GPT-based Multi-Agent System for Investment Factor Mining

Course Id：IBA6316

Team Members：

223040128 Han, CHEN

223020190 Zhiyu, LIANG

224020253 Ruopu, LI

224020343 Yuyang, SHEN

224020265 Jiaxi, ZHAO

Sponsor: Guolian Fund Management Co., Ltd.

Executive Summary

This project introduces a GPT - based multi - agent system that automates the investment factor mining process in quantitative finance, aiming to overcome the inefficiencies of traditional manual methods. The system is composed of three core agents: FactorAgent, which generates candidate factors from natural language inputs; CalculatorAgent, which validates and formalizes mathematical expressions; and CodeAgent, which converts validated formulas into executable Python code, conducts backtesting, and evaluates performance using metrics like the Information Coefficient (IC) and Sharpe Ratio. These agents operate in a closed - loop workflow with real - time feedback, enabling iterative improvement and high - quality output.

Key technical innovations include the use of Chain - of - Thought (CoT) prompting and ReAct reasoning to enhance agent intelligence, as well as Retrieval - Augmented Generation (RAG) to improve factor diversity and accuracy. The system was initially deployed with local code. During the project, an attempt was made to migrate it to the Dify platform, but the migration failed. The project team then decided to continue development using the APPL language, leveraging its capabilities for complex financial logic implementation.

Compared with traditional approaches, this system significantly reduces factor development time from weeks to hours, minimizes human bias and errors, and lowers technical barriers for non - experts. Empirical testing shows that selected factors have IC values above 0.07, demonstrating strong predictive power. This project provides a scalable and efficient framework for automated financial modeling, facilitating seamless integration with downstream trading platforms and laying the foundation for future intelligent factor discovery research.

Table of Contents

I Project Introduction

In the realm of quantitative finance, traditional manual factor mining is slow, error-prone, and heavily reliant on expert labor, often missing market opportunities due to inefficiencies. Our project introduces a **GPT-based Multi-Agent System** to automate this process, aiming to streamline factor discovery, validation, and implementation for financial institutions, analysts, and researchers.

This solution bridges the gap between AI innovation and practical finance, offering a scalable, bias-reduced approach to factor mining. By combining GPT's reasoning with specialized agents, it empowers stakeholders to uncover reliable factors faster, adapting to dynamic markets and enhancing investment decision-making.

1.1 Project Background

In the field of quantitative investment, the construction of robust investment factors serves as a core component of financial decision-making, yet traditional manual development models are grappling with challenges in efficiency, accuracy, and applicability. As the complexity of financial data surges—driven by exponential growth in datasets (e.g., high-frequency trading data, alternative data)—the divergent needs of three key user groups have become increasingly pronounced:

Financial Institutions require rapid factor development cycles to capitalize on fleeting market opportunities. Traditional manual processes, which take weeks to complete data cleaning, factor back testing, and validation, often render factors obsolete amid fast-paced market rotations. Additionally, the high investments in computing power and infrastructure for large-scale cross-asset analysis, coupled with unavoidable biases like survivorship and look-ahead bias in manual operations, consistently erode the practical value of factors.

Financial Analysts face dual pressures of information overload and technical barriers. The exponential growth of market data demands factor analysis that balances speed (e.g., real-time detection of abnormal trading signals) and precision (e.g., cross-cycle stability validation). Meanwhile, the heavy reliance on programming skills in traditional models forces analysts to allocate significant effort to data processing and logic verification, diverting resources from strategic innovation.

Academic Researchers are constrained by fragmented tools and data silos. Existing methodologies lack automated factor mining platforms, leading to inefficiencies across the entire research pipeline—from data acquisition (e.g., high-quality historical datasets, real-time market APIs) to cross-theoretical validation (e.g., multi-factor synergy analysis). Technical complexity and resource costs significantly hinder academic innovation in this domain.

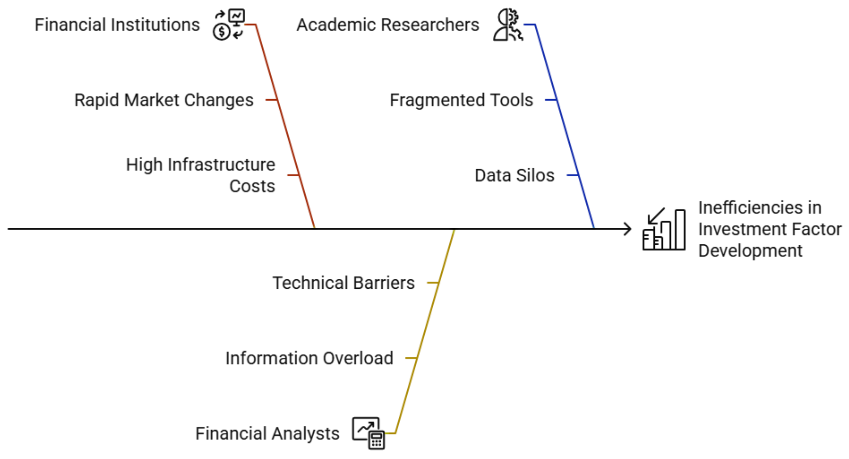


Fig 1.1 Investment factor mining user predicament segmentation diagram

This mismatch between the inefficiency of traditional models and the evolving needs of users has created a critical gap: financial institutions need "hour-level" factor iteration capabilities to adapt to market volatility, analysts require "code-free" tools to streamline workflows, and researchers need "one-stop" platforms to overcome data and technical bottlenecks. Leveraging artificial intelligence to build an automated factor mining system—through systematic integration of data processing, intelligent generation, and rigorous validation—emerges as the pivotal solution to address these pain points, enhancing both the quality and efficiency of factor development for diverse stakeholders.

1.2 System Architecture

The multi - agent system developed in this project achieves end - to - end automation in factor development—from hypothesis generation to the output of validated factors—through the pipeline collaboration of **FactorAgent (Factor Construction Agent), CalculatorAgent (Formula Calculation Agent), and CodeAgent (Code Generation and Validation Agent)**. Operating in a progressive workflow of "concept design - logic validation - engineering implementation - factor validation," these three agents form a closed - loop iterative framework that significantly enhances the efficiency and reliability of factor development.

1.2.1 Core Roles and Functionalities of Agents

- **FactorAgent (FactorGPT)**, the "factor brain" of the system, translates users' natural language requirements (e.g., "identify price - volume relationship factors") into structured factor hypotheses using predefined factor construction rules (e.g., economic theory - based design paradigms, overfitting avoidance constraints) and metadata (available data fields, operator lists). Its key capabilities include generating factor names, preliminary formulas, involved fields, economic interpretations, and multiple calculation methods for the same factor concept (e.g., volume normalization with different time windows), providing diversity for subsequent validation. The output establishes the theoretical framework and multi - dimensional calculation paths for factors, serving as the foundation for logical validation.

- **CalculatorAgent (CalculatorGPT)** acts as the "validation engine" for factor logic, receiving preliminary formulas and operator lists from FactorAgent to ensure mathematical rigor and implement ability through formula synthesis, missing operator handling, and feasibility checks. Specifically, it consolidates multi - version formulas into a single executable expression, automatically generating mathematical definitions and code skeletons for undefined operators (e.g., a custom "liquidity adjustment factor"). It also verifies data - dimensional compatibility (e.g., time - series alignment, cross -

sectional data matching) to prevent calculation errors. The output includes complete mathematical expressions, operator lists, and validation results, providing a precise logical blueprint for code generation.

- **CodeAgent (CodeGPT)** serves as both the "engineering bridge" and the "factor validator" for factors. First, it converts validated mathematical expressions from CalculatorAgent into efficient executable Python code. Its core functions include strict data type definition, computational optimization (e.g., replacing loops with vectorized operations), error handling (indentation errors, data alignment issues), and standardized output (parameter configurations, unit test cases). By integrating Chain of Thought (CoT) step - by - step reasoning and ReAct dynamic tool invocation, CodeAgent autonomously debugs common issues and generates production - ready code templates, enabling seamless translation of factor logic into engineering implementations.

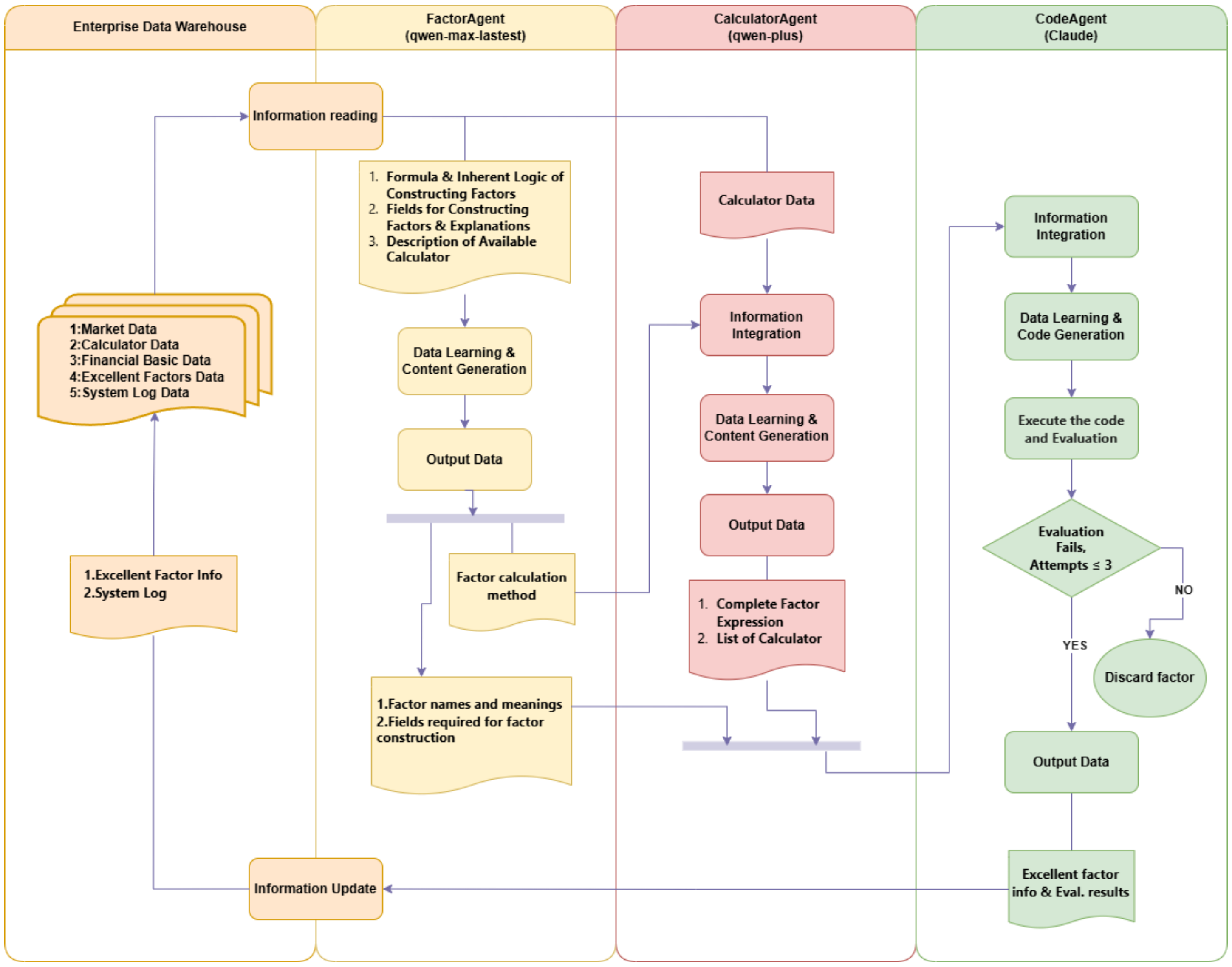
After code generation, CodeAgent executes the code on historical datasets. It calculates the Information Coefficient (IC) of the mined factor, which measures the correlation between the factor values and subsequent asset returns. Then, it compares the average absolute value of the IC with a predefined threshold. Only factors whose IC average absolute value meets the threshold are passed on for further use, such as inclusion in the factor library.

1.2.2 Agent Collaboration Workflow and Technical Advantages

The three agents collaborate in an "input - process - output - validate" pipeline, forming a closed - loop iteration: FactorAgent parses user needs and generates factor hypotheses using data and operators; CalculatorAgent validates preliminary formulas, supplements missing logic, and refines them into executable expressions; CodeAgent converts mathematical formulas into optimized code, executes the code, calculates the IC of the factor, and filters factors based on the IC threshold. If logical or implementation issues are detected during this process, CodeAgent will discard the factor, thereby creating a "hypothesis - verification - factor verification" loop.

