Advanced ICT Project 2

Technical Report

Tech No.	SPEIT-ICE7902P-001
Course Number	ICE7902P
Subject	Information Engineering
Author(s)	Maria STIVALA, Hazael SOLEDADE, Léo BONNEVILLE, Théo COUERBE
Document Title	AI-Agent for Financial Time Series Data

Contents

1 Introduction

Financial time series analysis is a challenging engineering problem due to the non-stationary nature of market data, the presence of noise, and the disproportionate importance of rare but high-impact events such as market crashes or flash crashes [1]. Unlike many engineered signals, financial markets exhibit abrupt regime changes, volatility clustering, and structural breaks, which complicate modelling and inference. Traditional quantitative analysis tools often require significant programming expertise [2], creating a barrier for domain experts such as traders and economists. Moreover, real-world OHLCV data feeds frequently contain anomalies—including duplicate timestamps, missing observations, and logically invalid price combinations—that are rarely handled automatically by standard analysis pipelines.

Figure 1 provides an illustrative long-run perspective on equity market volatility, highlighting the clustered and regime-dependent behaviour of large price movements. Extended periods of relatively stable dynamics are punctuated by bursts of extreme volatility around major macroeconomic, financial, or geopolitical events. This behaviour, widely documented in the financial literature as a stylized fact of asset returns, underscores the need for robust preprocessing and analysis pipelines: models calibrated on "normal" market conditions often fail precisely when reliability is most critical.

Recent work has explored the use of large language models (LLMs) for financial analysis and decision support [3]. However, many proposed systems either delegate numerical computation directly to language models, thus risking hallucination, non-reproducibility, and lack of auditability, or rely on rigid, non-interactive pipelines that limit adaptability and user interaction. As a result, there remains a gap between flexible, language-driven interfaces and the strict reproducibility requirements of quantitative financial analysis.

To address this gap, this project presents *a modular AI agent for OHLCV time series analysis*. The proposed system combines the adaptability of local LLMs, via Ollama, for natural language interaction and analytical planning with fully deterministic, auditable computation for data enhancement and technical indicator calculation [4]. The agent supports both batch execution and an interactive chat interface, while guaranteeing that all numerical results are reproducible, cacheable, and isolated from LLM-induced errors.

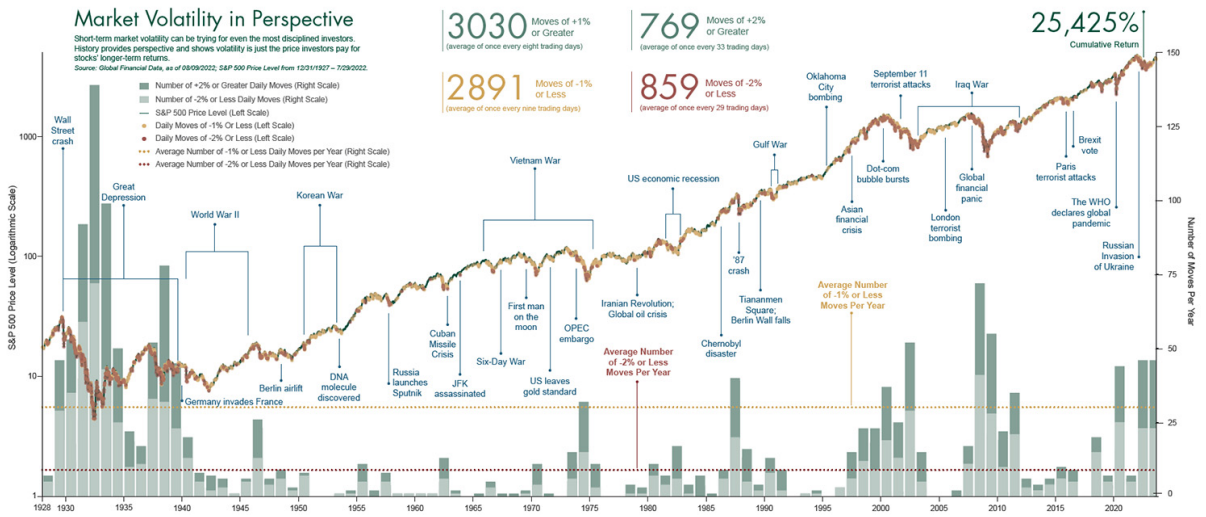

Figure 1: Illustrative long-run perspective on equity market volatility (S&P 500) highlighting clustered volatility and crisis regimes. Source: Fisher Investments.

2 Main Objective

This project was conceived with the goal of building a financial analysis system that overcomes two key limitations of existing approaches: (1) the fragility of LLM-only pipelines [3], and (2) the rigidity of traditional quantitative tools [2].

As a result, the primary objective of this paper is to develop a transparent and reproducible AI agent for OHLCV time series analysis that strategically confines Large Language Models (LLMs) to language-oriented tasks, while ensuring all numerical computation remains deterministic.

Specifically, the system is designed to use LLMs only for:

1. **User intent parsing**: Where the LLM extracts symbols and timeframes from natural language

2. **Analytical planning**: Where the LLM selects data enhancement methods and indicators based on data profiles

3. **Report generation**: Where the LLM produces human-readable narratives from structured results.

All other components, being data validation, enhancement, indicators, and caching, must be implemented as auditable, deterministic functions, supporting both batch and interactive usage without compromising reproducibility.

3 Organizational Implementation

The project is implemented using a modular software architecture, where each subsystem is isolated into clearly defined components with standardized input/output interfaces. This design naturally supports collaborative development by minimizing dependencies between modules.

Work was organised around clear architectural boundaries, beginning with the user interfaces, being a command-line interface and an LLM-powered chat agent, followed by data ingestion and the data pipeline, where merging and profiling are performed. This is followed by LLM-based planning, deterministic tool execution for data enhancement and indicator computation, state management, and finally an LLM-based reporting component. This structure enabled focused development of individual layers while ensuring consistent integration through shared data formats, using CSV files for time series data and JSON files for profiles, plans, and results.

