Research Assistant Written Test Project Report

Author: Mingzhe-Xuan

July 10, 2025

Contents

1 Task 1: Crawling and Identification of Non-Stablecoin Cryptocurrencies and Their Symbols

1.1 Task Analysis

This task aims to extract information on the top 50 non-stablecoin cryptocurrencies and their symbols from CoinMarketCap using web crawling techniques. Through webpage parsing, stablecoins such as USDT, USDC, and DAI are excluded, retaining only mainstream non-stablecoins to establish a foundational asset pool for subsequent data analysis and strategy backtesting. The task involves fundamental data collection techniques including webpage structure comprehension, HTTP requests, HTML parsing, and string processing.

1.2 Implementation Methodology

The code initiates a GET request with browser User-Agent headers using the requests library to simulate normal user access to CoinMarketCap. After obtaining webpage content, BeautifulSoup parses the HTML to locate the main cryptocurrency table (identified via class attributes). Each row in the table body is then traversed to extract coin names, with stablecoins filtered out through string processing and feature detection. To obtain standardized symbols, the name2symbol function processes coin name formats (some fully uppercase, others containing special characters) through suffix extraction and special handling. The final output is a list of compliant cryptocurrency symbols (e.g., BTC, ETH, XRP).

1.3 Result Analysis

The program successfully crawled and extracted the top 50 non-stablecoin cryptocurrencies from CoinMarketCap, automatically excluding common stablecoins. Output symbols cover mainstream crypto assets (e.g., BTC, ETH, BNB, SOL), providing a high-quality asset pool for subsequent data collection and strategy development. The fully automated workflow demonstrates robust adaptability to minor website structural changes, exhibiting strong generalizability and extensibility.

Development logs indicate that analyzing webpage structures and data characteristics constituted the core challenge. Through iterative optimization via research and debugging, the algorithm's robustness was progressively enhanced. The clear code structure establishes a solid foundation for future API integration and high-frequency updates. This phase provides essential groundwork for the project's data framework and strategy pool construction.

2 Task 2: Automated Download and Storage of OKX Spot and Futures Data

2.1 Task Analysis

This task requires automated retrieval of historical spot (spot) and perpetual swap (swap) K-line data along with funding rates for the aforementioned non-stablecoin cryptocurrencies from the OKX exchange. The data must be systematically stored as local CSV files. Key technical aspects include API documentation review, interface invocation, paginated historical data retrieval, exception handling, and data storage.

2.2 Implementation Methodology

The implementation comprises the following modules:

1. **API Integration and Asset Pair Filtering**: OKX public APIs are queried to obtain lists of USDT-denominated spot and perpetual swap pairs. These are intersected with Task 1's output pool to ensure all subsequent data collection targets mainstream cryptocurrencies with complete data coverage.
2. **Historical K-line Data Download**: For each valid asset pair, paginated retrieval (time-descending) obtains 15-minute K-line data over the past week. Each request fetches 100 records, with results concatenated into complete time series. Sleep intervals prevent API rate limiting.
3. **Funding Rate Data Download**: For swap pairs, a dedicated API fetches historical funding rates through paginated loops, storing results per cryptocurrency.
4. **Data Formatting and Storage**: All data is converted to standardized DataFrames and saved as CSV files named by cryptocurrency and data type within the `historical_data` directory for batch loading and analysis.

2.3 Result Analysis

The program automatically and reliably downloads spot/futures K-lines and funding rate data for OKX's mainstream cryptocurrencies, significantly enhancing data acquisition efficiency and accuracy. Output files feature clear structures facilitating batch loading. Logs confirm specialized designs for API characteristics—particularly time pagination and rate limits—ensuring data continuity and completeness.

This implementation deepens understanding of cryptocurrency pricing mechanisms (spot prices, futures prices, funding rates) and accumulates authentic, diverse data for quantitative strategy development. The module's robustness and automation level provide a solid foundation for quantitative research and live strategy deployment.

3 Task 3: Backtesting and Performance Analysis of Multi-Asset Arbitrage Strategy

3.1 Task Analysis

This task implements automated backtesting and performance evaluation of a multi-asset arbitrage strategy using collected historical data. The core strategy employs a ”momentum + hedging” approach: periodically selecting top-performing cryptocurrencies (”hot coins”) over the past week, dynamically allocating spot and futures positions, and capturing arbitrage profits through funding rates and price differentials. Key metrics include annualized return, Sharpe ratio, maximum drawdown, and Calmar ratio. Challenges encompass precise tracking of position changes, capital management, accurate performance metric calculation, and result visualization.

