EVALUATING LARGE LANGUAGE MODELS FOR FINANCIAL FORECASTING: A COMPARATIVE STUDY OF AUTOREGRESSIVE GENERATIVE, FIXED-HORIZON ROLLING-ORIGIN, AND BLOCK-BASED WORKFLOWS

By

Maya Magdlen

A capstone project submitted with University Honors

February 20, 2026

University Honors

University of California, Riverside

Dr. Rich Yueh

Department of Information Systems

Dr. Begoña Echeverria, Howard H Hays, Jr. Chair

University Honors

ABSTRACT

This study evaluates the effectiveness of large language models known as LLMs in numeric financial time-series forecasting by comparing three distinct forecasting workflow architectures, including prompt-driven, autoregressive generative workflows, fixed-horizon rolling-origin forecasting, and a structured, block-based forecasting pipeline with periodic resetting. The autoregressive generative workflows, represent the LLM-based aspect of the overall experiment. While LLMs have shown strong performance in sequence modeling tasks, how well they perform in the area of financial forecasting remains unclear. All workflows were evaluated under a shared out-of-sample period from December 2023, through December 2024, with identical accuracy, bias, and directional metrics, using S&P 500 daily closing prices. The autoregressive workflows are deployed through the use of prompt-driven LLM generation, where each predicted value is reused as input for the next prediction without periodic adjustment. The fixed-horizon rolling-origin workflow uses classical statistical models that are re-estimated at each forecast origin using a progressively increasing historical data window. Lastly, the block-based workflow creates forecasts over fixed horizons and then replaces the predicted values with the realized observations before refitting. The results of the experiment reveal that structured workflows, including rolling-origin and block-based design, achieve strong accuracy and relatively stable bias. The block-based workflow performs notably the best overall due to its near-zero bias and high stability because of its emphasis on error containment and periodic re-anchoring. The autoregressive workflows in contrast, underperform across all accuracy and bias metrics, showing strong tendencies to accumulate error, drift, and instability over long horizons. The findings from this experiment imply that forecasting performance is driven more by workflow and structural constraints as opposed to model complexity. Because of the nature of

LLMs, they are able to capture short-term patterns or directional signals, however, they aren't suited for sustained numeric forecasting without explicit stabilizing mechanisms and error control. Overall, the study reinforces the overall relevance of classical forecasting principles and emphasizes the central role that forecasting workflow architectures play in achieving reliable financial forecasts.

ACKNOWLEDGEMENTS

Above all I would like to sincerely thank my faculty mentor, Rich Yueh, for his enthusiastic and unwavering commitment and support throughout my capstone project and encouraging me to bring my ideas to life. I am truly grateful for the opportunity to work with someone of such dedication and insight. He has consistently provided me valuable advice to polish my ideas and provided me with all the resources I needed to act on my curiosity. For this he forever has my heartfelt appreciation.

I would also like to thank University Honors and all their staff for the great privilege to pursue research I'm deeply passionate about, allowing me to grow significantly in the process. University Honors staff has provided assistance and given me all the tools I needed to develop this project.

TABLE OF CONTENTS

Abstract 2

Acknowledgements 4

Introduction.....6

Literature Review.....7

Methodology.....11

Results.....19

Discussion.....24

Conclusion.....28

References.....30

Appendix.....32

1. Introduction

Large language models, otherwise known as LLMs, have been rapidly adopted across sectors of finance, analytics, and the broader workplace, often with the assumption that strong performance in sequence modeling tasks naturally translates into time-series forecasting. LLMs are increasingly applied to numeric sequences like financial prices, macroeconomic indicators, and operational metrics as a result of their excellence at next-token prediction in textual domains. However, whether this type of generative sequence modeling is fit for numeric financial time-series forecasting remains an open question requiring empirical validation.

It's important to note that financial time series differ fundamentally from natural language. Financial forecasting requires maintaining numeric stability across extended horizons, where small errors can easily accumulate and have meaningful economic consequences. Classical forecasting literature has emphasized that less complex, well-structured models are challenging to surpass in equity markets, and that success in forecasting heavily depends on assumptions about persistence, trend, and noise. Prompt-driven LLM-based forecasting, by comparison, typically operates as an autoregressive generative workflow, where predictions are generated sequentially and are recursively incorporated into subsequent inputs without any external re-anchoring to real values. While some configurations may introduce heuristic structure through utilizing prompting, these mechanisms ultimately do not involve formal statistical estimation or periodic recalibration using realized data. Because of this, the suitability of autoregressive generative workflows for the purpose of long-horizon numeric forecasting remains an open question to be explored.