4 System Workflow

The system executes a linear, auditable pipeline that transforms raw user input into a structured financial report. Although initiated from two different interfaces, the core workflow is identical once the analysis parameters are defined:

1. **User input**: This is provided either via command-line arguments to `main.py` in batch mode or as natural language in the Streamlit chat interface defined and implemented in `chat_app.py`. In the chat mode, an LLM extracts the variables `symbol`, `timeframe`, and `recent_limit` from the user message.

2. **Configuration loading**: The global settings, such as data directory and Ollama endpoints, are loaded from `config.py`.

3. **Logging initialization**: Unified debug/info logging is configured.

4. **State loading**: Cached results from previous runs stored in any `state_*.json` are loaded, to enable result reusability.

5. **Historical data loading**: Local OHLCV data is read from `historical_*.csv`, however, an empty file is created if missing.

6. **Recent data fetching**: Live candles are retrieved from Binance API, however it fallbacks to empty if it fails.

7. **Data merging**: Historical and recent data are combined into a continuous time series, as defined and implemented in `merged_*.csv`.

8. **Data profiling**: A structured data quality profile is generated and saved as `profile_*.json`. This JSON contains:

    - *Metadata*: symbol, timeframe, row count, and column names
    - *Missing data diagnostics*: count and timestamps of missing OHLCV fields
    - *Duplicate analysis*: number of duplicate groups and their timestamps
    - *OHLC validity checks*: count and reasons for invalid price sequences, such as `low_gt_min(open,close)`
    - *Outlier detection*: volume anomalies identified via robust Z-score, with timestamps.

    Critically, this profile contains only summaries and diagnostics, and never raw price or volume values.

9. **Tool planning**: the LLM receives only the above data quality profile, and never the raw OHLCV series. Based exclusively on these diagnostic summaries, it selects which enhancement methods, such as either deduplication, or even gap filling, and technical indicators, such as RSI or MACD, to apply, producing `planner_*.json`.

10. **Tool execution**: The system executes tools in the order defined by the planner. First, all enhancement methods are applied sequentially to the OHLCV data. The resulting cleaned time series is immediately saved as `enhanced_data_*.csv`. Then, each technical indicator is computed on this cleaned data, with results temporarily held in memory.

11. **Persistence**: All final outputs are written to disk:

    - Indicator results → `indicators_*.json`
    - Updated execution state → `state_*.json`

    The state includes cached indicator results to enable incremental updates on subsequent runs.

12. **Report generation**: Results are formatted into a Markdown report, as defined and implemented in `reporter.py`, and saved in the directory under `reports/`.

This unified pipeline ensures consistent behaviour, full reproducibility, and complete traceability regardless of the entry point.

5 Technical Specifications

The system implements a collection of deterministic data enhancement and technical indicator methods tailored to financial OHLCV time series. All computations are fully auditable, reproducible, and deliberately free from stochastic elements, ensuring consistent behaviour across repeated executions.

5.1 Data Enhancement Methods

Financial OHLCV time series data frequently contains structural irregularities that must be addressed before reliable analysis can be performed. Such anomalies arise from data acquisition processes, exchange outages, and inconsistencies introduced during data merging. The enhancement methods implemented in this system are designed to ensure data integrity, robustness, and reproducibility prior to indicator computation.

Duplicate Timestamps

Duplicate timestamps occur when multiple observations share the same time index, commonly as a result of API inconsistencies or errors during data aggregation. If not corrected, duplicates violate the assumptions of time series models and distort indicator calculations, including rolling averages and volatility measures.

The financial data preprocessing literature identifies deduplication as a fundamental cleaning step, as most analytical methods assume a strictly ordered and unique time index [2]. Common strategies include retaining a single observation, aggregating values, or discarding redundant entries.

Our implementation: The system removes duplicate rows with identical timestamps by retaining the final occurrence by default, with an option to retain the first instead. Targeted deduplication over specific date ranges is supported, and the number of removed rows is recorded for auditability.

Missing Values

Missing OHLCV values may result from exchange downtime, transmission errors, or incomplete historical records. Such gaps disrupt rolling-window calculations and compromise the validity of volatility and momentum estimates.

Imputation methods are widely used in financial time series analysis, with forward-fill and backward-fill approaches commonly adopted due to their simplicity and ability to preserve temporal continuity without introducing artificial trends [5]. While more advanced techniques such as Kalman filtering or expectation–maximisation can be applied, these methods introduce additional stochastic assumptions.

Our implementation: Missing values are filled using forward propagation of the most recent valid observation, followed by backward filling where necessary. The filling process

can be restricted to specific rows or limited in length. Volume data is treated consistently with price fields to maintain coherence across OHLCV dimensions.

Invalid OHLC Sequences

Invalid OHLC sequences arise when fundamental price constraints are violated, for example when the recorded high price is lower than the open or close price, or when the low price exceeds other values. These inconsistencies typically stem from corrupted feeds or misaligned merges.

Ensuring logical consistency in OHLC data is emphasised in financial preprocessing research, as violations can lead to misleading indicator values and incorrect interpretation of candlestick-based signals [6].

Our implementation: Logical violations are corrected using a clipping strategy in which the high price is adjusted to the maximum of the open and close prices, and the low price is adjusted to the corresponding minimum. Where required, invalid rows may alternatively be removed entirely.

Extreme Outliers

Extreme outliers in price or volume are often caused by erroneous ticks, reporting errors, or brief periods of illiquidity. These values disproportionately affect volatility estimates and may generate spurious trading signals.

Robust statistical methods recommend the use of the Median Absolute Deviation (MAD) for outlier detection, as it is substantially less sensitive to extreme values than standard deviation-based approaches [7]. In financial applications, both MAD-based thresholds and percentile-based clipping are commonly employed.

Our implementation: Extreme values in selected fields, typically closing price and volume, are capped using MAD-based Z-scores with a default threshold of 6.0. As an alternative, values may be clipped to the 1st and 99th percentiles, depending on configuration.