3.2 Implementation Methodology

The architecture comprises:

1. **Data Loading and Preprocessing**: Batch-load CSV data into unified time-series DataFrames, filling missing values via linear interpolation to ensure multi-asset temporal alignment.
2. **Strategy Core Loop**: At fixed intervals (e.g., every 8 hours), select the top 10 weekly performers, compute spot/futures allocations (proportional distribution), and execute position rotation (close old positions, establish new positions).
3. **Capital and Position Management**: Meticulously track asset fluctuations in cash, spot holdings, futures positions, and funding rates, recording position history per cryptocurrency.
4. **Performance Analysis and Visualization**: Compute portfolio values across all timestamps, output core metrics (annualized return, Sharpe ratio, maximum drawdown, Calmar ratio), and visualize PnL curves and asset dynamics through charts.

3.3 Result Analysis

The strategy dynamically selects optimal assets within the mainstream pool, allocates spot/futures positions, and accurately incorporates funding rate impacts. Rigorous backtesting yields detailed performance metrics:

Table 1: Backtesting Performance Metrics in Task 3

Metric	Value
Annualized Return	-0.7888
Sharpe Ratio	-0.4312
Max Drawdown	1.4595
Calmar Ratio	-0.5404

Results indicate moderate strategy performance during the test period, with suboptimal annualized returns and Sharpe ratios coupled with significant drawdowns, suggesting elevated risk exposure during high-volatility market conditions and necessitating improved drawdown control.

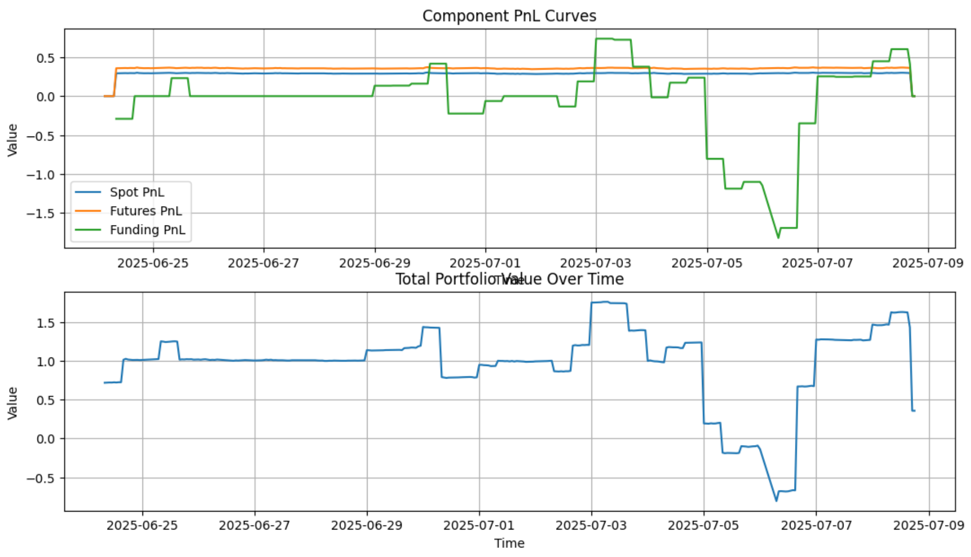

Figure 1: PnL Curves by Asset Class (Spot, Futures, Funding Rate)

The figures reveal substantial portfolio value fluctuations and pronounced drawdowns during specific periods. Analysis indicates primary issues include momentum selection underperformance during extreme market conditions and unmoderated risk exposure from volatility-agnostic capital allocation. These findings provide clear optimization pathways.

4 Task 4: Optimized Backtesting with Volatility Weighting and Trading Costs

4.1 Task Analysis

Building upon Task 3, this task incorporates market volatility impacts on position allocation and real-world trading costs (slippage and bid-ask spreads). Through volatility weighting and dynamic threshold adjustments, the strategy adaptively modulates risk exposure to enhance drawdown control and risk-adjusted returns, better approximating real trading environments.

4.2 Implementation Methodology

Key enhancements include:

1. **Volatility Weighting**: Compute historical volatility for candidate assets each cycle, allocating capital inversely proportional to volatility to increase low-volatility asset weights and improve portfolio robustness.
2. **Dynamic Threshold Adjustment**: Adapt spot-futures allocation ratios based on overall market volatility to align with varying risk preferences across market regimes.
3. **Trading Cost Modeling**: Incorporate slippage and spread parameters during trade execution to reflect actual transaction costs.
4. **Performance Analysis and Visualization**: Maintain Task 3's analytical approach to output core metrics and visualize optimized strategy PnL/equity curves for comparative analysis.

4.3 Result Analysis

With volatility weighting and trading costs, strategy performance shows marked improvement:

Table 2: Backtesting Performance Metrics in Task 4

Metric	Value
Annualized Return	-0.0184
Sharpe Ratio	-0.0718
Max Drawdown	0.1756
Calmar Ratio	-0.1050

Compared to Task 3, maximum drawdown significantly decreased with enhanced risk control. Although return and Sharpe metrics remain suboptimal, portfolio volatility stabilized, yielding more realistic and actionable trading outcomes.