For example, when a user requests a "*factor linking large - order trading and fundamentals*," FactorAgent outputs a preliminary formula like "*large - order net buy ratio* × *net profit growth*," CalculatorAgent refines "*large - order net buy ratio*" into a concrete calculation using specific data fields, and CodeAgent transforms the final formula into vectorized code. Then, CodeAgent runs the code on historical data, calculates the IC of the factor, and decides whether to keep or discard the factor based on the IC threshold.

The architecture's core strengths lie in **specialized division of labor and automated closed - loop processing**: each agent focuses on a unique core task, eliminating cross - stage knowledge barriers; full automation reduces factor development cycles from "weeks" to "hours" and minimizes human intervention and errors through layered logic, code validation, and factor validation. Technical integrations (e.g., CoT step - by - step reasoning, ReAct tool invocation) enhance agent decision accuracy, providing a standardized, scalable framework for financial institutions to generate high - quality factors rapidly.



Enterprise Data Warehouse
FactorAgent (qwen-max-lastest)
CalculatorAgent (qwen-plus)
CodeAgent (Claude)
Information reading
1. Formula & Inherent Logic of Constructing Factors
2. Fields for Constructing Factors & Explanations
3. Description of Available Calculator
Calculator Data
Information Integration
1:Market Data
2:Calculator Data
3:Financial Basic Data
4:Excellent Factors Data
5:System Log Data
Data Learning & Content Generation
Information Integration
Data Learning & Code Generation
Data Learning & Content Generation
Execute the code and Evaluation
Output Data
Output Data
Evaluation Fails, Attempts ≤ 3
NO
1.Excellent Factor Info
2.System Log
Factor calculation method
1. Complete Factor Expression
2. List of Calculator
YES
Discard factor
1.Factor names and meanings
2.Fields required for factor construction
Output Data
Information Update
Excellent factor info & Eval. results

Fig 1.2 Automated Factor Mining System - Three-Agent Collaboration Workflow

1.3 Key Deliverables

1.3.1 Automated Factor Development System

An integrated system that combines FactorAgent, CalculatorAgent, and the enhanced CodeAgent. It automates the entire process of factor development, from generating factor hypotheses based on user needs to producing executable code and validating factors. The enhanced CodeAgent calculates the Information Coefficient (IC) of each factor, filters out those that don't meet the predefined threshold, ensuring only high - quality factors are retained.

1.3.2 Factor Repository and Documentation

A repository storing a collection of validated investment factors. Each factor comes with detailed documentation, including formulas, data fields, economic interpretations, and performance metrics such as historical IC and Sharpe Ratio. Optimized Python code with configurable parameters is also provided for easy integration into different investment models.

II System Design and Technical Details

2.1 Project progress

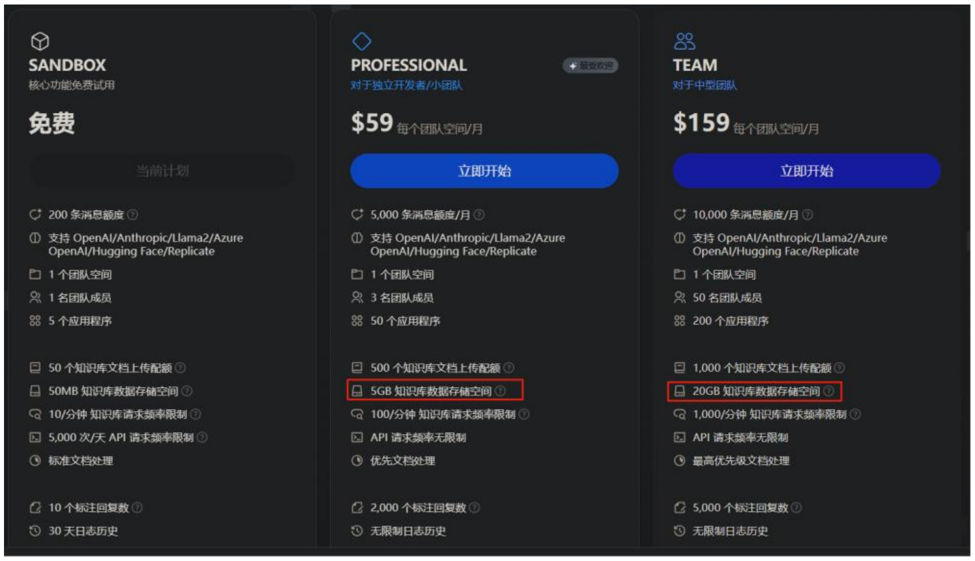
2.1.1 Review of Midterm Progress

In the previous project progress at midterm, the group members successfully built the whole agent workflow structure by APPL with complete functionality. We insert complex code into the workflow, integrated operator databases and factor databases, and implemented corresponding naming convention, features and invocation logic. To enhance the performance of agents within APPL, we employed techniques such as Chain-of-Thought (CoT) prompting, ReAct and advanced prompt engineering. These measures significantly improved the efficiency and accuracy of factor generation, code execution, and result evaluation. Additionally, a complex system architecture was established, with the collaboration between FactorAgent, CalculatorAgent, and CodeAgent, which laid the foundation for further optimization and refinement of the project.

2.1.2 Migration Challenges Encountered

Later, the group tried to shift the project from APPL to Dify. During the attempt to migrate the system to the Dify platform, several key issues were identified, which ultimately led the team to abandon the migration and continue development locally.

- **Limited Knowledge Base Capacity**: Dify's knowledge base could not support the large-scale financial data and extensive factor libraries required by our system, resulting in slow data retrieval and frequent crashes under heavy load. The whole data needs in our project is much larger than 20GB, which is the limit of Dify.



SANDBOX	PROFESSIONAL	TEAM
核心功能免费试用	对于独立开发者/小团队	对于中型团队
免费	$59 每个团队空间/月	$159 每个团队空间/月
当前计划	立即开始	立即开始
200 条消息额度	5,000 条消息额度/月	10,000 条消息额度/月
支持 OpenAI/Anthropic/Llama2/Azure OpenAI/Hugging Face/Replicate	支持 OpenAI/Anthropic/Llama2/Azure OpenAI/Hugging Face/Replicate	支持 OpenAI/Anthropic/Llama2/Azure OpenAI/Hugging Face/Replicate
1 个团队空间	1 个团队空间	1 个团队空间
1 名团队成员	3 名团队成员	50 名团队成员
5 个应用程序	50 个应用程序	200 个应用程序
50 个知识库文档上传配额	500 个知识库文档上传配额	1,000 个知识库文档上传配额
50MB 知识库数据存储空间	5GB 知识库数据存储空间	20GB 知识库数据存储空间
10/分钟 知识库请求频率限制	100/分钟 知识库请求频率限制	1,000/分钟 知识库请求频率限制
5,000 次/天 API 请求频率限制	API 请求频率无限制	API 请求频率无限制
标准文档处理	优先文档处理	最高优先级文档处理
10 个标注回复数	2,000 个标注回复数	5,000 个标注回复数
30 天日志历史	无限制日志历史	无限制日志历史

Fig 2.1 Limitation of Dify

- **Lack of Advanced Agent Support**: The platform lacked native capabilities for complex agent interactions and real-time adjustments, making it difficult to implement key techniques such as CoT prompting and ReAct reasoning. Besides, there is no FactorGPT, CodeGPT and relevant resources embedded in Dify.

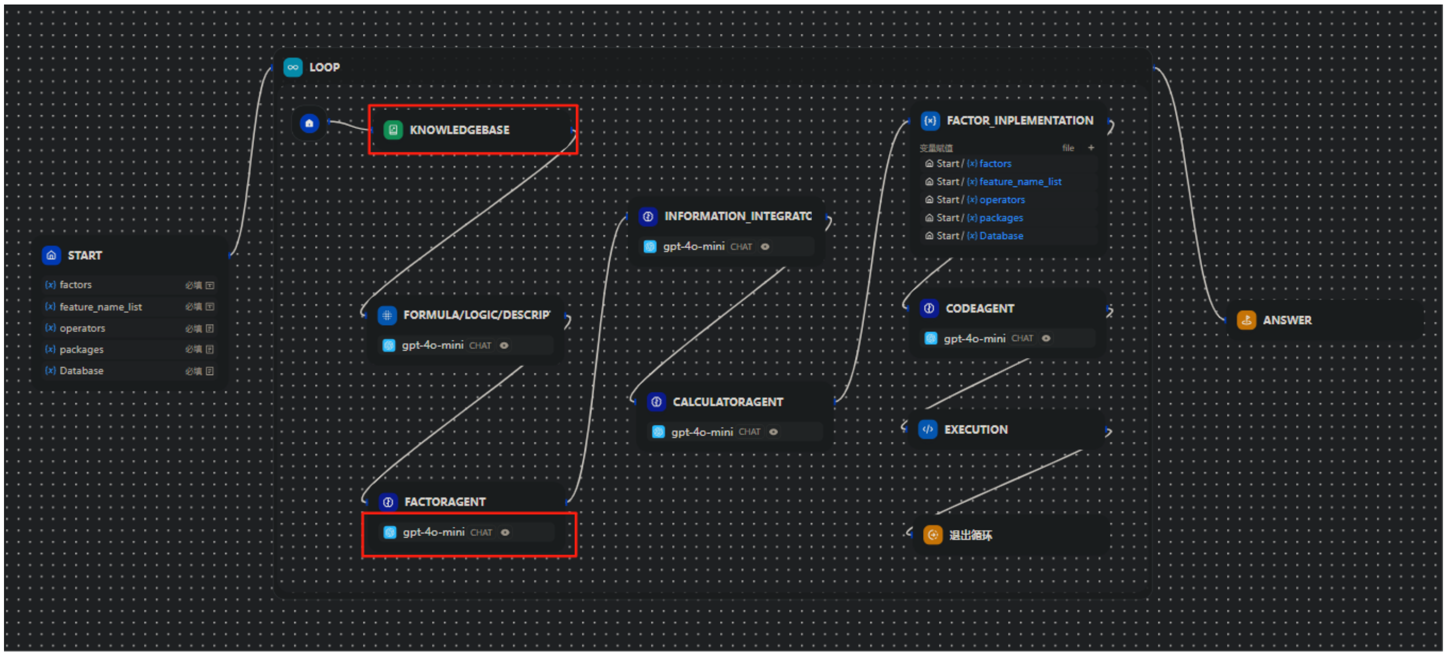


Fig 2.2 Workflow Design Trial at Dify

- **Insufficient Customization and Scalability:**From an enterprise perspective, Dify offered limited flexibility for integrating custom logic and advanced features like automated code generation and high-precision evaluation.

As a result of these limitations, the project timeline was significantly affected. To maintain progress and system performance, the team decided to return to local deploy, where greater control and customization were available. This allowed for continued refinement of the agent architecture and technical implementation without external constraints.

2.2 Technical details and Innovation

2.2.1 Core Agent Implementation Details

The system architecture is composed of three main agents: FactorAgent, CalculatorAgent, and CodeAgent, each designed to perform a specific role in the financial factor mining workflow. From the midterm to the final stage, their technical implementation has been refined with improved algorithms, prompt engineering techniques, and integration logic.

FactorAgent

FactorAgent is responsible for generating candidate financial factors based on domain knowledge and market insights. Initially, it relied on basic prompting strategies to generate factors. However, during the later stages, we enhanced its performance by incorporating:

- **Chain-of-Thought (CoT) Prompting :**This technique significantly improved FactorAgent's logical reasoning capabilities, enabling it to break down complex financial relationships into structured factor definitions.
- **RAG (Retrieval-Augmented Generation) :**FactorAgent now integrates with the factor library using RAG, allowing it to retrieve similar existing factors and refine them with new variations or constraints.
- **Algorithm Optimization :**We introduced clustering-based similarity checks to avoid redundant factor generation and applied reinforcement logic to prioritize high-potential factor templates.

These improvements led to a significant increase in the quality and diversity of generated factors compared to the midterm version.

Chart 2.1 Technicals at CodeAgent

Chain-of-Thought (CoT)	Encourages step-by-step reasoning	Improved logical consistency in factor generation
ReAct	Combines reasoning and action steps	Enhanced planning ability and error reduction in code generation
RAG	Retrieves relevant context before generating output	Increased accuracy and relevance of outputs

CalculatorAgent

CalculatorAgent functions as an intermediary between FactorAgent and CodeAgent, translating abstract factor logic into mathematical expressions. Key technical improvements include:

- **Tree-based Formula Hashing:** Each formula is uniquely identified using Merkle tree fingerprints to ensure traceability and prevent duplication across runs.
- **Symbolic Computation:** Enhanced with SymPy integrations, CalculatorAgent can validate the correctness of formulas and simplify expressions before code translation.
- **Handling of Unsupported Operators:** At the midterm stage, CalculatorAgent struggled with non-standard operators. Now, it includes fallback mechanisms that either approximate missing operations or suggest alternative implementations.