5.2 Technical Indicators

Technical indicators are mathematical transformations applied to OHLCV data in order to extract information about market trends, volatility, and momentum. They are widely employed in both academic research and practical trading systems. This section describes the indicators implemented in the system, outlines their theoretical foundations, and explains their deterministic computation.

Total Return

Total return measures the cumulative change in price over the entire observation period and is defined as $P_{\text{end}}/P_{\text{start}} - 1$, where P denotes the closing price. It provides a baseline

assessment of overall performance and is a standard metric in portfolio evaluation and financial analysis [8].

Our implementation: The system computes cumulative price return over the full dataset using closing prices.

Relative Strength Index (RSI)

The Relative Strength Index (RSI), introduced by Wilder [9], is a momentum oscillator that measures the magnitude of recent price changes in order to identify overbought or oversold conditions. The indicator ranges from 0 to 100, with values above 70 typically interpreted as overbought and values below 30 as oversold. RSI is widely used in technical analysis and algorithmic trading research.

Our implementation: RSI is computed using average gains and losses over a 14-period window, producing values within the standard 0 to 100 range.

Annualized Volatility

Volatility quantifies the variability of asset returns and plays a central role in risk assessment. Annualised volatility is calculated as the standard deviation of daily log returns, scaled by $\sqrt{252}$, which approximates the number of trading days in a year. This measure is standard in quantitative finance [10].

Our implementation: The system computes daily log returns, evaluates their standard deviation, and applies annualisation using the square-root-of-time rule.

Maximum Drawdown

Maximum drawdown measures the largest peak-to-trough decline in cumulative returns over the observation period and is expressed as a percentage. It captures downside risk beyond what is reflected by volatility alone and is widely used in performance and risk analysis [11].

Our implementation: Cumulative returns are tracked over time, peak-to-trough declines are identified, and the largest observed drawdown is reported.

MACD (12,26,9)

The Moving Average Convergence Divergence (MACD) indicator, introduced by Appel [12], is a trend-following momentum measure based on the difference between a short-term and a long-term exponential moving average. Specifically, it uses 12-period and 26-period EMAs, together with a 9-period EMA signal line. MACD is commonly used to identify trend reversals and shifts in momentum.

Our implementation: The system computes the 12-period and 26-period EMAs of closing prices, calculates their difference, and applies a 9-period EMA as the signal line.

Bollinger Bands (20,2)

Bollinger Bands, introduced by Bollinger [13], are volatility bands constructed around a simple moving average. They consist of a 20-period simple moving average with upper and lower bands set at plus and minus two standard deviations of price. These bands provide a dynamic representation of volatility and potential breakout regions.

Our implementation: A 20-period simple moving average is computed, and upper and lower bands are added at two standard deviations above and below the average.

Average True Range (ATR)

The ATR, introduced by Wilder [9], measures market volatility as the average of the true range over a 14-day period. True range is defined as $\max(H - L, |H - C_{\text{prev}}|, |L - C_{\text{prev}}|)$, capturing intraday and overnight volatility.

Our implementation: True range is computed for each period, followed by a 14-period simple moving average to obtain ATR values.

Volume Analysis

Volume analysis assesses unusual trading activity by comparing current trading volume with historical norms. A common approach involves computing the Z-score of volume relative to a rolling mean and standard deviation, allowing abnormal spikes in activity to be identified [14]. Such spikes are often associated with heightened market interest or forthcoming price movements.

Our implementation: The system computes a 20-period rolling mean and standard deviation of trading volume, calculates corresponding Z-scores, and flags periods of unusually high or low activity.

6 Technical Challenges and Engineering Decisions

6.1 Handling Noisy and Inconsistent Financial Data

Challenge. Real-world OHLCV time series data, obtained from local CSV archives and the Binance API, frequently exhibited structural inconsistencies. These included duplicated timestamps, missing observations, logically invalid price sequences, such as cases where the recorded high price was lower than the open price, and extreme spikes in trading volume [5, 7]. Such anomalies violate the assumptions underlying standard technical indicators and compromise the validity of derived measures [9, 12].

Initial approaches. Early prototypes relied on manual inspection and generic preprocessing operations, such as dropping missing values or applying linear interpolation. These approaches proved insufficient, as they were neither auditable nor capable of adapting to the specific characteristics of different datasets.

Final design. A deterministic, diagnosis-driven data enhancement pipeline was implemented. Four specialised tools, namely `deduplicate_by_timestamp`, `fill_missing`, `fix_bad_ohlc`, and `winsorize_outliers`, are executed only when a preceding data profiling step explicitly identifies their corresponding anomalies. This design ensures that corrections are minimal, traceable, and fully reproducible.

6.2 Preventing LLM Exposure to Sensitive Market Data

Challenge. Providing raw OHLCV time series directly to the LLM introduced several risks, including potential leakage of sensitive financial data, excessive token consumption, and the possibility of hallucinated or non-deterministic modifications to numerical values [3].

Initial approaches. Attempts were made to reduce exposure by truncating the time series or transmitting only the most recent candles. Although these measures reduced input size, they still involved sharing raw price data and offered no formal guarantees of correctness or reproducibility.

Final design. A strict data abstraction layer was introduced. The LLM receives only a compact, anonymised data quality profile in JSON format, containing metadata such as the asset symbol and timeframe, together with diagnostic summaries, for example the number of duplicate timestamp groups or the presence of missing fields at specific timestamps. No raw price or volume values are included, thereby fully decoupling language-based reasoning from numerical market data [15].

6.3 Avoiding Redundant Computation Across Pipeline Runs

Challenge. Repeated execution of the analysis pipeline on unchanged data resulted in full recomputation of enhancement steps, indicator calculations, and LLM interactions, leading to unnecessary computational overhead.

Initial approaches. Naïve solutions, such as unconditional re-execution or checking for the presence of output files, failed to detect incremental data updates and did not support fine-grained reuse of intermediate results.