This study addresses this gap in knowledge by providing a controlled empirical comparison between three forecasting workflow architectures, namely autoregressive generative

LLM workflows, fixed-horizon rolling-origin statistical workflows, and a block-based forecasting workflow with regular resetting to observed values. All of the approaches are evaluated on S&P 500 daily closing prices using a shared test-set evaluation period, identical metrics, as well as carefully designed forecasting workflows. Rather than optimizing models for maximum performance, the analysis centers around understanding how different model structures and workflow designs affect accuracy, bias, and long-horizon stability in numeric forecasting.

This paper examines the following research questions:

1. How do LLM-based autoregressive generative forecasting workflows compare to classical statistical time-series workflows in the task of forecasting daily equity prices?
2. How does forecasting workflow design, including autoregressive generation versus fixed-horizon and block-based forecasting, impact accuracy and bias metrics?
3. To what degree can heuristic-guided prompting improve the overall performance of autoregressive generative workflows?

A key goal of this study is to distinguish between different forms of generative forecasting, including unconstrained sequence continuation and heuristic-guided autoregressive generation, and to evaluate how increasing levels of such implicit structure directly impact forecasting performance.

2. Literature Review

This study connects classical time-series forecasting, where simple statistical models are often difficult to outperform under strict evaluation, and more recent work applying large language models (LLMs) and transformer-based architectures to time-series data. While both sectors are well developed, comparatively fewer studies provide controlled comparisons between

specifically, prompt-driven generative workflows and classical statistical forecasting workflows under a shared evaluation framework. Additionally, financial forecasting introduces further complexity at its base, as equity prices are widely considered to be highly difficult to predict due to the efficient incorporation of information into markets (Fama, 1970; Malkiel, 2003).

Classical Time-Series Forecasting

A consistent theme in forecasting research is that simple statistical benchmarks can be highly competitive, and that conclusions about model superiority strongly depend on evaluation design (Makridakis et al., 2018; Makridakis et al., 2020). Best practices prioritize strict out-of-sample testing as well as rolling-origin evaluation to reduce any optimism while avoiding look-ahead bias (Hyndman & Athanasopoulos, 2021). These results have led to the widespread use of naïve and exponential smoothing-based forecasting workflows as strong baseline comparisons when it comes to applied forecasting studies.

Exponential smoothing methods, such as the Holt-Winters framework, remain widely used due to their simplicity, interpretability, and proven performance across numerous domains (Gardner, 1985; Goodwin 2010). These approaches explicitly model level, trend, and seasonality, allowing for explicit structured control over factors like forecast dynamics and error propagation. Variants like damped-trend models further improve upon long-horizon stability by limiting unrealistic extrapolation (Taylor, 2003). More practical implementations of Holt-Winters forecasting focus on the importance of initialization, model structure, and parameter selection in achieving reliable, stable performance (Chatfield & Yar, 1988). In addition to this, robust extensions of exponential smoothing have been observed to perform well despite the presence of outliers and non-Gaussian noise, conditions that are typically observed in financial time series (Gelper et al., 2007).

In financial contexts, forecasting is challenging in many aspects, primarily due to the random nature of asset prices. According to the Efficient Market Hypothesis, price changes are largely unpredictable, and being able to consistently outperform simple benchmarks is difficult, especially in the cases of out-of-sample evaluation (Fama, 1970; Malkiel, 2003). While several empirical studies identify deviations from pure random walk behavior, these signals tend to be weak and difficult to translate into reliable forecasting gains (Lo & MacKinlay, 1988). As a result, simple models such as random walks and exponential smoothing more often than not serve as strong baselines in equity forecasting.

Language Models and Time-Series Forecasting

Recent work looks into the application of transformer-based and LLM-based forecasting workflows to time-series forecasting through architectural modifications, specialized training objectives, or reprogramming strategies, which differ in core principles from the prompt-only generative approach evaluated in this study. For example, Time-LLM is a method of reprogramming LLMs by mapping time-series inputs into representations compatible with language-model backbones. This process enables few-shot or zero-shot forecasting without any task-specific fine-tuning (Jin et al., 2023). In this context, “few-shot” and “zero-shot” forecasting refer to applying pretrained forecasting models to new time series without retraining their parameters. Similarly, Chronos, a transformer-based model trained specifically on time-series data using a language-modeling objective, reports strong zero-shot performance across a variety of diverse datasets (Ansari et al., 2024).