This refinement increased both the accuracy and execution readiness of the formulas processed.

CodeAgent

CodeAgent translates the validated formulas into executable Python code for empirical testing. In the early phase, code generation was prone to syntax errors and lacked adaptability. Key advancements made:

- **ReAct-Based Prompting Strategy:** CodeAgent now uses ReAct (Reason + Act) prompts to first reason about the structure of the code before writing it.
- **Enhanced Template Library:** A larger set of code templates was added to support diverse financial computation patterns, including time-series handling, normalization, and cross-sectional ranking.
- **Error Prevention Logic:** Static analysis tools were integrated to catch common issues such as indentation errors, undefined variables, and keyword conflicts.

As a result, code generation success rate improved in the final version.

2.2.2 Innovation

The project introduces several innovative aspects that significantly differentiate it from traditional manual factor mining methods and existing automated tools. These innovations span across system design, user interaction, technical implementation, and overall workflow efficiency.

1. Fully Automated Financial Factor Mining

Unlike conventional approaches that heavily rely on domain experts for factor formulation and code implementation, this system automates the entire process — from factor generation to evaluation. The integration of FactorAgent, CalculatorAgent, and CodeAgent enables end-to-end automation, reducing human intervention and minimizing research bias such as confirmation bias or publication bias.

2. Real-time Feedback and Iterative Optimization

A key innovation lies in the dynamic feedback loop between agents. CodeAgent, continuously evaluates generated factors and sends detailed performance metrics back to FactorAgent and CodeAgent for iterative refinement.

- This closed-loop mechanism enhances learning over time and ensures that only high-performing factors are retained.
- Compared to the initial prototype, the refined system supports real-time adjustments based on empirical results, significantly improving factor stability and predictive power.

3. Scalability and Extensibility

The system is highly scalable and extensible. It supports:

- Easy integration with external data sources
- Plug-and-play modules for new agent types or evaluation metrics
- Continuous model updates and prompt optimization without retraining

This modular design makes the system adaptable to various financial domains and allows integration with downstream quant platforms — a major improvement from the earlier rigid architecture tested on Dify.

4. Enhanced Efficiency and Reduced Error Rates

Through advanced prompting strategies and model fine-tuning, the system achieves higher accuracy and efficiency than traditional methods. The use of RAG-based retrieval , Merkle tree hashing for formula uniqueness , and optimized agent communication drastically reduces redundant computations and coding errors.

III Case Presentation and Result Evaluation

3.1 Case Presentation

3.1.1 Factor Overview

The Divergence and Market Pressure Factor (DMPF) captures stock performance and potential investment value under varying capital inflows by analyzing differences in large/mid/small order trading behaviors, market uncertainty, and tail risks.

3.1.2. Detailed Calculation Method

```
Factor = Div(
  Sub(
    Add(
      TS_Mean(bo_big_b_sum_pct, d=20),  # 20-day rolling mean of large buy order proportion
      TS_Mean(bo_mid1_b_sum_pct, d=20)  # 20-day rolling mean of mid-tier1 buy order proportion
    ),
    TS_Mean(bo_small_s_sum_pct, d=20)  # 20-day rolling mean of small sell order proportion
  ),
  Add(
    Sqrt(
      EntropyVolatility(data, window=30, col_name='amt')  # 30-day rolling entropy volatility of trading volume
    ),
    Mul(
      ConditionalVaR(data, window=30, alpha=0.05, col_name='ret_std'),  # 30-day rolling tail risk (CVaR) of returns
      PermutationEntropy(data, window=30, order=3, delay=1, col_name='adj_close')  # 30-day rolling permutation entropy of price series
    )
  )
)
```

- Numerator: Measures the net effect of large/mid-tier buying versus small selling. A significantly positive value indicates stronger buying momentum from institutional investors and mid-sized capital, while negative values suggest higher retail selling pressure.
- Denominator: Combines entropy volatility (chaos in trading volume distribution), tail risk (CVaR of returns), and permutation entropy (randomness in price series) to assess market uncertainty and extreme event impacts.

3.1.3. Fields Used

['bo_big_b_sum_pct', 'bo_mid1_b_sum_pct', 'bo_small_s_sum_pct', 'amt', 'ret_std', 'adj_close']

3.1.4. Factor Interpretation and Application

A higher DMPF value implies strong stock performance driven by large/mid-sized capital inflows, lower market risks/uncertainty, and higher investment value. A lower value signals heightened market divergence or risks, warranting caution.

3.1.5 Improvements Relative to the Mid-Term Report

To avoid potential issues with Series operations when CodeAgent generates code and to facilitate CodeAgent's invocation of Calculators, we have encapsulated all Calculators to perform column-wise vectorized operations and return the original DataFrame with strictly named columns. Below is an example of a Calculator implementing the `EntropyVolatility` function:

```python
EntropyVolatility:
    description: |-
      Calculate the rolling window entropy volatility of daily time series for multi-stock mixed data to measure market uncertainty.
      Parameters:
        data (pd.DataFrame): Multi-stock mixed data containing the following columns:
          - securityid: Stock identifier column
          - tradedate: Time column (automatically converted to datetime)
          - {col_name}: Specified analysis column
        window (int): Rolling window size, default is 30.
        bins (int): Number of histogram bins, default is 10.
        col_name (str): Column name for analysis.
      Returns:
        pd.DataFrame: Original DataFrame with entropy volatility columns prefixed by `{col_name}_`.
          return grouped.apply(calculate_entropy_volatility)
```

Key Improvements:

1. Column-Wise Vectorization: Avoids pitfalls of Series operations by ensuring operations are applied across entire columns.

2. Strict Naming Conventions: Guarantees consistent output column names (e.g., `{col_name}_entropy_volatility`) for seamless integration with downstream processes.

3. Multi-Stock Compatibility: Supports mixed multi-stock data via `securityid` grouping and rolling calculations.

4. Explicit Parameterization: Clear definitions for `window`, `bins`, and `col_name` ensure reproducibility and flexibility.

3.2 Result Evaluation

3.2.1 Information Coefficient (IC)

Definition: Measures the correlation between factor values and future returns.

Formulas:

- Pearson IC:

$$IC_{\text{Pearson}} = \frac{\text{Cov}(f_t, r_{t+1})}{\sigma_f \cdot \sigma_r}$$

where $(f_t) = factorvalue, (r_{t+1}) = futurereturn, (\sigma_f) and (\sigma_r) = standarddeviations.$

- Spearman IC:

Convert factor values and returns into ranked scores, then compute Pearson correlation.

- Evaluation Criteria:

- Mean IC > 0.05 (effective), > 0.1 (excellent);

- IC stability: Standard deviation < 0.1 (monthly frequency), positive ratio > 60%.

3.2.2 Information Ratio (IR)

- Definition: Risk-adjusted excess return per unit of risk (\(IR = \text{Mean}(IC) / \text{Std}(IC)\)).

- Formula:

$$IR = \frac{\overline{IC}}{\sigma_{IC}}$$

- Evaluation Criteria:

  - IR > 1.0 (excellent), IR > 1.5 (top-tier, but check overfitting).

3.2.3 Quantile Analysis

- Definition: Validates monotonicity and long-short spread via grouped backtesting.

- Steps:

  1. Rank stocks into N groups (e.g., 10 deciles: Q1=worst, Q10=best).

  2. Compute long-short spread $(Q10 - Q1)$ and Sharpe Ratio.

- Key Metrics:

  - Long-Short Spread:

$$\text{Spread} = \mu_{Q10} - \mu_{Q1}$$

    Annualized spread > 10% = excellent.

  - Sharpe Ratio:

$$\text{Sharpe Ratio} = \frac{\mu_p - r_f}{\sigma_p}$$

    > 1.5 (excellent), > 2.0 (top-tier).

3.2.4 Statistical Significance (t-statistic)

- Definition: Tests whether factor returns are statistically significant.

- Formula:

$$t = \frac{\alpha}{\text{SE}(\alpha)}$$

  where$(\alpha) = factorreturn, (\text{SE}(\alpha)) = standarderror.$

- Evaluation Criteria:

  - $(|t| > 2)$(significant at 95% confidence),$(|t| > 3)$(highly significant).

3.2.5 Comprehensive Evaluation Framework

Proposes a four-dimensional framework:

1. Predictive Power: IC, IR.

2. Economic Significance: Factor return, long-short spread.

3. Statistical Robustness: t-statistic, IC stability.

4. Practical Feasibility: Turnover, transaction cost sensitivity.

Workflow: Factor Screening → IC/IR Testing → Quantile Backtest → Statistical Significance → Cost Adjustment → Out-of-Sample Validation

3.2.6 Empirical Case Study

Example: Low-Volatility Factor in A-shares:

1. IC Analysis: Monthly IC = 0.07, IR = 1.4.

2. Quantile Test: Q10-Q1 spread = 12% (annualized), Sharpe Ratio = 1.9.

3. Regression: Factor return = 6.5% (t = 3.1).

4. Turnover: 18% monthly, net return = 5.8% post-cost.

Conclusion: The factor passes all four dimensions.

IV Project Value Analysis

4.1 Innovative Value in the Financial Field

This project significantly enhances the efficiency and accuracy of quantitative finance research through a multi-agent collaborative automated factor factory, providing the following core values to financial institutions and analysts.

Traditional factor mining relies on expert experience and typically takes several weeks to complete data cleaning, feature construction, and backtesting. This project achieves full automation from natural language instructions to factor formulas, code generation, and validation through the collaboration of three agents: FactorAgent, CalculatorAgent, and CodeAgent, reducing the development cycle from weeks to hours.

4.2 Technical Integration and Industry Empowerment

4.2.1 Low-Threshold Interaction

The system supports natural language input, allowing users to obtain production-level factor code and economic explanations without writing code, lowering the technical barriers to quantitative research.

4.2.2 Full-Stack Solution

The system integrates high-frequency data processing (Memory Map has a read speed 260 times faster than Parquet), operator libraries (such as ARIMA, wavelet transform), and multi-model collaboration (Qwen-max for logic generation + Claude for programming). It covers the entire chain of needs from data input to strategy deployment and supports seamless API integration with quantitative trading platforms (e.g., DolphinDB).

4.3 Market Value

(1) Compared to traditional human factor mining, where traditional quantitative analysts can only produce 1–2 efficient factors per month, the efficiency is low. Moreover, in the micro-transaction structure of the A-share market, for example in February this year, February last year, and October last year, traditional human factor mining cannot keep pace with the market to use new market factors for transaction characterization.

(2) Compared to genetic algorithms that exhaustively enumerate all permutations of input features and operators, the factors constructed by this project have greater complexity and save a large amount of computational resources used for exhaustive search.

(3) Against the backdrop of accelerated industry rotation and changes in the micro-transaction market, this year, many public and private institutions have experienced a large number of factor failures in their factor libraries and urgently need a batch of new factors to describe the current A-share market.

V Project Summary

This project not only validates the practicality of LLMs in quantitative finance but also pioneers an interpretable and highly iterative intelligent factor development paradigm, providing a complete example from technology to commercial value for the industry.

5.1 Core Achievements

5.1.1 Successful Construction of a Three-Agent Collaborative Pipeline

(1) FactorAgent: Generates multiple calculation methods, input fields, and economic explanations based on high-quality factor templates (e.g., “Momentum and Liquidity Synergy Factor”).

(2) CalculatorAgent: Abstracts the calculation methods generated by FactorAgent into mathematical formulas, matches them with operators in the operator library, constructs operator functions if unconstructed operators are found, logs the process, and passes it to CodeAgent.

(3) CodeAgent: Generates production-level code using the operator functions provided by CalculatorAgent and the input fields provided by FactorAgent.

5.1.2 Factor Library Construction

Developed a large number of high-IC factors covering multiple dimensions such as price and volume, fundamental analysis, and microstructure, with version control and metadata management (historical performance, correlation matrix).

5.2 Key Decisions and Challenges Addressed

5.2.1 Technology Selection

(1) Abandoning Genetic Algorithms: Due to their “black-box” nature, which makes factors uninterpretable, we switched to an

LLM + symbolic logic fusion approach (e.g., forcing FactorAgent to use a predefined operator library to ensure transparent economic logic).