Final design. A content-aware state caching mechanism was developed using versioned `state_*.json` files. Each tool output is indexed by the most recent timestamp present in its input data. During subsequent runs, the executor compares the current data timestamp with the cached key and reuses existing results when no change is detected, enabling efficient incremental execution [16].

6.4 Ensuring Robustness with a Local, Unstable LLM (Ollama)

Challenge. Operating a local LLM through Ollama occasionally resulted in malformed JSON responses, timeouts, or incomplete outputs, such as plans missing required fields [3]. These failures risked silent errors or pipeline interruption.

Initial approaches. Increasing timeout thresholds or switching to larger models provided marginal improvements in stability but significantly increased latency and did not address parsing failures.

Final design. Robustness was achieved through a combination of structured prompt engineering, strict JSON schema validation, retry mechanisms, and deterministic fallback behaviour. For example, if the reporting LLM fails, a rule-based template generates a summary using only precomputed indicator values, ensuring that a valid output is always produced [1].

6.5 Orchestrating Dynamically Selected Tools with Correct Dependencies

Challenge. The LLM-based planner is free to select arbitrary subsets of enhancement and indicator tools. However, correct execution requires that all data enhancement steps be completed before any indicator computation, an ordering constraint not enforced by the planner itself.

Initial approaches. Early implementations relied on a monolithic execution function with hard-coded phases, which proved inflexible and difficult to extend as new tools were added.

Final design. Responsibilities were separated through the introduction of a tool registry and a generic executor. The registry maintains metadata and callable references for each tool, while the executor enforces a two-stage workflow in which all selected enhancement methods are applied first, followed by computation of all selected indicators on the cleaned dataset. This approach supports arbitrary tool combinations while guaranteeing correctness and extensibility.

7 Methodology and Results

This section presents a self-contained methodology for forecasting short-term Bitcoin volatility and generating probabilistic price trajectories using only locally stored historical data. The entire analysis is performed on an hourly BTC/USDC price series read from a static CSV file, with no reliance on external APIs or real-time feeds.

From the closing prices in the file, we derive a set of financial features that capture recent market dynamics, including absolute returns over 1, 4, and 24 hours, rolling volatility computed as the standard deviation of hourly returns, the asset's relative position within the past 24-hour price range, and the MACD momentum indicator. The target variable is defined as the absolute value of the next-hour return, which serves as a direct measure of forward-looking volatility at the hourly horizon.

A Random Forest regressor is trained on the most recent 90 days of engineered features to predict this target. The model's internal logic is made transparent through two diagnostic visualizations shown in Figure 2. The left panel displays feature importance scores, confirming that the most predictive features are recent volatility over 4 hours and the immediate past return (return 1h), followed by the MACD momentum indicator and 24-hour volatility. This aligns with the financial stylized fact of volatility clustering, where recent price action strongly informs near-term uncertainty. The right panel presents a correlation heatmap between input features and the target, revealing strong autocorrelation in volatility and weaker but meaningful relationships with momentum and price position.

Rather than producing a static point forecast, the model's prediction seeds a stochastic 72-hour volatility trajectory. This is generated by simulating a mean-reverting process that occasionally incorporates upward jumps to reflect realistic market shocks, such as news events or liquidity crises. The resulting forecast is non-constant and dynamically responsive to inferred market regimes. Figure 3 shows this projection (red dashed line) alongside the last five days of observed volatility (blue line).

This time-varying volatility forecast then drives 50 Monte Carlo simulations of the Bitcoin price path over the next three days. Each path assumes zero drift and Gaussian returns scaled by the forecasted hourly volatility, reflecting the high uncertainty inherent in short-term crypto price movements. The ensemble of simulated trajectories, displayed in Figure 4, quantifies the range of plausible outcomes: while the mean path remains near the current price, individual paths span a wide interval, illustrating potential extreme moves under elevated volatility.

All visual outputs are generated automatically by the pipeline and saved as `ml_explanatory_insights.png`, `volatility_forecast.png`, and `monte_carlo.png`. Together, they provide a reproducible, interpretable, and visually intuitive foundation for short-term risk assessment in cryptocurrency markets, grounded entirely in local data and deterministic machine learning.

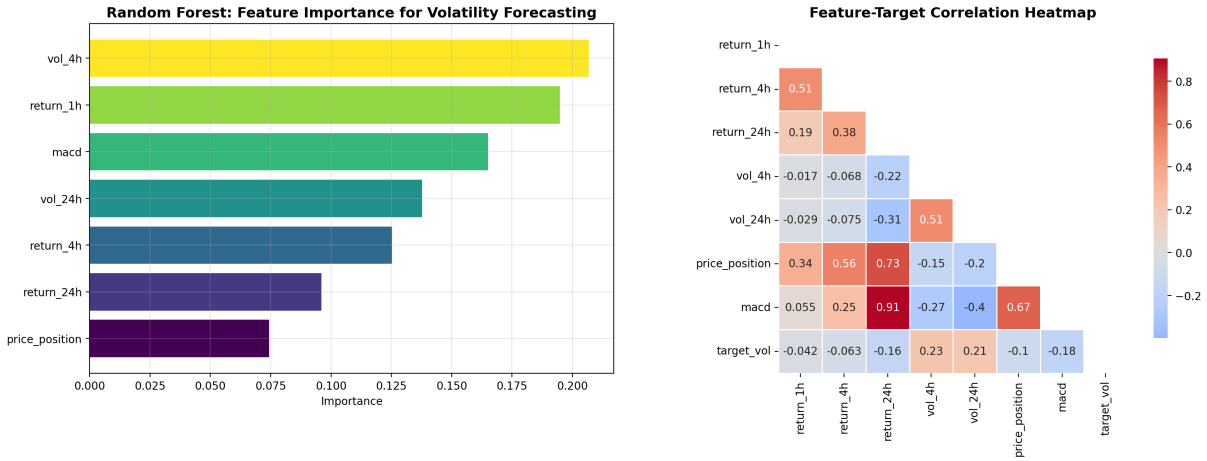

Figure 2: Model interpretability: feature importance (left) and feature–target correlation heatmap (right).