Other related foundation-model approaches, such as Lag-Llama, train large transformer models on wide collections of time series data for generating probabilistic forecasts in a zero-shot setting (Rasul et al., 2023). These model types further illustrate the potential of adapting

transformer architectures for the purpose of probabilistic time-series forecasting when architectures are explicitly designed for time-dependent data. Benchmarking work also implies that performance gains from language-model-based forecasting can be sensitive to specified forecast horizons, evaluation protocols, and metric choices (Tan et al., 2024). These findings only serve to further highlight the necessity for comparing new approaches to strong statistical baselines under controlled evaluation conditions.

On a more general level, LLMs have demonstrated strong performance in sequence modeling and natural language tasks, especially in cases with few-shot and zero-shot settings. However, these capabilities don't necessarily extend to numeric reasoning or stability over long horizons. Prior research indicates that LLMs often find difficulty with multi-step reasoning, displaying inconsistencies across generations, particularly as task complexity increases (Cobbe et al., 2021; Frieder et al., 2023). These limitations imply potential hurdles for applying autoregressive generative models to tasks involving long-horizon numeric forecasting. Methods like self-consistency have been suggested in order to improve reasoning accuracy by aggregating multiple outputs, illustrating the inherent variability and instability of generative predictions (Wang et al., 2022).

Research Gap

The majority of existing literature on LLM-based forecasting looks at adapted architectures, specialized training procedures, or benchmark datasets, leaving unresolved questions concerning the behavior of prompt-driven autoregressive generative workflows with respect to fully recursive forecasting conditions without external re-anchoring or statistical re-estimation. This study addresses this gap by comparing autoregressive generative workflows,

including both unconstrained sequence continuation and heuristic-guided generation, to structured statistical forecasting workflows under a shared evaluation framework.

3. Methodology

This study uses S&P 500 daily closing prices as the target time series. The dataset is divided into a training period and an out-of-sample evaluation period. The training period spans from January 2023 through November 2023, while the evaluation period covers December 2023 through December 2024. All price observations correspond only to valid U.S. equity trading days defined in the official market trading calendar.

This training period was chosen to provide a contiguous, recent historical window while refraining from overlap with the evaluation horizon. The evaluation period spans an entire calendar year in order to assess varying market conditions, including changes in volatility, trend, and momentum. All models assessed rely solely on historical closing prices, without reliance on intraday data or exogenous variables.

All models were trained on the same historical data described above and are evaluated against the same realized price series in order to ensure comparability across all forecasting methods. There was no smoothing, data revisions, or preprocessing beyond trading calendar alignment. The experiment was designed to isolate differences in forecasting performance to model structure and workflow rather than data availability or feature engineering. All forecasts were then evaluated using historical S&P 500 closing prices from December 2023 through December 2024.

Forecasting Framework

All forecasting workflows were evaluated under out-of-sample conditions using evaluation procedures tailored to each modeling approach, while preserving a common

evaluation period from December 2023 to December 2024. The forecasts were then aligned to the same sequence of realized S&P 500 daily closing prices and were evaluated using identical metrics. This design ensures that all workflow architectures mentioned above are evaluated on a consistent basis while preserving the natural operational behavior of each forecasting model. Three forecasting workflow architectures were evaluated in this experiment, described in detail below. LLM-based autoregressive generation workflows were implemented in ChatGPT. The fixed-horizon rolling-origin workflows were implemented using classical statistical models in Julius AI, an AI-assisted analytics platform, where the models produced were evaluated based on their statistical outputs rather than through the LLM front end. The block-based forecasting pipeline was implemented in Altair AI Studio (formerly RapidMiner), using its Time Series extension.

LLM-Based Autoregressive Generative Workflows

LLM-based forecasting workflow architectures operated as autoregressive generative workflows, where predictions were generated sequentially and recursively incorporated into subsequent inputs. Each model was initialized using historical daily closing prices from January 2023 through November 2023.

In all configurations, forecasts were generated without using realized values during the forecast horizon, and with no external re-anchoring, statistical re-estimation, or model updating. The LLM workflows analyzed in this study differ in the degree of structure introduced through different prompting, ranging from unconstrained sequence continuation to heuristic-guided autoregressive generation. The actual prompts used in the study, as well as ChatGPT's responses, are recorded in Appendix A.