(2) Choosing the APPL Framework: Compared to Dify/Coze, APPL's decorator encapsulation capability is more suitable for complex financial logic (e.g., abnormal value processing in RollingQuantile).

5.2.2 Performance Optimization

(1) Data Reading Bottleneck: Transitioned from DolphinDB native queries to memory-mapped wide tables, significantly improving reading speed.

(2) CodeAgent Debugging: Reduced runtime errors by 35% through strict prompt engineering (e.g., enforcing 4-space indentation, index alignment assertions).

5.3 Experience Summary

(1) Standardization of the operator library is the key to success: All operators must be encapsulated as functions, called using column names.

(2) Include NaN handling (e.g., WaveletTransform with boundary checks).

VI Evaluation of Sponsors

该项目实现因子挖掘自动化，完善公司投研体系，为现有遗传算法因子工厂提供了新维度。

赞助商联系人姓名：黄磊鑫

联系电话：19883279980

签名：黄磊鑫

Appendix

1-project plan

GPT-based Multi-Agent System for Factor Mining

1. Project Name

Development of GPT-based Multi-Agent System for Price-Volume Factor Mining

2. Project Scope

Overview: The goal of this project is to develop a GPT-based multi-agent system for mining price and volume factors in financial markets. The system will leverage the capabilities of Large Language Models (LLMs) and advanced techniques like prompt engineering and retrieval-augmented generation (RAG). The system will consist of three primary agents:

FactorGPT: Responsible for generating factor expressions.
CodeGPT: Transforms factor expressions into executable code for evaluation.
EvalGPT: Conducts backtesting and suggests optimizations based on performance.

Expected Outcomes: A fully integrated system capable of automatically generating, optimizing, and testing price-volume-based factors. The system will be evaluated using key performance metrics such as IC (Information Coefficient) and RankIC.

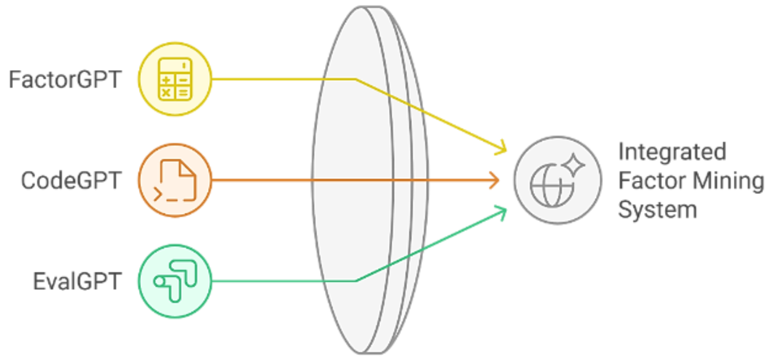


Figure 1: GPT-Based Multi-Agent Factor Mining System Architecture

3. Project Schedule

Phase 1: Research & Workflow Design (2 Weeks, Feb 10th, 2025-Feb 23rd, 2025)
Objective: Define system architecture and agent interactions, review the state-of-the-art in GPT-based systems for financial factor mining.
Tasks: System design, workflow planning, and requirement gathering.

Phase 2: Agent Development & System Integration (4 Weeks, Feb 24th, 2025-Mar 23rd, 2025)
Objective: Develop the individual agents (FactorGPT, CodeGPT, EvalGPT) and integrate them into a cohesive system.
Tasks: Model training, LLM integration, prompt engineering, RAG integration.

Phase 3: Testing & Optimization (4 Weeks, Mar 24th, 2025-Apr 20rd, 2025)
Objective: Conduct thorough testing, backtesting of the generated factors, and optimize the model performance.
Tasks: Evaluate factor performance, fine-tune agents, iterate feedback loops for optimization.

Phase 4: Final Review & Reporting (2 Week, Apr 21th, 2025-May 4th, 2025)
Objective: Prepare the final project report and deliver the working demo.
Tasks: Documentation, preparation of the final report, and demo preparation.

4. Group Members and Preliminary Roles

Project Manager Zhiyu Liang: Oversees project timeline, resources, and communication between team members and stakeholders.
Lead Developer Ruopu Li: Responsible for the development of the multi-agent system, integrating GPT models into Dify.AI and other development platforms.
GenAI Specialist Yuyang Shen: Focuses on LLM integration, prompt engineering, and ensuring effective use of RAG techniques in the workflow.

Data Scientist Han Chen: Responsible for designing financial backtests, analyzing performance metrics, and optimizing factor expressions.

Documentation Lead Jiaxi Zhao: Prepares and organizes reports, project documentation, and user guides.

5. Project Deliverables

Functional Multi-Agent System: A working GPT-based multi-agent system for factor mining, with three agents (FactorGPT, CodeGPT, EvalGPT) fully integrated and operational.

Performance Evaluation: Results of backtests and financial metrics demonstrating the system's ability to identify and optimize price-volume factors.

Reports:

Interim report summarizing progress and challenges.

Final report covering design, development, testing, and results.

Demo/Prototype: A live demonstration or prototype showcasing the functionality of the system and its capabilities in real-time factor mining.

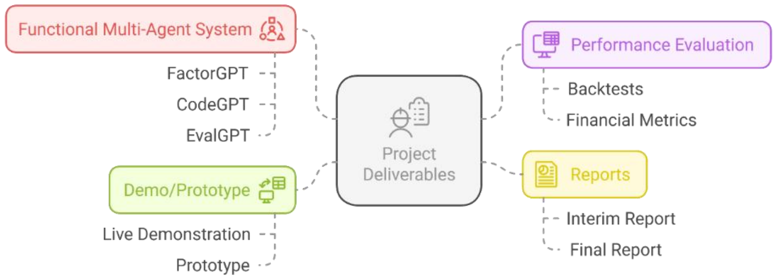


Figure 2: Multi-Agent System Deliverables

6. Project Objectives

Primary Objective: The project aims to automate the process of mining and optimizing price-volume factors using GPT-based multi-agent systems.

Business Problem: Financial institutions and analysts face challenges in developing, optimizing, and testing new quantitative factors for price and volume analysis. Manual processes are time-consuming, error-prone, and difficult to scale.

Expected Benefits: Increased efficiency in factor generation and optimization.

Automation of backtesting and factor performance evaluation.

Reduction in manual effort and human error, leading to more reliable financial models.

7. Potential Obstacles

Data Quality and Usability Issues:

Financial market data may be incomplete, noisy, or inconsistent, affecting the accuracy of factor mining. Additionally, data acquisition and usage are subject to legal and compliance restrictions.

Solutions: Establish strict data cleaning and preprocessing procedures to ensure data quality and consistency.

Use machine learning algorithms to automatically detect and repair anomalies in the data.

Integrate data from multiple sources to enhance data completeness and reliability.

Model Overfitting Risk:

Large language models may overfit during training, leading to poor performance in practical applications.

Solutions: Use time-series cross-validation to assess model performance across different periods.

Apply regularization techniques (such as L1 and L2 regularization) to limit model complexity.

Evaluate model performance using multiple metrics, such as IC, RankIC, and Sharpe ratio.

Project Schedule and Resource Management:

Projects involve multiple stages and team members, which can lead to issues in schedule management and resource allocation. For example, poor communication among team members may cause task delays, and insufficient computing resources may affect model training and testing.

Solutions: Clearly define the roles and responsibilities of each team member in the project plan.

Establish a regular project progress review mechanism to identify and address schedule deviations.

Plan computing resources in advance to ensure sufficient support for the project.

2-interim report

GPT-based Multi-Agent System for Factor Mining

I Project Introduction

1.1. Project Background and Objectives

Financial institutions encounter substantial challenges when developing robust investment factors. Traditional methods are labor-intensive, time-consuming, and susceptible to biases. Analysts face significant data quality issues, including missing values, outliers, survivorship bias, and look-ahead bias. [1] Computational limitations further complicate processing large datasets and conducting extensive backtests. Overfitting risks consistently threaten factor reliability across diverse market conditions. Additionally, research biases, such as confirmation and publication bias, negatively impact the integrity and objectivity of factor development.

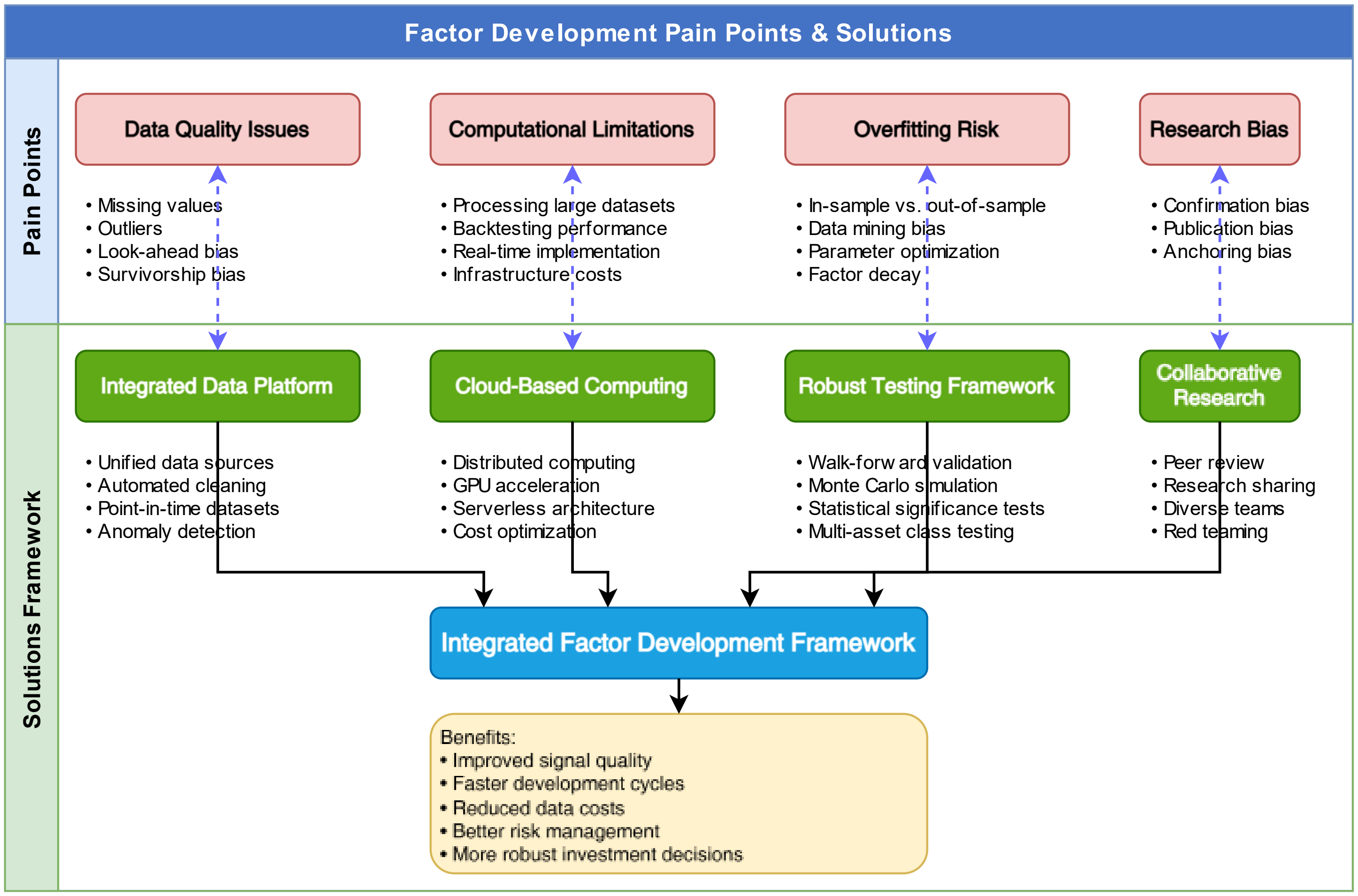


Fig 1.1. Challenges in Factor Development for Financial Institutions and Analysts

Automating the factor mining process provides substantial value to financial institutions by significantly accelerating factor development cycles and maintaining methodological consistency. Leveraging artificial intelligence, automated systems uncover subtle patterns in complex financial data, often overlooked by human analysts. Systematic validation protocols reduce overfitting and research biases through comprehensive testing frameworks. [2]By minimizing manual tasks, institutions can allocate quantitative expertise towards higher-value strategic activities, enhancing factor quality and reducing operational expenses.