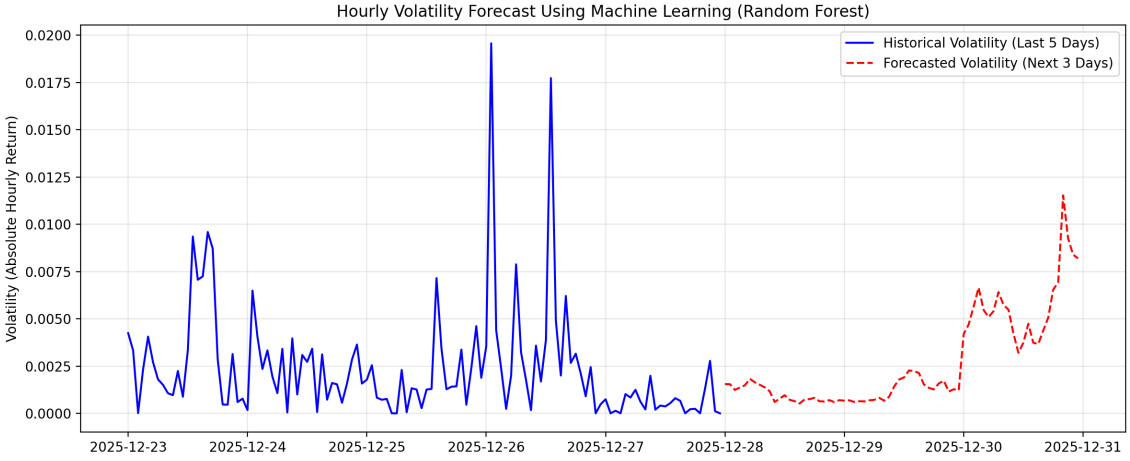

Figure 3: Historical and forecasted hourly volatility over an 8-day window (5 days observed, 3 days predicted).

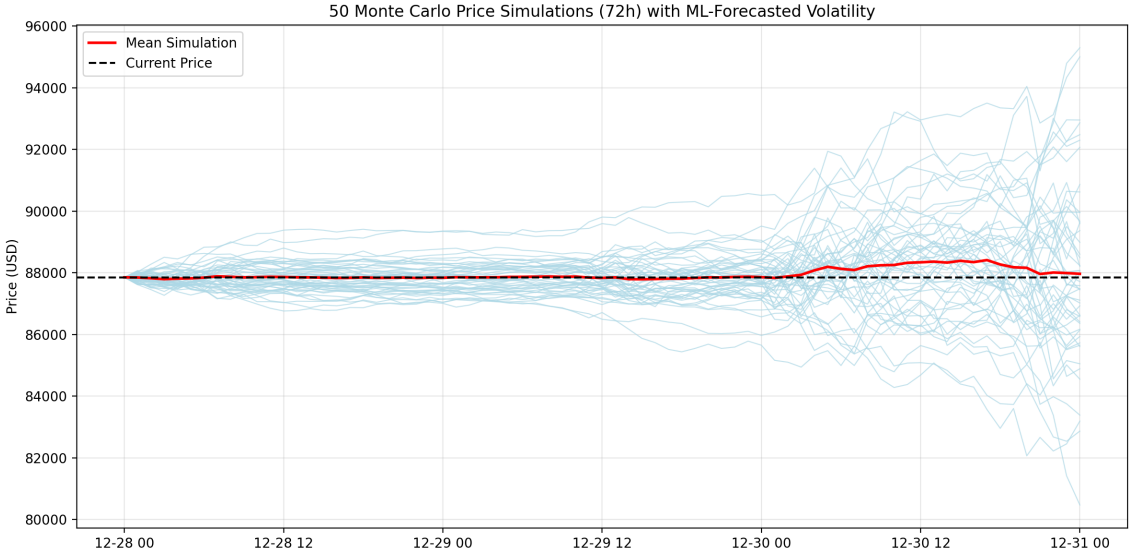

Figure 4: Fifty Monte Carlo price simulations over 72 hours, driven by ML-forecasted volatility.

7.1 Discussion

Thus, The system effectively shows that it can use LLM-assisted planning and reporting to carry out deterministic financial analysis. The project guarantees that numerical computations are fully auditable and repeatable by limiting the language model to tasks like intent parsing, analytical planning, and narrative generation. In practice, this division of duties worked well: technical indicators like RSI, MACD, and Bollinger Bands were calculated consistently over several runs, and the LLM produced understandable summaries of the outcomes. The result shows that deterministic pipelines and language-driven interfaces can coexist without sacrificing reliability.

The architecture's modular design was equally crucial for resilience and maintainability. With standardized input and output formats, each subsystem—data ingestion, enhancement, indicator computation, and reporting—was isolated, enabling autonomous development and debugging. This structure made sure that errors in the LLM components, like incorrect JSON or timeouts, did not affect the numerical analysis. Deterministic fallbacks, on the other hand, ensured that the pipeline generated valid outputs in every scenario. By avoiding unnecessary calculations and allowing for incremental updates when new data was added, the caching mechanism further improved efficiency.

To sum up, the project shows a solid and open method for incorporating AI methods into the analysis of financial data. The findings demonstrate that the limitations of LLM-only pipelines and the inflexibility of conventional quantitative tools can be overcome by a hybrid design that combines deterministic computation with language-driven planning. The system offers a solid basis for upcoming additions, like more sophisticated forecasting models or increased reporting capabilities, by guaranteeing reproducibility, auditability, and adaptability. In the end, the project confirms that employing AI agents to close the gap between adaptable human interaction and rigorous quantitative rigor in financial time series analysis is feasible.

8 Future research

While the current system demonstrates robustness and reproducibility in financial time series analysis, several directions for future research remain. These potential improvements span methodological advances, computational scalability, integration of state-of-the-art techniques, and enhancements to user interaction. Addressing these areas would not only improve system performance but also bring the framework closer to current research frontiers in computational finance and artificial intelligence.