There were four distinct LLM-based configurations evaluated:

- **LLM Version 1: Sliding-Window Autoregressive Forecasting (ChatGPT-4o)**
    - Model: ChatGPT-4o
    - Structure: Fixed, sliding 10-day window of the most recent predicted closing prices
    - Prompt methodology: This model was told to predict the next daily S&P 500 closing price through an LLM-style sequence continuation approach based on recent values. There were no explicit modeling assumptions, statistical estimation procedures, or stabilization mechanisms introduced in this run.
    - Forecasting behavior: One-day-ahead predictions were generated sequentially. After each prediction, the value was appended to the input window and used as context for the next prediction.
    - Purpose: The purpose was to evaluate baseline performance under unconstrained autoregressive generative forecasting.
- **LLM Version 2: Sliding-Window Autoregressive Forecasting (ChatGPT-5.2, Run 1, Error Accumulation Failure)**
    - Model: ChatGPT-5.2
    - Structure: Fixed, sliding 10-day window
    - Prompt methodology: Identical to Version 1
    - Forecasting behavior: It behaved fully autoregressive, utilizing day-by-day forecasting with predicted values reused as inputs.
    - Purpose: The purpose was to investigate whether a newer LLM model (ChatGPT 5.2) improves numeric stability under unconstrained autoregressive generation.

- **LLM Version 3: Heuristic-Guided Autoregressive Forecasting (ChatGPT-5.2, Run 2)**
    - Model: ChatGPT-5.2
    - Structure: Fixed, sliding 10-day window
    - Prompt methodology: This configuration introduced more structure through prompting by tasking the model with estimating internal state variables such as the current level (defined as a rolling mean), short-term drift, and volatility. The model was also prompted to classify the prevailing volatility regime and create predictions using a combination of drift, mean-reversion toward the estimated level, volatility-scaled stochastic variation, and a stability constraint intended to limit drift over longer horizons. The prompt also specified periodic adjustment of drift scaling at fixed intervals. While these components appear to resemble elements of statistical time-series models, they were employed entirely through prompting instead of formal parameter estimation or model fitting. The explicit recalibration mechanism represents internal heuristic adjustment as opposed to re-estimation using observed values.
    - Forecasting behavior: Predictions were generated in a fully autoregressive manner, utilizing day-by-day forecasting with predicted values reused as inputs. No realized values were used during the forecast horizon, and no outside correction or re-anchoring mechanisms were utilized.
    - Purpose: To assess whether introducing structured, prompt-based heuristics can improve autoregressive generative forecasting even without the use of external re-anchoring.

- **LLM Version 4: Pure Numeric Token Continuation (ChatGPT-5.2)**
    - Model: ChatGPT-5.2
    - Structure: Entire historical price sequence from January 2023 through November 2023 was provided as the initial dataset
    - Prompt methodology: The model was tasked with generating S&P 500 daily closing prices using a strict autoregressive numeric token prediction process to mimic how a language model predicts the next token in a sequence. There were no statistical models, regression techniques, parameter estimation, or explicit numerical system modeling allowed. The predictions were generated solely on pattern-based sequence continuation, without accuracy optimization, recalibration, or stability constraints being applied.
    - Forecasting behavior: Predictions were generated one day at a time, and each newly predicted price was added to the sequence and became a part of the context for the next prediction. This framework created a fully recursive autoregressive process, where the forecast trajectory was entirely dependent on prior predicted values. Unlike other LLM versions that relied on more narrative or heuristic continuation, this approach enforced strict sequential dependence as well as explicit state accumulation. Unlike the structured algorithmic approach in LLM Version 3, there were no explicit calculations of drift, volatility, or system parameters performed.
    - Purpose: The purpose of this design was to isolate the behavior of LLMs under unconstrained autoregressive generation, allowing for the direct observation of

factors such as error accumulation, drift, and long-horizon instability as a result of the recursive dependence without corrective mechanisms in place.

These designs intentionally evaluate a variety of autoregressive generative forecasting behaviors, ranging from unconstrained sequence continuation to heuristic-guided generation, while still maintaining fully recursive prediction dynamics in the absence of external re-anchoring or re-estimation.