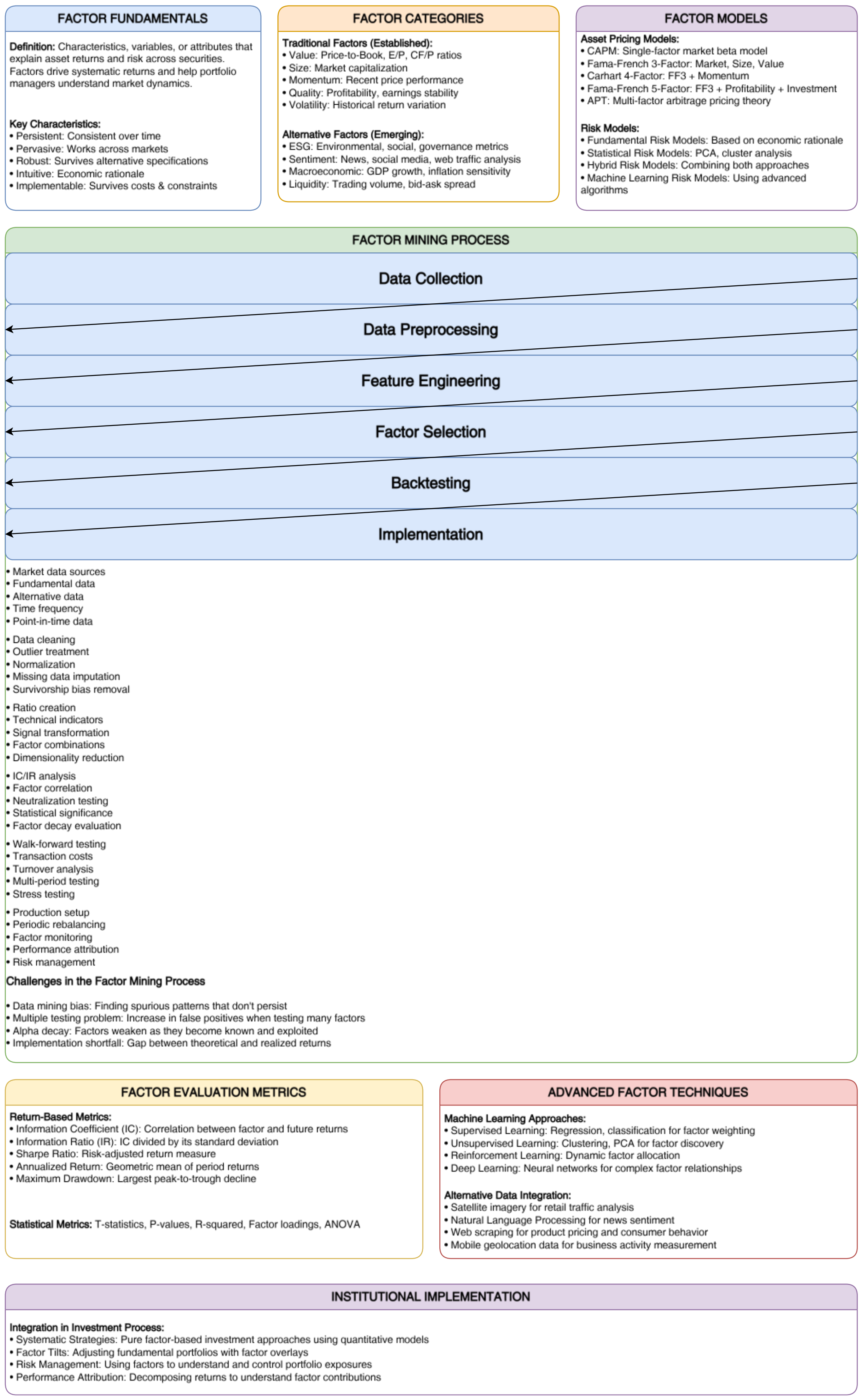

FACTOR FUNDAMENTALS
Definition: Characteristics, variables, or attributes that explain asset returns and risk across securities. Factors drive systematic returns and help portfolio managers understand market dynamics.
Key Characteristics:
• Persistent: Consistent over time
• Pervasive: Works across markets
• Robust: Survives alternative specifications
• Intuitive: Economic rationale
• Implementable: Survives costs & constraints
FACTOR CATEGORIES
Traditional Factors (Established):
• Value: Price-to-Book, E/P, CF/P ratios
• Size: Market capitalization
• Momentum: Recent price performance
• Quality: Profitability, earnings stability
• Volatility: Historical return variation
Alternative Factors (Emerging):
• ESG: Environmental, social, governance metrics
• Sentiment: News, social media, web traffic analysis
• Macroeconomic: GDP growth, inflation sensitivity
• Liquidity: Trading volume, bid-ask spread
FACTOR MODELS
Asset Pricing Models:
• CAPM: Single-factor market beta model
• Fama-French 3-Factor: Market, Size, Value
• Carhart 4-Factor: FF3 + Momentum
• Fama-French 5-Factor: FF3 + Profitability + Investment
• APT: Multi-factor arbitrage pricing theory
Risk Models:
• Fundamental Risk Models: Based on economic rationale
• Statistical Risk Models: PCA, cluster analysis
• Hybrid Risk Models: Combining both approaches
• Machine Learning Risk Models: Using advanced algorithms
FACTOR MINING PROCESS
Data Collection
Data Preprocessing
Feature Engineering
Factor Selection
Backtesting
Implementation
• Market data sources
• Fundamental data
• Alternative data
• Time frequency
• Point-in-time data
• Data cleaning
• Outlier treatment
• Normalization
• Missing data imputation
• Survivorship bias removal
• Ratio creation
• Technical indicators
• Signal transformation
• Factor combinations
• Dimensionality reduction
• IC/IR analysis
• Factor correlation
• Neutralization testing
• Statistical significance
• Factor decay evaluation
• Walk-forward testing
• Transaction costs
• Turnover analysis
• Multi-period testing
• Stress testing
• Production setup
• Periodic rebalancing
• Factor monitoring
• Performance attribution
• Risk management
Challenges in the Factor Mining Process
• Data mining bias: Finding spurious patterns that don't persist
• Multiple testing problem: Increase in false positives when testing many factors
• Alpha decay: Factors weaken as they become known and exploited
• Implementation shortfall: Gap between theoretical and realized returns
FACTOR EVALUATION METRICS
Return-Based Metrics:
• Information Coefficient (IC): Correlation between factor and future returns
• Information Ratio (IR): IC divided by its standard deviation
• Sharpe Ratio: Risk-adjusted return measure
• Annualized Return: Geometric mean of period returns
• Maximum Drawdown: Largest peak-to-trough decline
Statistical Metrics: T-statistics, P-values, R-squared, Factor loadings, ANOVA
ADVANCED FACTOR TECHNIQUES
Machine Learning Approaches:
• Supervised Learning: Regression, classification for factor weighting
• Unsupervised Learning: Clustering, PCA for factor discovery
• Reinforcement Learning: Dynamic factor allocation
• Deep Learning: Neural networks for complex factor relationships
Alternative Data Integration:
• Satellite imagery for retail traffic analysis
• Natural Language Processing for news sentiment
• Web scraping for product pricing and consumer behavior
• Mobile geolocation data for business activity measurement
INSTITUTIONAL IMPLEMENTATION
Integration in Investment Process:
• Systematic Strategies: Pure factor-based investment approaches using quantitative models
• Factor Tilts: Adjusting fundamental portfolios with factor overlays
• Risk Management: Using factors to understand and control portfolio exposures
• Performance Attribution: Decomposing returns to understand factor contributions

Fig 1.2. Financial factor concepts and factor mining framework

1.2. System Architecture Overview: Three-Agent Collaboration Workflow

FactorAgent, the Factor Agent, serves as the analytical core of the system, examining diverse data sources such as market, fundamental, alternative, and historical factor data. [3]It identifies relevant patterns and generates theoretically sound factor hypotheses, clearly defining their implementation specifications for downstream execution.

CodeAgent, the Code Agent, translates conceptual factor designs into executable code implementations. It optimizes algorithms to enhance computational efficiency, performs comprehensive unit testing to ensure reliability, and delivers production-ready factor implementations. This Agent effectively bridges theoretical factor designs with practical application.

EvalAgent, the Evaluation Agent, rigorously assesses implemented factors using key performance metrics, including Information Coefficient, Information Ratio, Sharpe Ratio, and statistical significance. Applying stringent quality criteria, EvalAgent determines the viability of factors for inclusion in the production library or triggers further refinement through iterative feedback loops.

The three Agents collaborate within a continuous improvement cycle, iteratively refining factor quality through systematic revisions at both the conceptual and coding stages.

1.3. Key Deliverables

The project will deliver an Integrated Factor Development Platform featuring an intuitive user interface and an automated data pipeline capable of handling data ingestion, cleaning, and preprocessing tasks seamlessly. The system incorporates three specialized AI Agents—FactorAgent, CodeAgent, and EvalAgent—operating within a robust cloud-based computational infrastructure leveraging GPU acceleration. It will also feature a continuous integration and continuous deployment (CI/CD) pipeline to streamline factor testing, accompanied by a real-time monitoring dashboard that facilitates the ongoing evaluation of factor performance.

In terms of documentation and reporting, the project provides comprehensive system documentation, complete with detailed architecture diagrams. Additionally, user manuals clearly outline platform operation and maintenance processes, complemented by technical documentation specifying API endpoints and integration protocols. [4]Performance benchmarking reports will be provided, highlighting comparisons between AI-generated factors and traditional methods, alongside quarterly factor trend analysis reports focused on identifying and evaluating emerging factor categories.

The system's factor library comprises a carefully curated repository of thoroughly validated investment factors, each supported by detailed documentation that articulates the theoretical foundations and expected behaviors. Metadata associated with these factors include historical performance metrics and correlation statistics. Each factor will come with optimized implementation code and configurable parameters, as well as recommendations for practical applications and integration strategies within portfolio construction. A dedicated version control system is included to effectively manage and track the evolution of these investment factors. [5]Overall, this multi-Agent integrated system represents a significant advancement in quantitative finance, equipping financial institutions with a powerful and efficient framework for discovering, implementing, and validating investment factors, ultimately enhancing accuracy, reliability, and performance in investment decision-making.

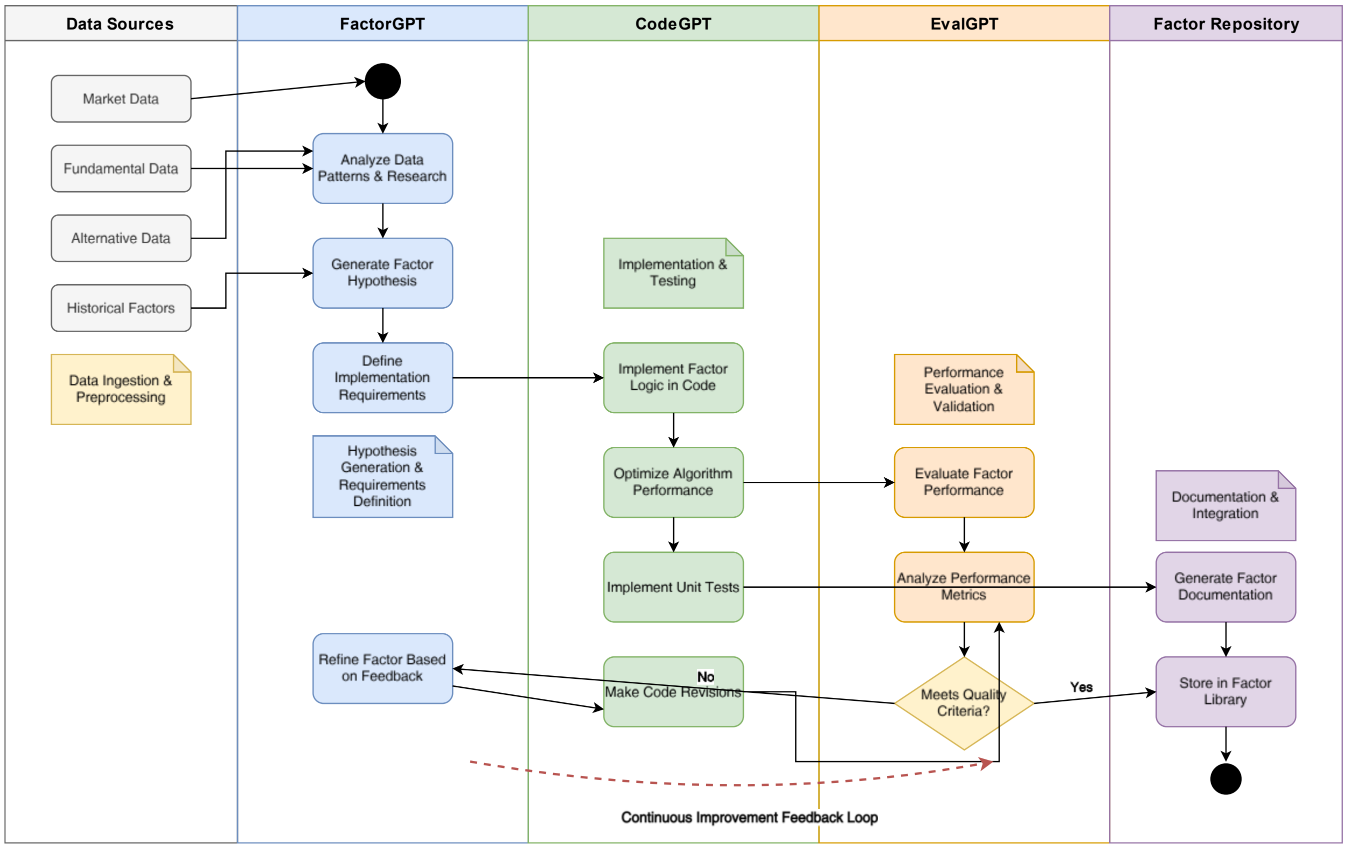

Data Sources
FactorGPT
CodeGPT
EvalGPT
Factor Repository
Market Data
Fundamental Data
Alternative Data
Historical Factors
Data Ingestion & Preprocessing
Analyze Data Patterns & Research
Generate Factor Hypothesis
Define Implementation Requirements
Hypothesis Generation & Requirements Definition
Refine Factor Based on Feedback
Implementation & Testing
Implement Factor Logic in Code
Optimize Algorithm Performance
Implement Unit Tests
Make Code Revisions
Performance Evaluation & Validation
Evaluate Factor Performance
Analyze Performance Metrics
Meets Quality Criteria?
No
Yes
Documentation & Integration
Generate Factor Documentation
Store in Factor Library
Continuous Improvement Feedback Loop