One important direction concerns advanced data cleaning and preprocessing. The existing enhancement pipeline, which includes deduplication, imputation, logical consistency checks, and outlier detection, provides a reliable baseline. However, more sophisticated approaches could further improve data quality. Probabilistic imputation methods, such as Kalman filtering or Expectation–Maximisation algorithms, allow missing values to be

estimated based on underlying stochastic models of time series dynamics, potentially capturing latent market structure more effectively than deterministic forward and backward filling. In addition, recent research has explored machine learning–based data cleaning, using models such as autoencoders and recurrent neural networks to learn normal market behaviour and identify anomalous observations. Adaptive outlier detection techniques, including robust Bayesian changepoint detection, could also be investigated in order to adjust sensitivity dynamically in response to regime shifts, rather than relying on fixed thresholds.

Another avenue for future work lies in expanding the suite of technical indicators. While the current implementation focuses on widely used measures such as RSI, MACD, Bollinger Bands, and ATR, modern quantitative finance increasingly incorporates more advanced metrics. Risk-adjusted performance measures, including the Sharpe ratio, Sortino ratio, and Value-at-Risk, could provide deeper insight into downside risk and return efficiency. In addition, machine learning–derived features, such as latent factors obtained through Principal Component Analysis or autoencoder embeddings, may offer richer representations of market dynamics. Incorporating market microstructure information, such as bid–ask spreads, order book depth, or trade imbalances, could further improve short-term analytical accuracy, particularly in high-frequency settings.

Scalability and hardware acceleration represent a further research direction. Running large language models locally through Ollama imposes practical constraints on model size, prompt length, and inference latency. Leveraging modern GPU hardware could substantially reduce inference time and enable more complex analytical workflows, particularly when processing large datasets. Beyond single-node acceleration, distributed computing architectures could support parallel execution of the pipeline, enabling near real-time analysis of high-frequency financial data. In addition, deploying lightweight models on edge devices, such as ARM-based systems or FPGAs, could facilitate mobile or embedded financial applications while reducing dependence on centralised computing resources.

Future research could also explore tighter integration of state-of-the-art artificial intelligence models, while preserving the system's commitment to reproducibility. Transformer-based time series architectures, such as Temporal Fusion Transformers or Informer models, have demonstrated strong performance on non-stationary forecasting tasks and could enhance predictive capabilities if carefully integrated. Graph neural networks offer another promising direction, as financial assets often exhibit networked relationships across sectors or markets, and modelling these dependencies could improve understanding of systemic risk. In addition, reinforcement learning approaches to strategy optimisation could extend the system beyond analysis towards decision support, enabling the evaluation of long-term policies under uncertainty.

Improvements to user experience and interpretability also warrant further investigation. Interactive visualisation tools could allow users to explore indicators and data transformations dynamically, improving transparency and insight. The reporting agent could

be extended to provide more contextualised natural language explanations of indicator signals, helping bridge the gap between quantitative outputs and actionable understanding. Expanding multilingual support would further increase accessibility and reflect the global nature of financial markets.

Finally, ethical and security considerations should form an integral part of future research. Ensuring the privacy and secure handling of market data is essential, particularly when interacting with external data sources. As language models may inherit biases from their training data, systematic bias detection and mitigation strategies should be incorporated to improve reliability. Moreover, financial analysis systems are potential targets for adversarial manipulation, and research into robustness against adversarial inputs in time series models would strengthen the overall resilience of the framework.

9 Conclusion

The project shows that creating a financial analysis agent involves more than simply feeding market data into a sizeable language model. The team developed a flexible and repeatable system by carefully separating deterministic computation from language-oriented tasks. This design decision guarantees that traders, researchers, and analysts can engage with the agent in a natural way while maintaining confidence that the numerical results are auditable and free from the dangers of hidden stochasticity or hallucinations. In actuality, this implies that the agent can reconcile the rigorous requirements of quantitative finance with user-friendly interfaces.

The report also emphasizes the engineering difficulties associated with real-world financial data. The system's enhancement pipeline demonstrates how noise, missing values, and logical inconsistencies can be addressed in a transparent, rule-based manner. These issues are not edge cases, but rather everyday occurrences. A practical approach is reflected in the caching mechanism, stringent data abstraction layer, and fallback techniques for unstable LLM outputs: the system is built to fail gracefully, reuse previous work, and safeguard sensitive data. By making these choices, the agent becomes resilient enough to function in changing environments without compromising reproducibility.

Lastly, the project gains a forward-looking aspect through the incorporation of machine learning for volatility forecasting. The agent diagnoses the past and offers probabilistic insights into the future by fusing interpretable models with Monte Carlo simulations. This careful fusion of engineering rigour and financial intuition is demonstrated by the harmony between deterministic preprocessing, technical indicators, and stochastic forecasting. When considered collectively, the project provides a solid basis for future research, allowing for the incorporation of richer datasets and more sophisticated models without compromising the fundamental values of trust, reproducibility, and transparency.

References

[1] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[2] R. S. Tsay, *Analysis of Financial Time Series*. Wiley, 2010.

[3] C. Wu *et al.*, "Large language models for financial text analysis," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[4] Y. Zhao *et al.*, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2014.

[5] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*. Wiley, 2002.

[6] E. Chan, *Algorithmic Trading: Winning Strategies and Their Rationale*. Wiley, 2013.

[7] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1273–1283, 1993.

[8] Z. Bodie, A. Kane, and A. J. Marcus, *Investments*. McGraw-Hill, 2014.

[9] J. W. Wilder, *New Concepts in Technical Trading Systems*. Trend Research, 1978.

[10] J. Hull, *Options, Futures, and Other Derivatives*. Prentice Hall, 2017.

[11] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," *Risk Magazine*, 2004.

[12] G. Appel, "The moving average convergence-divergence indicator," *Technical Analysis of Stocks and Commodities*, 1979.

[13] J. Bollinger, *Bollinger on Bollinger Bands*. McGraw-Hill, 2001.

[14] A. W. Lo and J. Wang, "Trading volume: Definitions, data analysis, and implications," *The Review of Financial Studies*, vol. 13, no. 2, pp. 257–300, 2000.

