 5 alternative-data datasets, each evaluated on 10 tickers. The datasets were selected to cover different types of financial information channels, including market microstructure, retail foot traffic, social sentiment, news flow, and earnings expectations. Table I summarizes the dataset suite.

Using multiple datasets is important because a prompt edit may improve one signal domain while weakening another. The evaluation therefore uses a multi-dataset objective rather than optimizing against a single data source.

B. Feature Evaluation Procedure

For each candidate prompt, the feature-generation pipeline produces executable feature functions for each dataset. These functions are evaluated ticker by ticker. For each ticker, the raw dataset is joined with a price series, and two forward-return targets are constructed: 1 trading day and 21 trading days. For horizon k , the target is the forward log return

$$y_{t,k} = \log(P_{t+k}) - \log(P_t), \quad (1)$$

where P_t is the adjusted closing price when available and the closing price otherwise. The 1-day and 21-day horizons are evaluated as separate tests.

Each generated feature is aligned with the corresponding target series on the time index. Missing values are removed, stationarity checks are applied, and differencing is used when necessary before Granger testing. Candidate lags are selected

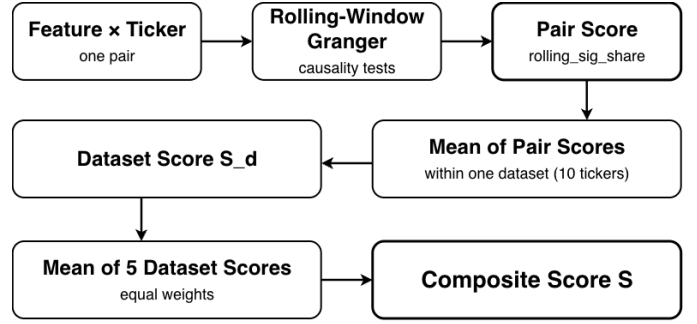


Fig. 2. Composite-score construction. Each evaluable feature–ticker–horizon test produces a `rolling_sig_share`; values are averaged within each dataset and then averaged equally across datasets.

from $\{1, 2, 3, 5\}$, and the lag used for each feature–ticker–horizon test is chosen by minimizing the Akaike information criterion (AIC) [12].

The main feature-level statistic is `rolling_sig_share`. For each feature–ticker–horizon test, the evaluator uses 252-trading-day rolling windows with a stride of 5 trading days. Within each window, the feature and target are standardized, a VAR stability check is applied, and unstable windows are excluded from the denominator. For each stable window, an analytic Granger F-test is run at the selected lag. The `rolling_sig_share` value is the fraction of stable windows in which the test rejects the no-predictive-relevance null hypothesis at $\alpha = 0.05$.

Thus, `rolling_sig_share` is not a p-value. It is a regime-frequency measure that captures how often a feature exhibits Granger-predictive structure across rolling market windows.

C. Composite Score

The autoresearch loop optimizes a single scalar composite score. Let D denote the set of datasets and let E_d denote the set of evaluable feature–ticker–horizon tests for dataset d . For each test $e \in E_d$, the evaluator computes `rolling_sig_share`(e, d). The composite score is defined as

$$S = \frac{1}{|D|} \sum_{d \in D} \frac{1}{|E_d|} \sum_{e \in E_d} \text{rolling_sig_share}(e, d). \quad (2)$$

The score is computed in two stages. First, `rolling_sig_share` values are averaged within each dataset. Second, the per-dataset means are averaged with equal weight across datasets. This prevents a dataset with more evaluable tests from dominating the optimization target.

The composite score is used only as an internal optimization signal. Full-sample Granger p-values and bootstrap p-values are computed as part of the broader evaluation output, but they are not aggregated into the optimization objective.

D. Optimization Protocol

Prompt optimization follows a hill-climbing algorithm. The baseline prompt is evaluated first across the full five-dataset

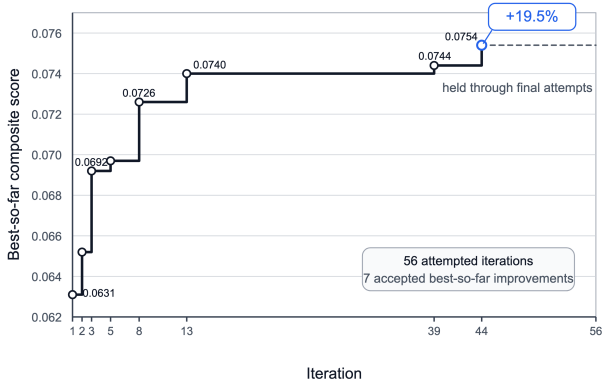


Fig. 3. Best-so-far autoresearch learning curve. The plot shows only the baseline and the 7 best-so-far improvements, not all 56 attempted iterations.