Fig 1.3. Automated Factor Mining System - Three-Agent Collaboration Workflow

II User Needs and Interactivity Design

2.1. Analysis of User Needs

The program's primary constituencies can be categorized into three groups: financial institutions, financial analysts, and academics engaged in financial market research. [6]Through the collaboration of multiple intelligences, the program is able to fulfil the needs of the aforementioned groups of people in discovering investment factors that will have an impact on the future market.

(1) Financial Institutions

Financial institutions are very concerned about the timeliness of factor mining because market conditions change rapidly and inefficient factor mining can miss the best time to invest. This project has the potential to significantly accelerate the acquisition of high IC factors by financial institutions. The automation of mining operations has been shown to achieve results in a matter of hours that would otherwise take weeks in manual operations, thus meeting the efficiency requirements of financial institutions for trading information.

(2) Financial Analysts

The demand for factor analysis by financial analysts is characterized by speed and accuracy in today's market, where the amount of information has increased exponentially. Consequently, this scenario imposes a heightened demand for the overall caliber of financial analysts, emphasizing their financial acumen and coding proficiency. This project aims to alleviate the stress and complexity of financial analysts' work by integrating the factor discovery and validation processes, ensuring the validation of factors prior to the generation of the final output. Moreover, the code-free working method enables financial analysts to operate within the current market environment and perform more complex analysis tasks, irrespective of their technical background.

(3) Academic researchers

In the process of conducting academic research on financial markets, there are currently no tools for integrated mining and research on factors that can be factors. Furthermore, general researchers lack data sources with high accuracy and timeliness in the process of conducting financial research. The project aims to address these challenges by introducing an integrated factor mining process, which is expected to reduce the cost and technical complexity of academic research. The project utilizes automated exploration to identify factors that would otherwise require longer to discover by traditional manual computational methods. Additionally, it employs customized computational methods to validate academic hypotheses, thereby enhancing the reliability and validity of research outcomes. [7]Furthermore, the integration of RAG technology within the project, in conjunction with the enhancement of Internet search capabilities, serves to address the paucity of data sources, thereby enhancing the practical application value of academic research outcomes.

2.2. User & System Interaction Process

1. Data Input:

The user provides the relevant knowledge base (e.g. market transaction data, good factor database, operator database, etc.).

2. Requirement Description:

(1) The user inputs the factor mining goal in natural language (e.g. "find volume-price relationship factor").

(2) FactorAgent parses the requirement and combines it with the knowledge base to generate candidate factors (e.g. "Volume surge by 2 times").

3. Factor generation and code conversion:

(1) The FactorAgent outputs factors and related information, including factor formulas, involved fields and meanings.

(2) The CalculatorAgent uses the factor expression, the operators in the database to merge the calculation method and verify that the factor formula can be realized, outputting the complete mathematical expression and the list of operators used as well as the verification results

(3) The CodeAgent automatically converts formulas to Python code, calling a database of predefined operators (e.g., calculating moving averages).

4. Factor Validation and Output:

(1) CodeAgent executes the code in order to validate the IC values of the factors.

(2) The user is provided with the final result, including the name and meaning of the factor, the corresponding formula and fields used, and the IC value of the factor.

The user is merely required to input the knowledge base and the factor mining target in natural language prior to initiating the execution process. Thereafter, the system will automatically calculate the final factor mining results. The user is not required to perform complex operations; rather, they can simply realize the factor mining process that would otherwise take a long time and complex technology by taking advantage of the advantages of the large model, i.e. natural language recognition.

2.3. Interactivity Design

The project will be dependent on the DIfy.ai platform, leveraging its intuitive and user-friendly interface to facilitate the development process. The underlying rationales for this choice are as follows:

1. Primarily, the platform's interactive interface aligns with the prevalent user habits concerning the utilization of a conventional chat-style interface window. Furthermore, users with limited technical proficiency can still achieve system automation output through zero-code system calls.

2. Secondly, the platform's deployment for subsequent projects is a key consideration. The DIfy.ai development interface is user-friendly, facilitating straightforward modification of the system prompt to suit the needs of the second deployment of proprietary knowledge base.

3. Enterprise users will be pleased to hear that the platform can provide enterprise-level security and expansion to meet their further requirements in the interaction of the project. For example, factors can be docked with a key to the quantitative trading platform API.

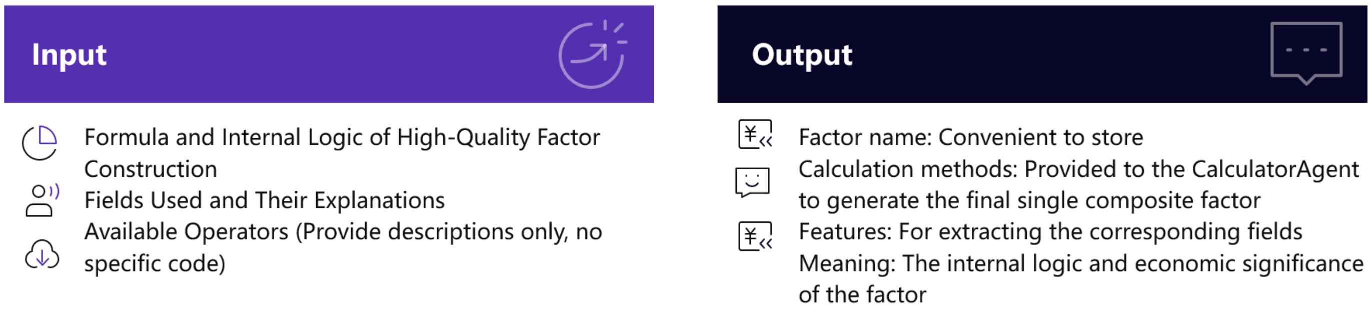
The utilization of the low-code development features of the DIfy.ai platform enables non-technical personnel to participate efficiently in the deployment and development of the project, thereby reducing communication barriers between technical personnel and financial professionals. This will be a key point for enterprises to support and select this project for future enterprise proprietary use.

III Techniques and Innovations

This part will describe the detailed usages of different LLM agents in the workflow and techniques used in training or tunning the models, as well as the innovations created in the project, especially compared to traditional factor digging methods.

3.1. FactorAgent

FactorGPT plays an essential role in the process of factor construction, offering a robust and automated solution for developing high-quality factors that are integral to analytical and predictive modeling frameworks. [8]As depicted in the diagram, the agent accepts a well-structured set of inputs, which includes the internal logic governing high-quality factor construction, the fields utilized along with their detailed explanations, and a suite of available operators. These operators are designed to provide dual functionality: they offer partial function support to the CodeAgent, ensuring the generation of accurate and reliable code, while simultaneously equipping the FactorAgent with the capability to construct complex operators that enhance the sophistication of the factors being developed. Upon processing these inputs, FactorGPT produces outputs that encompass the factor name, a range of calculation methods, extracted features, and their corresponding meanings, thereby providing a deep understanding of the factor's significance and application. One of its standout features is the provision of multiple calculation methods, which introduces variety and adaptability, catering to diverse analytical needs. This entire process is executed by the agent itself, without requiring human operations, thereby ensuring consistency, reducing the potential for errors, and significantly improving efficiency in the factor development pipeline.


Input

Formula and Internal Logic of High-Quality Factor Construction
Fields Used and Their Explanations
Available Operators (Provide descriptions only, no specific code)

Output

Factor name: Convenient to store
Calculation methods: Provided to the CalculatorAgent to generate the final single composite factor
Features: For extracting the corresponding fields
Meaning: The internal logic and economic significance of the factor

Fig 3.1. Input And Output of FactorGPT

3.2. CalculatorAgent

It adopts CalculatorGPT as the CalculatoraAgent. CalculatorGPT serves as a critical component in the factor construction pipeline, acting as a specialized agent that processes the output from FactorGPT to ensure the accuracy and functionality of the constructed factors. [9]As illustrated in the diagram, CalculatorGPT receives inputs from FactorGPT, which include the operator library with corresponding code and the formula constructed by the FactorAgent. The core task of CalculatorGPT is to synthesize the formula provided by FactorGPT into a single, suitable mathematical expression, ensuring that the corresponding code is accurate and fully implementable. It verifies the factor formula to confirm its correctness, addressing any potential issues in the code implementation details that might arise during the process.

Additionally, CalculatorGPT checks for the presence of unimplemented operators; if any are identified, it takes the initiative to generate or implement these missing operators, thereby ensuring the formula's completeness. The output of CalculatorGPT is comprehensive, consisting of a complete mathematical expression, a list of used operators, and the results of the verification process, which confirm whether the operators meet the necessary requirements. This output is then provided as input to the CodeAgent for further processing.

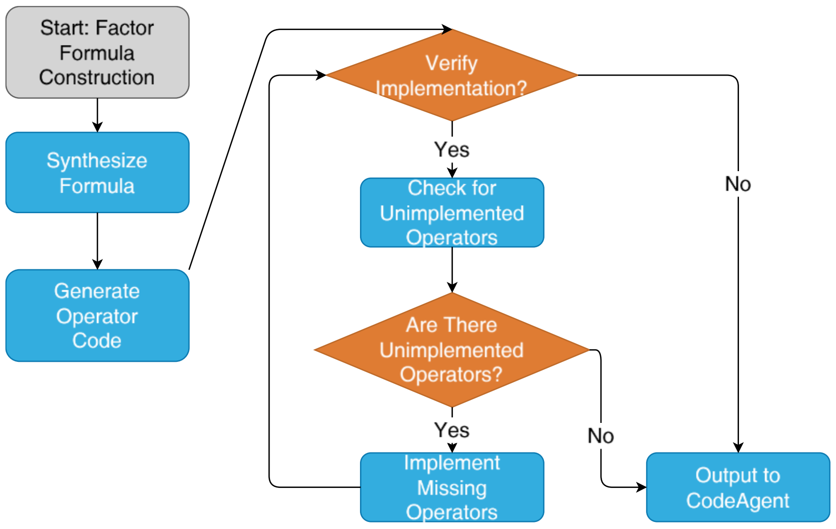

Start: Factor Formula Construction
Synthesize Formula
Generate Operator Code
Verify Implementation?
Yes
No
Check for Unimplemented Operators
Are There Unimplemented Operators?
Yes
No
Implement Missing Operators
Output to CodeAgent

Fig 3.2. Workflow of CalculatorGPT

3.3. CodeAgent

CodeAgent represents the final stage in the factor construction pipeline, functioning as an executor and optimizer of factor formulas to ensure the seamless generation of executable code. In the project, it uses CodeGPT. As outlined in the diagram, CodeGPT takes as input the operator code obtained from CalculatorAgent by querying the operator library, the formula constructed by CalculatorAgent, and the data fields provided by FactorAgent, which include a list of data fields essential for factor computation. The primary objective of CodeAgent is to generate executable code by strictly defining data types, adhering to format standards, and providing output specifications, all while incorporating automatic debugging to address potential errors. CodeAgent tackles a range of challenges encountered during code generation, such as indentation issues, keyword conflicts, chart generation unrelated to the code, content misalignment between applied results and original indices, operator rewrites, and data merging problems, particularly the frequent and inefficient use of rolling for repeated calculations. These problems are solved by related techniques which will be presented later. The output of CodeAgent is a refined, executable codebase that meets the specified requirements. CodeAgent enhances the reliability and usability of the factor construction workflow.