Statistical Forecasting Workflow

In contrast, the fixed-horizon rolling origin workflow followed a statistical forecasting design. An initial forecast origin was set to the final trading day of November 2023, from there, forecasts were generated for the following 21 consecutive trading days, simulating approximately one trading month. After each forecast block, the origin shifts to the final trading day of the next month, and this forecasting process is then repeated through November 2024.

At each forecast origin, the statistical models used an expanding historical window containing all realized data available up to that date to re-estimate. These forecasts operated on fixed multi-day paths, with no mid-horizon correction or recursive updating. The forecasted values were not replaced with actuals until the full 21-horizon had elapsed. This helps to ensure it was a strictly out-of-sample evaluation.

The following classical statistical time-series models were evaluated:

- **Random Walk (No Drift)**, a model in which future prices are forecast as equal to the most recently observed closing value.
- **Random Walk with Drift**, this model extends the random walk by adding in an estimated average daily change.

- **Simple Exponential Smoothing (SES)**, models price levels using exponentially weighted averages without any seasonality or trend components.
- **Holt-Winters Exponential Smoothing with Additive Trend**, this model incorporates a deterministic linear trend component.
- **Holt-Winters Exponential Smoothing with Additive Damped Trend**, this model allows the influence of the trend component to decay over the course of the set forecast horizon.
- **Local Linear Trend Model (State-Space/Unobserved Components)**, which allows both trend and level to evolve in a probabilistic manner over time
- **ARIMA (1,1,1)**, estimated using a fixed model specification and re-estimated at each forecast origin for improved accuracy using the expanding historical window

These models were chosen to provide theoretically grounded as well as empirically established baselines for numerical financial forecasting. These models encode explicit structural assumptions about trend, random variation, persistence, and limit error propagation through fixed-horizon forecasting as well as habitual re-estimation.

Block-Based Forecasting Workflow

In Altair AI Studio, I implemented a user-defined, block-based forecasting pipeline. The block-based workflow architecture included data ingestion, preprocessing, and transformation steps, followed by a Holt-Winters exponential smoothing model with specific parameters I specified. Forecasts were created in fixed 21-trading-day blocks similar to the 21 forecast horizon applied in the statistical forecasting framework. After predictions were generated, they were replaced using realized values before the model was refitted for the subsequent block.

The forecasted values were never used as inputs for the following model estimation. This workflow demonstrates a production-oriented forecasting workflow design that emphasizes overall stability and error containment as opposed to unconstrained autoregressive generation.

Evaluation Metrics

All forecasting workflow architectures were evaluated using the same set of accuracy, directional, and bias metrics, applied uniformly across workflows. The metrics were calculated by comparing the forecasted S&P 500 daily closing prices to realized values over the shared out-of-sample evaluation period from December 2023 through December 2024. This standardized evaluation ensures that differences in performance are a reflection of differences in forecasting behavior as opposed to inconsistencies in measurement. The design allows for meaningful comparison across all workflow architectures mentioned, while preserving the natural operational behavior of each forecasting workflow, including the structural assumptions and updating mechanisms inherent in their underlying models.

Accuracy Metrics

Forecast accuracy was assessed using four primary measures. Mean Absolute Error (MAE) calculates the average magnitude of forecast errors in index points, providing a clear and interpretable measure of typical deviation. Root Mean Squared Error (RMSE) is a metric that penalizes larger errors more heavily, this emphasizes the impact of extreme forecast deviations. Mean Absolute Percentage Error (MAPE) expresses errors as a percentage of realized prices, enabling for a scale-independent comparison across all models. Lastly, R-Squared measures the extent to which forecasts co-moved with actual prices, demonstrating how well each workflow tracks price dynamics.

Directional Accuracy

Directional accuracy was computed to assess each workflow's ability to correctly predict the direction of price changes daily. In this case, it's the percentage of trading days for which the sign of the predicted daily change equals the sign of the actual change. This metric captures information relating to trend and momentum that may not show up in magnitude-based error measures.

Bias Assessment

Forecast bias was measured using mean error and paired two-sample t-Tests comparing forecasted and actual daily closing prices. The paired t-Tests assess whether the average forecasted price deviates statistically from the average realized price across the entire evaluation period. These results provide insight into systematic over- or under-predictions, which differentiate forecasting workflows that track price movements closely but exhibit persistent level bias from those that remain aligned in the long run.

4. Results

This section presents the out-of-sample forecasting performance of all the evaluated approaches from the period of December 2023 through December 2024. The results are organized through various metrics such as overall accuracy, forecast bias, directional performance, and observed failures. Summary statistics for all models are reported in Table 1, while the best-performing workflows can be found in Appendix C.