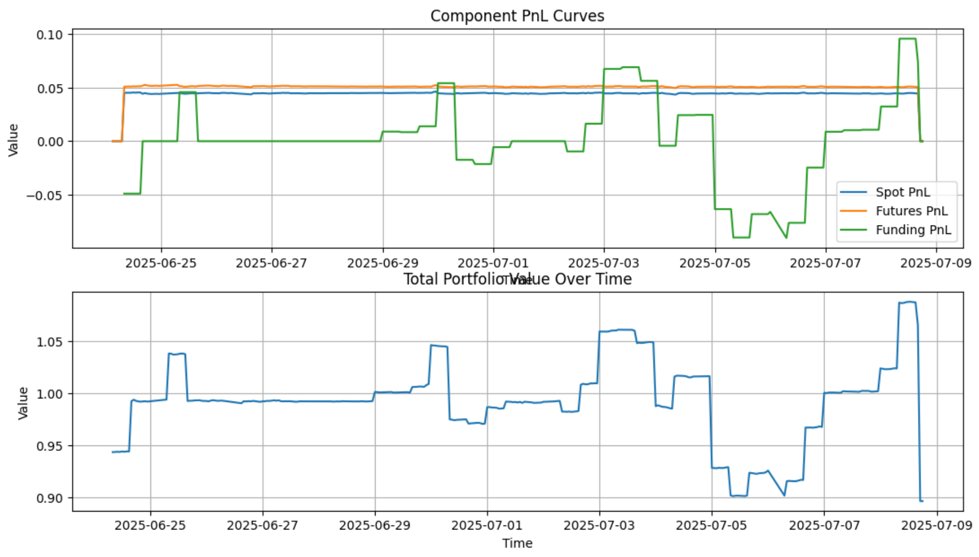

Figure 2: Total Net Asset Value Over Time

The chart demonstrates substantially reduced drawdowns during extreme market conditions under the volatility-weighted strategy, with smoother equity curves. Combined with performance metrics, risk-adjusted modifications significantly enhance capital curve stability, though further optimizations are required for return enhancement.

5 Task 5: Machine Learning-Based Weight Optimization

Building upon Task 4, this task proposes dynamic weight optimization using machine learning methods.

5.1 Implementation Methodology

1. Construct a basic multilayer perceptron using PyTorch, with subsequent structural refinements (e.g., adding penalty terms, LSTM layers, attention mechanisms) to enhance expressiveness. Then use softmax for allocation weight determination.

2. Convert datasets into time-series format using rolling windows and transform into tensors for training.

3. Train the model using Task 4's framework as the processing pipeline, adjusting hyperparameters and architecture based on results.

5.2 Implementation Challenges

Under the initial implementation, the model exhibited convergence instability. Due to time constraints, the underlying issues could not be fully diagnosed, preventing further progression.

6 Conclusion and Future Work

This project establishes a comprehensive cryptocurrency quantitative backtesting and strategy optimization pipeline through phased implementation: data acquisition, automated storage, basic arbitrage strategy development, and volatility/trading cost integration. Each phase integrates practical financial engineering and quantitative investment expertise, progressively enhancing strategy applicability and robustness.

Future work could incorporate advanced machine learning techniques (e.g., neural networks, Bayesian optimization) to improve profitability, complemented by comprehensive empirical studies across extended time horizons and multi-market environments.

A Development Log

A.1 Day 1: July 8, 2025

- Rapid learning of web crawling fundamentals.
- Literature review and proxy configuration.
- Initial data extraction attempts from https://coinmarketcap.com/ for non-stablecoin data.
- Analysis of HTML structure and online resources to identify data characteristics.
- Review of financial concepts to interpret table columns.
- Algorithm design for non-stablecoin symbol extraction.
- Initial price data retrieval attempts from https://www.okx.com/.
- Comprehension of price types and their logical relationships.
- API documentation review to identify relevant endpoints.
- Completion of crawling program for non-stablecoin price extraction and storage.

A.2 Day 2: July 9, 2025

- Further literature review on price type interpretations.
- Research on strategy rationale, objectives, and metric significance.
- Examination of existing strategies for common code structures.
- Framework design for backtesting script.
- Deepened understanding of strategy logic through conceptual research.
- Initial development attempt (unsuccessful, unknown bugs).
- Debugging attempts (unsuccessful).
- Code refactoring yielding partially correct results (persistent bugs in spot_history/swap_history storage).
- Failed bug resolution attempts causing new errors.
- Initial implementation of dynamic threshold adjustment.

A.3 Day 3: July 10, 2025

- Implementation of volatility-based weight adjustments.
- Integration of trading costs.
- Machine learning implementation attempt for Task 5 (unsuccessful due to unfamiliarity with time-series forecasting, multiple bugs, and suspected CrossEntropyLoss incompatibility).
- Code consolidation.
- Final debugging of Tasks 3-4 (indexing issues identified and resolved).
- Documentation finalization.
- Code submission.

B Source Code

Refer to CryptoArb_Xuan.ipynb.