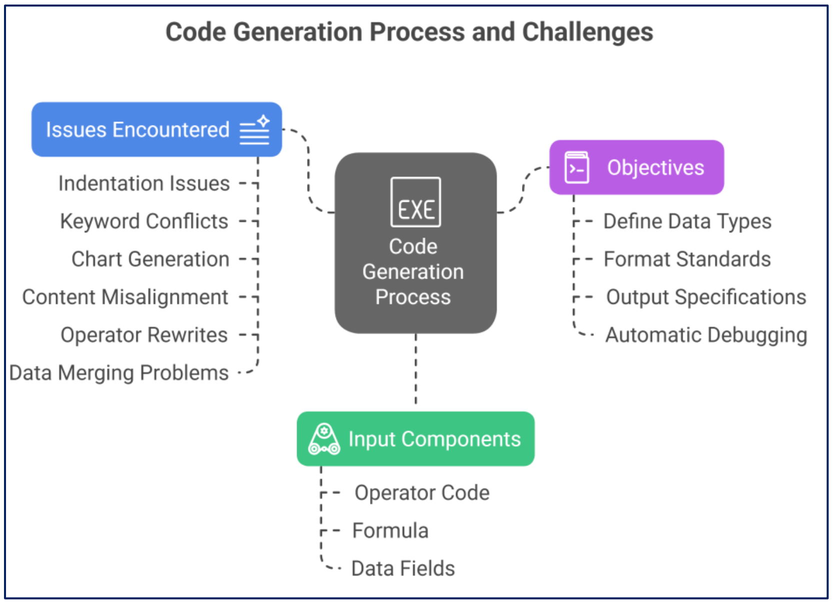


Fig 3.3. Structure of CodeGPT

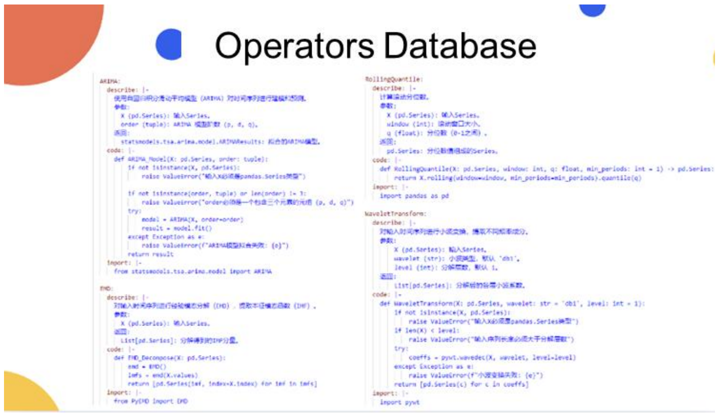
3.4. Development Tools and Other Training Techniques

The workflow presented in this project is designed using APPL, a platform accessible via its official website (https://appl-team.github.io/appl/). APPL, which stands for Advanced Prompting and Processing Language, is a sophisticated workflow design platform tailored for large-scale model applications. Its primary strength lies in its ability to encapsulate code and complex functions through decorators, enabling integration of complex logic within the workflow. This capability allows large models to finish advanced operations and generate high-level intelligence, making APPL an ideal choice for projects requiring robust computational frameworks. Compared to alternatives like Dify and Cozi, which tend to adopt a more frontend-oriented approach, APPL aligns more closely with the raw development process. This proximity combined with its flexibility in handling complex logic, makes APPL the preferred choice for this project, ensuring that the factor construction pipeline can be executed with precision and efficiency.

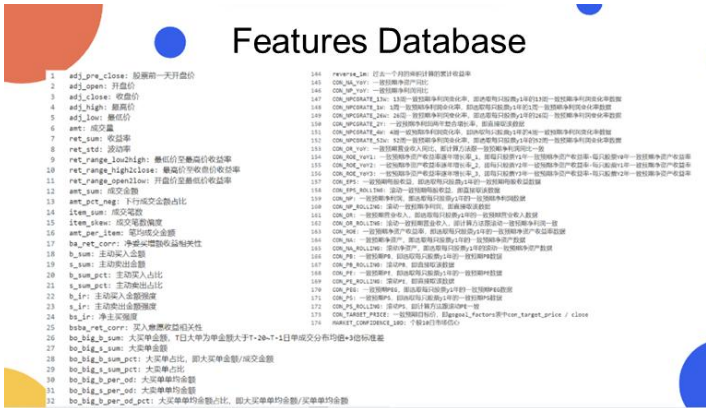
In addition to APPL, this project leverages advanced techniques such as Chain of Thought (CoT) and ReAct to enhance the performance of large models within the workflow. CoT, or Chain of Thought, is a reasoning strategy that encourages step-by-step thinking to improve model outputs. For instance, in the CodeAgent component, CoT is employed to systematically generate executable code by breaking down the process into logical, incremental steps, thereby boosting the model's accuracy and reducing errors. On the other hand, ReAct (Reasoning and Acting) enables the model to evaluate inputs and determine the optimal course of action. In the context of CodeAgent, ReAct is used to decide whether to utilize the calculator provided as input. Direct input of the calculator often leads CodeAgent to rewrite it unnecessarily; however, by applying ReAct, the relationship between CodeAgent and the calculator is better managed, allowing the agent to make informed decisions about leveraging existing tools without redundant modifications. These techniques collectively enhance the efficiency and reliability of the factor construction process, and solve the problems mentioned in CodeAgent before.

Furthermore, the project incorporates Retrieval-Augmented Generation (RAG) to address the challenges posed by lengthy prompt inputs, particularly in the FactorAgent component. RAG enhances the model's ability to generate complex ideas by retrieving relevant knowledge after forming an initial concept. In this workflow, FactorAgent benefits from RAG by accessing a knowledge base to synthesize intricate factor logic, as depicted in the provided diagram. The retrieved data includes detailed factor quality analyses, factor bases, calculator outputs, detailed inputs, and input segments, which collectively inform the agent's understanding of factor construction. This retrieval process ensures that FactorAgent can construct well-informed and sophisticated factors without being overwhelmed by the extensive prompt inputs, thereby improving the overall quality of the generated factors.

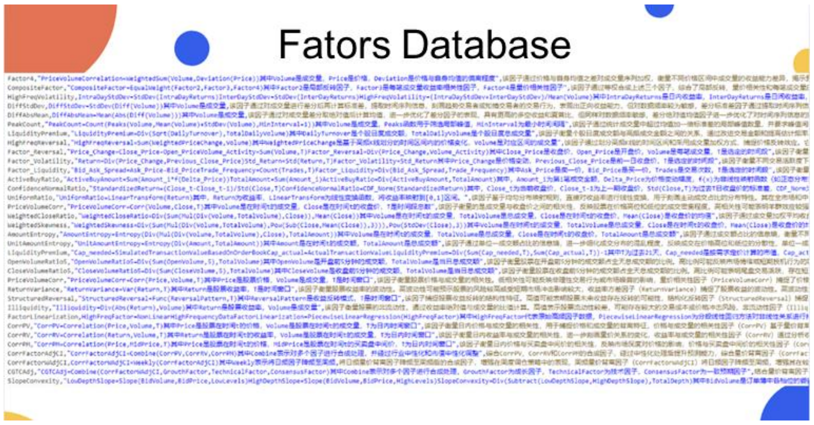
By integrating APPL's workflow design with CoT, ReAct, and RAG, this project establishes a robust framework for factor construction. The use of APPL ensures a development-centric approach, while CoT and ReAct optimize the reasoning and decision-making capabilities of the agents involved. Meanwhile, RAG addresses the challenge of managing complex inputs, enabling the workflow to produce high-quality factors efficiently.



```
ARIMA:
  describe: |-
    使用自回归积分滑动平均模型 (ARIMA) 对时间序列进行建模和预测。
    参数:
      X (pd.Series): 输入Series。
      order (tuple): ARIMA 模型阶数 (p, d, q)。
    返回:
      statsmodels.tsa.arima.model.ARIMAResults: 拟合的ARIMA模型。
  code: |-
    def ARIMA_Model(X: pd.Series, order: tuple):
        if not isinstance(X, pd.Series):
            raise ValueError("输入X必须是pandas.Series类型")

        if not isinstance(order, tuple) or len(order) != 3:
            raise ValueError("order必须是一个包含三个元素的元组 (p, d, q)")
        try:
            model = ARIMA(X, order=order)
            result = model.fit()
        except Exception as e:
            raise ValueError(f"ARIMA模型拟合失败: {e}")
        return result
  import: |-
    from statsmodels.tsa.arima.model import ARIMA

EMD:
  describe: |-
    对输入时间序列进行经验模态分解 (EMD) ，提取本征模态函数 (IMF) 。
    参数:
      X (pd.Series): 输入Series。
    返回:
      List[pd.Series]: 分解得到的IMF分量。
  code: |-
    def EMD_Decompose(X: pd.Series):
        emd = EMD()
        imfs = emd(X.values)
        return [pd.Series(imf, index=X.index) for imf in imfs]
  import: |-
    from PyEMD import EMD

RollingQuantile:
  describe: |-
    计算滚动分位数。
    参数:
      X (pd.Series): 输入Series。
      window (int): 滚动窗口大小。
      q (float): 分位数 (0-1之间) 。
    返回:
      pd.Series: 分位数值组成的Series。
  code: |-
    def RollingQuantile(X: pd.Series, window: int, q: float, min_periods: int = 1) -> pd.Series:
        return X.rolling(window=window, min_periods=min_periods).quantile(q)
  import: |-
    import pandas as pd

WaveletTransform:
  describe: |-
    对输入时间序列进行小波变换，提取不同频率成分。
    参数:
        X (pd.Series): 输入Series。
        wavelet (str): 小波类型，默认 'db1'。
        level (int): 分解层数，默认 1。
    返回:
        List[pd.Series]: 分解后的各层小波系数。
  code: |-
    def WaveletTransform(X: pd.Series, wavelet: str = 'db1', level: int = 1):
        if not isinstance(X, pd.Series):
            raise ValueError("输入X必须是pandas.Series类型")
        if len(X) < level:
            raise ValueError("输入序列长度必须大于分解层数")
        try:
            coeffs = pywt.wavedec(X, wavelet, level=level)
        except Exception as e:
            raise ValueError(f"小波变换失败: {e}")
        return [pd.Series(c) for c in coeffs]
  import: |-
    import pywt
```



```
1   adj_pre_close: 股票前一天开盘价
2   adj_open: 开盘价
3   adj_close: 收盘价
4   adj_high: 最高价
5   adj_low: 最低价
6   amt: 成交量
7   ret_sum: 收益率
8   ret_std: 波动率
9   ret_range_low2high: 最低价至最高价收益率
10  ret_range_high2close: 最高价至收盘价收益率
11  ret_range_open2low: 开盘价至最低价收益率
12  amt_sum: 成交金额
13  amt_pct_neg: 下行成交金额占比
14  item_sum: 成交笔数
15  item_skew: 成交笔数偏度
16  amt_per_item: 笔均成交金额
17  ba_ret_corr: 净委买增额收益相关性
18  b_sum: 主动买入金额
19  s_sum: 主动卖出金额
20  b_sum_pct: 主动买入占比
21  s_sum_pct: 主动卖出占比
22  b_ir: 主动买入金额强度
23  s_ir: 主动卖出金额强度
24  bs_ir: 净主买强度
25  bsba_ret_corr: 买入意愿收益相关性
26  bo_big_b_sum: 大买单金额，T日大单为单金额大于T-20~T-1日单成交分布均值+3倍标准差
27  bo_big_s_sum: 大卖单金额
28  bo_big_b_sum_pct: 大买单占比，即大买单金额/成交金额
29  bo_big_s_sum_pct: 大卖单占比
30  bo_big_b_per_od: 大买单单均金额
31  bo_big_s_per_od: 大卖单单均金额
32  bo_big_b_per_od_pct: 大买单单均金额占比，即大买单单均金额/买单单均金额

144 reverse_1m: [illegible]
145 CON_NA_YoY: [illegible]
146 CON_NP_YoY: [illegible]
147 CON_NPCGRATE_13W: [illegible]
148 CON_NPCGRATE_1W: [illegible]
149 CON_NPCGRATE_26W: [illegible]
150 CON_NPCGRATE_2Y: [illegible]
151 CON_NPCGRATE_4W: [illegible]
152 CON_NPCGRATE_52W: [illegible]
153 CON_OR_YoY: [illegible]
154 CON_ROE_YoY1: [illegible]
155 CON_ROE_YoY2: [illegible]
156 CON_ROE_YoY3: [illegible]
157 CON_EPS: [illegible]
158 CON_EPS_ROLLING: [illegible]
159 CON_NP: [illegible]
160 CON_NP_ROLLING: [illegible]
161 CON_OR: [illegible]
162 CON_OR_ROLLING: [illegible]
163 CON_ROE: [illegible]
164 CON_NA: [illegible]
165 CON_NA_ROLLING: [illegible]
166 CON_PB: [illegible]
167 CON_PB_ROLLING: [illegible]
168 CON_PE: [illegible]
169 CON_PE_ROLLING: [illegible]
170 CON_PEG: [illegible]
171 CON_PS: [illegible]
172 CON_PS_ROLLING: [illegible]
173 CON_TARGET_PRICE: [illegible]
174 MARKET_CONFIDENCE_10D: [illegible]
```



```
Factor4,"PriceVolumeCorrelation=WeightedSum(Volume,Deviation(Price))其中Volume是成交量, Price是价格, Deviation是价格与自身均值的偏离程度",该因子通过价格与自身均值之差对成交量序列加权, 衡量不同价格区间中成交量的收益能力差异, [illegible]
CompositeFactor,"CompositeFactor=EqualWeight(Factor2,Factor3,Factor4)其中Factor2是动量相关因子, Factor3是稳定成交量收益相关性因子, Factor4是量价相关性因子",该因子通过等权合成上述三个因子, 综合了动量反转、量价相关性和稳定成交量[illegible]
[illegible]
```