Overall Forecast Accuracy

Table 1, depicted below, reports accuracy methods, including MAE, RMSE, MAPE, and R^2 for all the forecasting approaches evaluated. There are distinct performance differences observed across these workflow architectures.

On average, the fixed-horizon rolling-origin workflow using classical statistical models achieved the lowest error levels in terms of magnitude-based accuracy measures. Among the statistical baselines, the random walk with drift and Holt-Winters-based models showed strong accuracy, particularly in terms of RMSE and R^2. These strong metrics reflect the models' ability to successfully track the overall level and co-movement of prices.

The block-based forecasting workflow employed in Altair AI Studio achieves one of the lowest MAE values among all the evaluated workflow architectures, while maintaining near-zero bias and stable performance across the entire evaluation period. This workflow consistently exhibited controlled deviations from realized values despite the workflow not being uniformly the lowest across all the error metrics.

The autoregressive generative LLM workflows, on the other hand, displayed significantly higher error levels across all included metrics. Performance across these configurations differed considerably, reflecting the variety in prompt structure and model behavior. Version 1 of the LLM-based workflow achieved the strongest performance, with the lowest MAE, RMSE, and MAPE percentage. Version 1 also had the highest R^2 of all generative configurations. Version 4 achieved moderate performance, where its error levels were higher than Version 1, but still substantially lower than the other configurations. Version 3, which utilized heuristic-guided prompting, ended up having higher error levels than Version 4 and lower explanatory power shown through its small R^2 value. Version 2 performed much worse than all the other approaches, with extremely large error values and a negative R^2, signifying performance below a naïve benchmark. Version 1 outperformed Version 4, followed by Version 3, while Version 2 clearly represents a failure case.

Table 1. ***Model forecast accuracy comparison. Lower MAE, RMSE, and MAPE indicate better performance, where a higher R^2 is preferred.***

Model	MAE	RMSE	MAPE (%)	R^2
LLM Version 1 (ChatGPT-4o)	166.12	188.67	3.02	0.78
LLM Version 2 (ChatGPT-5.2)	768.24	860.41	13.83	-3.54
LLM Version 3 (ChatGPT-5.2)	297.67	323.97	5.42	0.36
LLM Version 4 (ChatGPT-5.2)	202.51	224.21	3.78	0.58
Random Walk (no drift)	118.77	145.32	2.223	0.87
Random Walk with Drift	101.53	130.63	1.902	0.90
Simple Exponential Smoothing (SES)	121	147.82	2.263	0.87
Holt-Winters (Additive Trend)	106.01	138.04	1.974	0.88
Holt-Winters (Additive + Damped)	120.43	149.69	2.254	0.86
Local Linear Trend (State-Space)	121.22	150.13	2.229	0.85
ARIMA(1,1,1)	120.99	148.11	2.263	0.87
Altair AI Studio (Block-refit HW)	114.88	174.45	2.11	0.81

Bias and Mean Error Analysis

Bias analysis reveals insightful differences in how forecasting approaches manage recurring over- or under-prediction. The fixed-horizon rolling-origin workflow, particularly when implemented with models such as the random walk with drift, the local linear trend state-space model, and the damped Holt-Winters, exhibited fairly low MAE and RMSE while still exhibiting persistent mean error. This indicates that accurate short-term tracking doesn't necessarily coincide with long-run level alignment. In contrast, Holt-Winters with additive trend fails to reject the null hypothesis of zero mean error, suggesting effective long-run bias control within that workflow configuration.

The block-based forecasting workflow reflects the lowest absolute mean error across all the evaluated approaches, with a mean error of 6.68 index points, signifying strong long-run alignment with realized values.

All autoregressive generative LLM workflows had substantial negative mean errors, indicative of major bias. Paired t-Tests affirm that the mean forecast error of all LLM configurations differs significantly from zero. Among the LLM workflows, Version 1 had the

smallest absolute bias, followed by Version 4. Version 3 demonstrated larger bias even with the implementation of a heuristic structure. Version 2 was found to have the largest bias by a wide margin.

These results show that bias is a consistent feature in all autoregressive generative workflows, despite differences in prompt structure. These metrics can be viewed below in Table 2, and all formal paired t-Tests can be viewed in fuller detail in Appendix B.