suite. Then, each iteration proposes one edit to the feature-generation prompt, regenerates candidate feature functions, evaluates them under the fixed procedure, and computes the composite score in (2). If the candidate prompt produces a composite score higher than the current best prompt, the edit is accepted. Otherwise, the edit is rejected and the system reverts to the previous best prompt. During this process, only the feature-generation prompt changes.

The experiment completed 56 attempted prompt-optimization iterations. Because non-improving edits are reverted, the reported trajectory includes only the baseline and the 7 prompt edits that produced new best-so-far scores.

V. RESULTS

This section reports how the autoresearch loop behaved under the fixed evaluation protocol defined in Section IV. The main question is whether prompt edits can improve the internal Granger composite score while the dataset suite, feature execution logic, evaluator, and scoring rule remain unchanged.

A. Optimization Trajectory

The baseline feature-generation prompt achieved a composite score of 0.0631. After 56 attempted prompt-optimization iterations, the best prompt found by the loop achieved a score of 0.0754. This is an absolute increase of 0.0123 and a relative improvement of 19.5%. Since the composite score is based on rolling-window predictive-relevance frequency, this means that the generated features more often exhibited Granger-predictive structure under the fixed evaluator.

Table II summarizes the best-so-far optimization trajectory. The table reports only the baseline and the 7 prompt edits that produced a new best score; non-improving edits are omitted because they were reverted. The score improved quickly in the early iterations, reaching 0.0740 by iteration 13. It then plateaued until iteration 39, indicating that many plausible

TABLE II
AUTORESEARCH OPTIMIZATION TRAJECTORY

Iter.	Prompt-Edit Hypothesis	Score
1	Baseline	0.0631
2	Reduce perspectives from 7 to 3	0.0652
3	Remove Corp-Action / Market-Data guides	0.0692
5	Trim category priorities and add order-flow emphasis	0.0697
8	Remove economic-rationale-tag rules	0.0726
13	Add cross-regime perspective	0.0740
39	Emphasize asymmetric downside shock variants	0.0744
44	Add few-shot example for negative-shock count	0.0754

TABLE III
PER-DATASET BASELINE AND FINAL-BEST SCORES

Dataset	Baseline	Final	Change
Swiss market data	0.0671	0.0640	-4.6%
PassBy	0.0587	0.0912	+55%
ContextAnalytics_X	0.0623	0.0644	+3%
MTNewswires	0.0579	0.0751	+30%
Estimize	0.0694	0.0823	+19%

prompt edits did not improve the fixed objective. The final best score was reached at iteration 44.

The step-like trajectory in Fig. 3 reflects the hill-climbing structure of the optimization loop. A prompt edit affects the visible curve only when it improves the best-so-far score. This is important because the system is not manually selecting individual features after evaluation; it is selecting prompt states that generate stronger feature candidates on average.

B. Cross-Dataset and Prompt-Edit Behavior

The improvement was not uniform across datasets. Table III compares the baseline prompt with the final best prompt. Four of the five datasets improved, while one market-data dataset decreased modestly. The strongest gains appeared in PassBy and MTNewswires, while ContextAnalytics_X improved only modestly.

This mixed behavior supports the use of an equal-weight multi-dataset objective. A prompt optimized on only one dataset could improve one financial signal domain while degrading others. In contrast, the selected prompt improved the average score across heterogeneous alternative-data sources, even though it did not improve every source individually.

The accepted prompt edits also suggest that useful optimization did not come from simply adding more instructions. Early improvements simplified the prompt by reducing the number of requested perspectives and removing less useful guidance. Later improvements added more targeted financial patterns. Overall, the best-performing edits made the prompt more focused: they removed distracting guidance while adding specific, testable signal patterns.

VI. DISCUSSION AND LIMITATIONS

The results suggest that statistical feedback can guide prompt improvement in an agentic feature-engineering pipeline. Because the dataset suite, feature execution logic, evaluator, and scoring rule were held fixed, the observed improvement is best interpreted as evidence that the accepted prompt edits produced stronger feature candidates under the selected evaluation protocol. However, the result remains internal to this protocol. The composite score measures rolling-window Granger predictive relevance; it does not measure portfolio returns, transaction-cost-adjusted performance, risk-adjusted profitability, or live trading robustness.