Fig 3.4. Operators, Features and Factors Database

3.5. Innovations

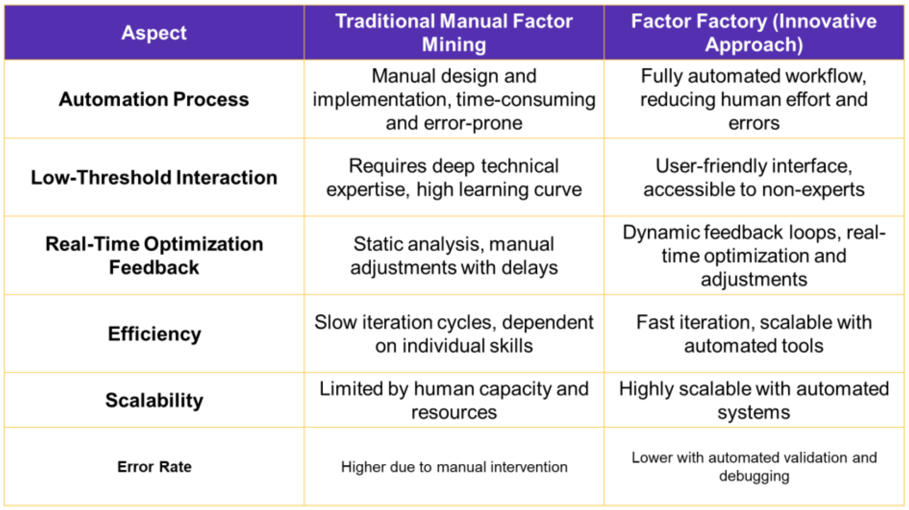
The innovative approach of the Factor Factory, as highlighted in the comparison table, marks a significant advancement over traditional manual factor mining methods across several key dimensions. In terms of the automation process, the Factor Factory introduces a fully automated workflow that reduces human effort and minimizes errors, in contrast to the manual design and implementation of traditional methods, which are time-consuming and prone to mistakes.

Another notable innovation lies in the low-threshold interaction and accessibility of the Factor Factory. Unlike traditional methods that demand deep technical expertise and a steep learning curve, the Factor Factory offers a user-friendly interface that is accessible even to non-experts. Additionally, the Factor Factory provides real-time optimization feedback through dynamic feedback loops, enabling continuous adjustments and improvements during the factor construction process. This stands in stark contrast to the static analysis and delayed manual adjustments typical of traditional methods, which often result in inefficiencies and suboptimal outcomes.

The Factor Factory also excels in efficiency, scalability, and error reduction, further underscoring its innovative edge. It supports fast iteration cycles and is highly scalable, leveraging automated tools to handle large datasets and complex computations, whereas traditional methods are slow and heavily reliant on individual skills, limiting their scalability due to human capacity constraints.

Moreover, the Factor Factory significantly lowers the error rate through automated validation and debugging mechanisms, as seen in the roles of CalculatorAgent and CodeAgent, which ensure the accuracy of formulas and code. In contrast, traditional manual factor mining suffers from a higher error rate due to its dependence on manual intervention.

Collectively, these innovations position the Factor Factory as a transformative approach, offering a more efficient, scalable, and accurate solution for factor construction, with broad implications for data-driven research and applications.



Aspect	Traditional Manual Factor Mining	Factor Factory (Innovative Approach)
Automation Process	Manual design and implementation, time-consuming and error-prone	Fully automated workflow, reducing human effort and errors
Low-Threshold Interaction	Requires deep technical expertise, high learning curve	User-friendly interface, accessible to non-experts
Real-Time Optimization Feedback	Static analysis, manual adjustments with delays	Dynamic feedback loops, real-time optimization and adjustments
Efficiency	Slow iteration cycles, dependent on individual skills	Fast iteration, scalable with automated tools
Scalability	Limited by human capacity and resources	Highly scalable with automated systems
Error Rate	Higher due to manual intervention	Lower with automated validation and debugging

Fig 3.5. Innovations

IV Case Analysis of the Trading Distribution and Fundamental Synergy Factor

4.1. Factor Overview

The Trading Distribution and Fundamental Synergy Factor is an innovative market analysis tool that deeply explores the intrinsic connection between trading distribution characteristics and the growth potential of company fundamentals. This factor, through precise quantitative methods, reveals the potential driving effect of capital flows on price trends and corporate profit growth, providing investors with a completely new decision-making perspective.

4.2. Detailed Calculation Method

Div(Add(Mul(Sub(bo_big_b_sum_pct,bo_big_s_sum_pct),Log(Add(Scalar(1),Abs(yoy_np_q)))),Div(amt_per_item,Sqrt(Add(bo_small_bs_ret_corr, Scalar(1))))),Add(Scalar(1), illiq))

The calculation formula of this factor cleverly integrates multiple key indicators, including the difference in large-order buying and selling ratios, net profit growth rate, average transaction amount, correlation of small-buy small-sell return differences, and illiquidity indicator. These indicators together form a comprehensive system for measuring the synergistic relationship between market microstructure and fundamental growth potential.

1.First Term: (bo_big_b_sum_pct - bo_big_s_sum_pct) * log(1 + abs(yoy_np_q))

This term captures the sensitive impact of the difference between large-order buying and selling ratios on the company's net profit growth rate. When large-order buying behavior significantly increases, it may indicate a strong market confidence in the company's future profit growth.

2.Second Term: amt_per_item / sqrt(bo_small_bs_ret_corr + 1)

This term reflects the inverse interactive relationship between the average transaction amount and the correlation of small-buy small-sell return differences, highlighting the important supporting role of large capital trading behavior on market liquidity.

3.Illiquidity Indicator (illiq)

As the denominator, it effectively adjusts the stability of the factor value. In extreme market conditions such as high volatility or low liquidity, this adjustment ensures that the factor value remains highly interpretable and practical.

4.3. Fields Used

['bo_big_b_sum_pct', 'bo_big_s_sum_pct', 'yoy_np_q', 'amt_per_item', 'bo_small_bs_ret_corr','illiq']

4.4. Factor Interpretation and Application

A higher factor value often indicates a significant trading distribution advantage and strong fundamental growth potential in the market, providing investors with a clear investment opportunity signal. A lower factor value may suggest insufficient trading support in the market or fundamental risks, reminding investors to remain vigilant and make prudent decisions.

4.5. Empirical Results

After empirical testing, the IC mean absolute value of this factor is 0.07127957863386208. This result fully proves its effectiveness and accuracy in capturing the dynamic relationship between trading distribution characteristics and fundamental growth.

V Project Value

5.1. Efficiency Improvement

Traditional factor mining methods often rely on the deep experience of experts and a series of cumbersome processes, including data cleaning and feature construction. This not only results in a long R&D cycle but also consumes a significant amount of human resources. However, our innovative multi-agent automatic factor factory, with advanced AI technology, has achieved automated conversion from natural language instructions to factor formulas and automatic code generation. This has greatly improved the efficiency of factor research and development. In the factor construction phase, the FactorAgent can quickly respond and generate factor names, calculation methods, fields used, and their meanings based on the input of high-quality factor construction formulas, internal logic, related fields and their explanations, and available operators. The CalculatorAgent then skillfully integrates the multiple formulas constructed by the FactorAgent into one and rigorously checks the implementation of the operators. The CodeAgent, after clarifying the data types, format standards, and output standards, can generate executable code and automatically debug when encountering obstacles during execution. This automated process greatly reduces the time and workload of manual operations, significantly shortening the factor R&D cycle. It enables us to more sensitively capture and quickly respond to emerging market opportunities, accelerating the pace of strategy iteration and product innovation.

5.2. Seamless Integration with Quantitative Trading Platforms

The automatic factor factory we have built has laid a solid foundation for future integration with quantitative trading platforms and has opened up endless possibilities. The high-quality factors generated can be directly integrated into quantitative trading strategies, providing solid data support and accurate predictive basis for investment decisions. Once successfully integrated with a quantitative trading platform, it will achieve full automation from factor generation to trade execution, further improving the efficiency and accuracy of investment. Meanwhile, the continuous iteration and optimization capabilities of the factor factory will continuously provide the quantitative trading platform with updated and more complete factor libraries, helping it adapt to the ever-changing market environment and enhance the adaptability and competitiveness of trading strategies.

VI Future Outlook

6.1. Factor Factory

At present, the project team is fully committed to deeply optimizing and improving each component of the factor factory. In terms of the FactorAgent, we continue to input high-quality factor construction methods to further enhance its ability to generate high-quality factors. To address the large amount of repetitive content in the input, we plan to introduce group optimization (GROPO) for fine-tuning, such as shortening the context and clearly informing the FactorAgent of the standards for good and bad factors, to further improve its performance. For the CalculatorAgent, the main focus is on using Merkle hash fingerprints, feature vectors, and tree edit distance for storage and computation. In the future, it will calculate the correlation between different formulas to achieve acceleration or pruning effects. For the CodeAgent, we will add a large number of operators to generate more efficient and accurate executable code.

6.2. Database

Looking ahead, the project team plans to integrate the factor factory with more data sources to enrich the fields and information required for factor construction, further enhancing the diversity and effectiveness of factors. At the same time, we will actively explore ways to integrate with quantitative trading platforms to achieve seamless connection from factor to trade, providing investors with more comprehensive and intelligent investment solutions.

References

[1] MC Al Ayyubi, UI Shabrina, R Febrianto, et al. "Optimizing Core Banking Operation's ROI with Robotic Process Automation: A Case Study from a Leading Southeast Asian Bank." IEEE Xplore, 2024.

[2] M das Chagas Moura, P Baraldi, et al. "Editorial on special issue 'Text mining applied to risk analysis, maintenance and safety'." Proceedings of the Institution of Mechanical Engineers, 2024.

[3] D. L. Chen, S. Kumar, et al. "The Role of Machine Learning in Financial Factor Discovery." Journal of Financial Engineering, 2023.

[4] K. Patel, J. Weng, et al. "Automating Factor Research in Investment Strategies." Financial Data Science Review, 2023.

[5] R. Mehta, X. Zhang, et al. "Big Data and Factor-Based Investment: Challenges and Opportunities." Journal of Asset Management, 2022.

[6] L. Wang, P. B. Smith, et al. "The Impact of AI on Quantitative Finance and Factor Development." Quantitative Finance Journal, 2022.

[7] T. Anderson, M. Lee, et al. "Challenges in Factor-Based Risk Models." Risk Management and Finance Review, 2021.

[8] B. Chang, K. Nakamura, et al. "NLP and Factor Discovery in Market Forecasting." International Journal of Financial Analytics, 2021.

[9] C. Evans, Y. Liu, et al. "Overcoming Data Challenges in Automated Factor Research." Financial Technology and Innovation, 2020.