[15] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–36, 2019.

[16] Y. Chen *et al.*, "Machine learning on edge devices: A survey," *IEEE Internet of Things Journal*, 2019.

Appendix

The following figure displays a screenshot of the Streamlit-based chat interface during an active analysis session, illustrating the user interaction and final report presentation.

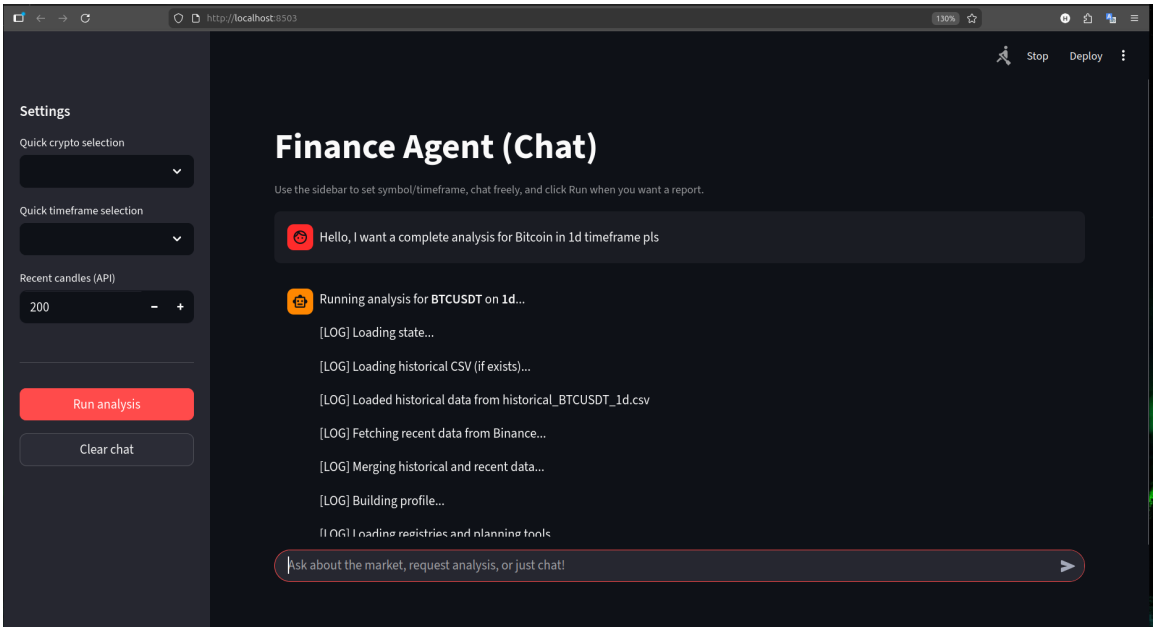

Figure 5: Chat interface working exemple part 1.

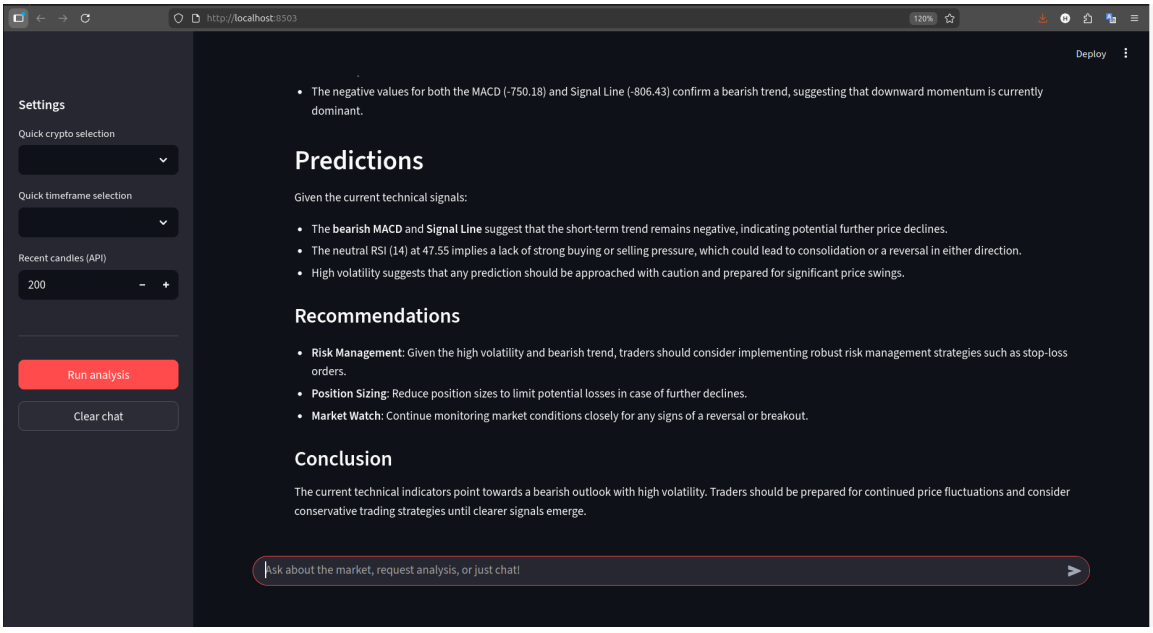

Figure 6: Chat interface working exemple part 2.

This appendix presents key artefacts from a representative execution for `BTCUSDT` at the daily timeframe. The system first generates a comprehensive data quality profile (`profile_BTCUSDT_1d.json`) for auditing and debugging. From this, a minimal,

anonymised version (`compact_profile_BTCUSDT_1d.json`) is derived and passed exclusively to the LLM in `planner.py`, ensuring no raw market data is ever exposed.