Table 2. ***Bias and statistical significance of forecasts. Mean Error near zero indicates low bias while p-values > 0.05 indicate bias isn't statistically significant.***

Model	Mean Error (Bias)	t-Statistic	p-Value (two-tailed)	df	Reject H_0 (Bias = 0)?
LLM Version 1 (ChatGPT-4o)	−157.94	25.18	*5.91E-73*	271	Yes
LLM Version 2 (ChatGPT-5.2)	−767.96	32.58	*9.97E-96*	271	Yes
LLM Version 3 (ChatGPT-5.2)	-297.38	37.51	*2.69E-109*	271	Yes
LLM Version 4 (ChatGPT-5.2)	−199.59	29.95	3.33E-81	271	Yes
Random Walk (no drift)	−61.96	7.67	*3.21E-13*	271	Yes
Random Walk with Drift	−17.47	2.19	*0.02*	271	Yes
Simple Exponential Smoothing (SES)	65.66	8.17	*1.11E-14*	271	Yes
Holt-Winters (Additive Trend)	**−7.29**	**0.87**	***0.38***	**271**	**No**
Holt-Winters (Additive + Damped)	−48.97	5.71	*2.95E-08*	271	Yes
Local Linear Trend (State-Space)	−17.01	2.56	0.01	271	Yes
ARIMA(1,1,1)	-65.5	8.13	1.50E-14	271	Yes
Altair AI Studio (block-refit HW)	**6.68**	**−0.63**	**0.53**	**271**	**No**

Directional Accuracy

Directional accuracy results can be found in Table 3 below. Across workflow architectures, directional performance varies independently of accuracy in magnitude. Several workflows that reflected relatively low MAE and RMSE delivered directional accuracy barely above random chance, such as the block-based Holt-Winters workflow, indicating limited directional information content.

The local linear trend (state-space) model demonstrates high directional accuracy as found in Table 3. This suggests that forecasting workflows that include stochastic trend evolution may be able to more effectively capture directional trends, even when level accuracy may remain moderate.

The LLM workflows, on the other hand, reflect mixed directional performance overall. Versions 1 and 4 had the highest directional accuracy, both exceeding 57 percent. Version 2 achieved directional accuracy close to random chance, and Version 3 performed worse than random guessing. Version 3 with heuristic-guided structure doesn't show improved directional accuracy relative to its less structured counterparts. While several configurations have a directional accuracy above 50 percent, these gains don't necessarily translate into improved overall accuracy or reduced bias. These results suggest that directional correctness doesn't necessarily coincide with improvements in magnitude-based accuracy or bias and should be interpreted together with other performance metrics.

Table 3. ***Directional forecast accuracy. Higher percentages indicate greater ability to correctly predict the direction of price changes.***

Model	Directional Accuracy (%)
LLM Version 1 (ChatGPT-4o)	57.90
LLM Version 2 (ChatGPT-5.2)	49.26
LLM Version 3 (ChatGPT-5.2)	43.17
LLM Version 4 (ChatGPT-5.2)	57.87
Random Walk (no drift)	45.66
Random Walk with Drift	50.19
Simple Exponential Smoothing (SES)	46.52
Holt-Winters (Additive Trend)	49.08
Holt-Winters (Additive + Damped)	47.62
Local Linear Trend (State-Space)	76.92
ARIMA (1,1,1)	46.52
Altair AI Studio (block-refit Holt-Winters)	50.92

Failure Modes of LLM-Based Forecasting

Tables 1 and 2 together reveal several consistent failure modes in association with autoregressive generative LLM workflows. Persistent level divergence is evident in the negative mean error found in all LLM workflows, indicating regular drift, which is found even in configurations without large short-term errors.

Version 2 is a clear example of instability, with very large error values and negative explanatory power. This indicates extreme divergence from the underlying series.

Version 3's configuration still demonstrates high error and bias despite it having a heuristic-guided structure, showing prompt-based mechanisms aren't fully able to mitigate drift.

Summary of Results

The results reported in the tables reflect a clear performance hierarchy, with fixed-horizon rolling origin statistical workflows and block-based workflows consistently outperforming generative LLM approaches in terms of accuracy, bias, and stability. Autoregressive generative workflows show wide variability in performance but were consistently seen underperforming in comparison to structured forecasting workflows in multiple evaluation criteria.