A second limitation comes from the Granger-based evaluator. Granger causality is useful for testing whether lagged feature values improve prediction beyond the target's own lagged history, but it should be interpreted as predictive relevance rather than structural causality [3]. The test is also based on a selected lag structure and is primarily linear, so it may miss nonlinear relationships that could still be useful for forecasting. Rolling-window evaluation helps capture regime-dependent behavior, but it does not remove the risk of unstable relationships, sample-specific effects, or false discoveries across many feature-ticker-horizon tests.

The prompt-selection objective was intentionally kept simple and repeatable. Although the broader evaluation output includes diagnostic full-sample and bootstrap p-values, the optimization loop selects prompts only using the rolling-window composite score. This makes the feedback signal practical for repeated prompt search and keeps the comparison between prompt variants consistent. Future work can extend the objective with stronger robustness criteria, such as multiple-testing correction, bootstrap-based aggregation, or nonlinear information-flow measures such as transfer entropy.

The optimization procedure itself has search and cost limitations. The autoresearch loop uses hill climbing, so it can find local improvements but does not guarantee a globally optimal prompt. The plateau observed between iterations 13 and 39 illustrates this limitation: many plausible edits did not improve the fixed objective. More exploratory strategies, such as maintaining multiple prompt candidates or using population-based search, could explore the prompt space more broadly, but they would also increase computational cost. Each iteration already requires feature generation and evaluation across five datasets and 10 tickers per dataset.

Finally, the current study focuses on financial time-series data, where heterogeneous alternative-data sources provide a demanding testbed for closed-loop feature engineering. The same framework could be adapted to other time-series domains by changing the feature-generation prompt, retrieval sources, target definitions, and statistical evaluator. This makes cross-domain generalization a natural direction for future work. Future extensions should also test whether the generated features improve downstream forecasting models, incorporate nonlinear validation methods, and add stronger robustness controls.

VII. CONCLUSION

This paper presented a closed-loop framework for agentic financial feature engineering. The framework combines LLM-based feature generation, fixed Granger-based statistical evaluation, and autonomous prompt optimization. By allowing only the feature-generation prompt to change while holding the other components fixed, the system provides a controlled way to test whether prompt edits improve generated feature quality.

Across five alternative-data datasets with 10 tickers each, the autoresearch loop completed 56 attempted iterations and produced seven best-so-far prompt improvements. The internal composite score increased from 0.0631 to 0.0754, corresponding to a 19.5% relative improvement. These results suggest that statistical feedback can guide prompt improvement in an agentic feature-engineering pipeline.

The reported improvement should be interpreted as stronger rolling-window predictive relevance under the fixed evaluator, not as evidence of trading profitability or structural causality. Future work will test whether the generated features improve downstream forecasting models, extend the evaluation with stronger robustness and nonlinear validation methods, and apply the framework to broader time-series domains.

REFERENCES

- [1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations*, 2023, arXiv:2210.03629.
- [2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.
- [3] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.
- [4] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–10.
- [5] F. Horn, R. Pack, and M. Rieger, "The autfeat python library for automated feature engineering and selection," *arXiv preprint arXiv:1901.07329*, 2020.
- [6] T. Zhang, Z. Fan, H. Luo, F. Liu, Q. Liu, W. Cao, and J. Li, "OpenFE: Automated feature generation with expert-level performance," in *International Conference on Machine Learning*, 2022, arXiv:2211.12507.
- [7] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.
- [8] C. F. Baum, S. Hurn, and J. Otero, "Testing for time-varying granger causality," *The Stata Journal*, vol. 22, no. 2, pp. 355–378, 2022.
- [9] R. S. Hacker and A. Hatemi-J, "A bootstrap test for causality with endogenous lag length choice: Theory and application in finance," *Journal of Economic Studies*, vol. 39, no. 2, pp. 144–160, 2012.
- [10] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large language models as optimizers," in *International Conference on Learning Representations*, 2024. [Online]. Available: <https://openreview.net/forum?id=Bb4VGOWELI>
- [11] A. Karpathy, "autoresearch: AI agents running research on single-GPU nanochat training automatically," GitHub repository, 2026, accessed: 2026-05-23. [Online]. Available: <https://github.com/karpathy/autoresearch>
- [12] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.