An example is provided below.


```
{
  "meta": {
    "symbol": "BTCUSDT",
    "timeframe": "1d",
    "rows": 216,
    "columns": [
      "Date",
      "Open",
      "High",
      "Low",
      "Close",
      "Volume"
    ]
  },
  "missing": {
    "count_cells": 1,
    "count_rows": 1,
    "counts_by_col": {
      "Open": 1
    },
    "items": [
      {
        "ts": "2025-11-26 00:00:00",
        "fields": [
          "Open"
        ],
        "missing_count": 1
      }
    ]
  },
  "duplicates": {
    "count_groups": 16,
    "count_rows_involved": 32,
    "groups": [
      {
        "group_size": 2,
        "ts": "2025-11-12 00:00:00",
        "diff_fields_count": 5,
        "is_identical": false
```

```
    },
    {
      "group_size": 2,
      "ts": "2025-11-13 00:00:00",
      "diff_fields_count": 5,
      "is_identical": false
    },
    {
      "group_size": 2,
      "ts": "2025-11-14 00:00:00",
      "diff_fields_count": 5,
      "is_identical": false
    },
    {
      "group_size": 2,
      "ts": "2025-11-15 00:00:00",
      "diff_fields_count": 5,
      "is_identical": false
    },
    {
      "group_size": 2,
      "ts": "2025-11-16 00:00:00",
      "diff_fields_count": 5,
      "is_identical": false
    },
    {
      "group_size": 2,
      "ts": "2025-11-17 00:00:00",
      "diff_fields_count": 5,
      "is_identical": false
    }
  ]
},
"bad_ohlc": {
  "count_rows": 3,
  "items": [
    {
      "ts": "2025-11-14 00:00:00",
      "reason": "low_gt_min(open,close)"
    },
    {
      "ts": "2025-11-18 00:00:00",
      "reason": "high_lt_max(open,close)"
```

```
      },
      {
        "ts": "2025-11-22 00:00:00",
        "reason": "low_gt_min(open,close)"
      }
    ]
  },
  "outliers": {
    "method": {
      "name": "robust_z_mad",
      "z_thresh": 6.0
    },
    "count_rows": 1,
    "count_by_field": {
      "Volume": 1
    },
    "items": [
      {
        "ts": "2025-11-21 00:00:00",
        "field": "Volume",
        "robust_z": 6.465544498556805
      }
    ]
  }
}
```


Listing 1: Full data quality profile `compact_profile_BTCUSDT_1d.json`

The planner output, `planner_BTCUSDT_1d.json`, shows the list of enhancement methods and technical indicators selected by the LLM based on the profile.


```
{
  "enhancement_plan": [
    {
      "step": 1,
      "method": "deduplicate_by_timestamp",
      "target": {
        "group_timestamps": [
          "2025-11-12 00:00:00",
          "2025-11-13 00:00:00",
          "2025-11-14 00:00:00",
          "2025-11-15 00:00:00",
          "2025-11-16 00:00:00",
          "2025-11-17 00:00:00"
        ]
```

```
    },
    "params": {
      "strategy": "keep_last"
    }
  },
  {
    "step": 2,
    "method": "fill_missing",
    "target": {
      "missing_rows": [
        "2025-11-26 00:00:00"
      ]
    },
    "params": {}
  },
  {
    "step": 3,
    "method": "fix_bad_ohlc",
    "target": {
      "bad_ohlcs": [
        "2025-11-14 00:00:00",
        "2025-11-18 00:00:00",
        "2025-11-22 00:00:00"
      ]
    },
    "params": {}
  },
  {
    "step": 4,
    "method": "winsorize_outliers",
    "target": {
      "outlier_field": "Volume",
      "outlier_row": "2025-11-21 00:00:00"
    },
    "params": {}
  }
],
"analysis_tools": [
  "returns",
  "rsi",
  "macd",
  "bollinger"
]
```

```
}
```

Listing 2: The planner_BTCUSDT_1d.json

Finally, the generated report (`BTCUSDT_1d_*.md`) synthesises the results into a concise technical analysis, confirming that all enhancements and indicators were applied correctly and that no raw market data was exposed to the LLM.

```
# Financial Analysis Report for BTCUSDT

## Summary
Generated on 2025-12-28T05:29:11.792759 UTC, the analysis is
   based on daily (1d) timeframe data.

## Technical Signals
- **Total Return**: -17.09%
  - This indicates a significant decline in the overall
     performance of BTCUSDT over the period under review.
- **Current Return**: -0.30%
  - The current return suggests a slight negative trend, but not
     as pronounced as the total return.
- **RSI (14)**: 47.55
  - RSI is currently at 47.55, indicating a neutral market
     condition where neither overbought nor oversold conditions
     are present.
- **MACD**: -750.18
  - The MACD value of -750.18 suggests a bearish trend, as the
     difference between the fast and slow moving averages is
     negative.
- **Signal**: -806.43
  - The signal line, at -806.43, further confirms the bearish
     stance by indicating that the short-term moving average has
     crossed below the long-term moving average.

## Risk Assessment
Based on the technical indicators above:
- The significant negative total return and current return
   suggest a high risk of continued downward pressure.
- The neutral RSI (14) indicates that there is no immediate
   urgency for a reversal, but it also means that further declines
    could occur without strong buying pressure to counteract them.
- Both MACD and Signal are in bearish territory, reinforcing the
   likelihood of further price decreases.

## Predictions
```

```
Given the current technical signals:
- The market appears to be trending downward with no clear signs
   of an imminent reversal.
- Traders should consider maintaining a cautious approach,
   possibly employing stop-loss orders to mitigate potential
   losses.
- Given the bearish MACD and Signal, it may be prudent to avoid
   long positions or consider shorting opportunities if risk
   tolerance is high.

### Recommendations
1. **Risk Management**: Implement strict risk management
   strategies, including stop-loss orders.
2. **Position Sizing**: Reduce position sizes given the current
   bearish trend.
3. **Monitoring**: Continue monitoring RSI and MACD for any signs
    of a potential reversal.

This analysis should be reviewed regularly as market conditions
   can change rapidly.
```

Listing 3: The report : `BTCUSDT_1d*.md`

This set of artefacts demonstrates the system's end-to-end functionality, reproducibility, and adherence to secure, deterministic processing.