5. Discussion

This study aimed to evaluate whether LLM-based autoregressive generative workflows can competently forecast numeric financial time series when compared against structured forecasting workflow architectures. These results ultimately indicate that forecasting performance is more reliant on workflow design, most notably the incorporation of mechanisms for error control, re-anchoring, and parameter updating, rather than model sophistication.

Forecasting Performance Interpretation

The design of these autoregressive workflows reflects how a practitioner would realistically deploy ChatGPT for forecasting tasks, namely through asking for sequential predictions and feeding the outputs back as context. The resulting error accumulation isn't a byproduct of the experimental design but rather an inherent characteristic of autoregressive generative forecasting workflows without re-anchoring. As a result, the comparisons between

LLM-based autoregressive generative, fixed-horizon rolling-origin, and block-based workflows is therefore not between equivalent conditions but between equivalent use cases, focusing on how each approach would perform if applied as intended.

The Version 1 LLM-based workflow operated on a relatively simple autoregressive structure, achieving the lowest error and bias across all LLM configurations, suggesting that more complex prompting doesn't necessarily create better performance results. Version 4, which used a pure numeric sequence continuation without additional structure, demonstrated moderate performance. This indicates that unconstrained autoregressive generation may be able to capture some aspects of short-term dynamics but falls short in maintaining stability compared to structured forecasting workflows. Version 3, which utilized heuristic-guided prompting through the use of components like drift, volatility, and mean reversion, was unable to outperform simpler LLM configurations and had a lower directional accuracy. This points to the idea that heuristic-guided prompting alone is inefficient in attempts to replicate the stabilizing properties of formally estimated statistical models, and that increasing prompt complexity can introduce additional variability without improving stability. This also suggests that prompt-based approximations of statistical structure don't necessarily carry over well in improving forecast performance, particularly when they aren't assisted by formal parameter estimation or external correction mechanisms. Version 2's extremely unfavorable metrics across the board reflect a failure mode where autoregressive generation deviates from the true series over time, leading to rapidly increasing forecast errors.

The comparison between Version 1, Version 3, and Version 4 reflects that introducing heuristic structure through prompting can influence short-term behavior, but it is unable to fundamentally alter the recursive nature of the forecasting process, leaving it open to cumulative

error over extended periods, as it lacks important mechanisms, including parameter estimation, likelihood optimization, or updating based on observed values.

In contrast, the strong performance of classical models employing a fixed-horizon rolling-origin workflow aligns with the decades of prior research concluding that simple, well-structured baselines are not easily surpassed in financial forecasting. Models under this workflow, such as random walk with drift and Holt-Winters variants, achieved small error levels and relatively minimal bias despite their simplicity, demonstrating the effectiveness of outlining clear assumptions regarding persistence, trend, and noise. Additive Holt-Winters, in particular, updates the level symmetrically around actual values, therefore limiting systematic over- or under-prediction when trends evolve smoothly. Holt-Winters with an additive damped trend performed worse in comparison due to its suppression of trend extrapolation, which can introduce systematic bias when trends remain persistent.

The block-based forecasting workflow achieves consistent performance through its utilization of a fixed forecast horizon and the periodic replacement of predictions with realized observations before refitting. This ensures forecasted values aren't used as inputs for subsequent estimation, and in doing so, eliminates within-block error propagation. This type of workflow architecture results in near-zero bias and strong long-run stability, even when its magnitude-based accuracy metrics, like RMSE, are not uniformly the lowest across all approaches.

The results reflect the importance of forecasting workflow design as a main driver of performance. Structured workflows such as the fixed-horizon rolling-origin approach and the block-based pipeline have their own mechanism to reduce error buildup and maintain close alignment with observed data. The fixed-horizon rolling-origin workflow does so through periodic re-estimation at each forecast origin, whereas the block-based workflow uses explicit

resetting to realized values between forecast blocks. Autoregressive generative workflows rely only on previously generated values. This distinction underlies the persistent bias and higher error levels seen in all LLM configurations. The contrast between these workflow designs demonstrates that forecasting performances are more strongly impacted by the utilization of error control and re-anchoring mechanisms, rather than model complexity alone. The block-based workflow reinforces the idea that stability can be obtained with relatively simple models when paired with effective workflow design, while highlighting the significant role of periodic correction and alignment with observed data when it comes to maintaining reliable forecasts.

Limitations

Several limitations should be considered when assessing the results from this study. The analysis focuses on a single equity index, the S&P 500, which can limit generalizability to other markets. It's also important to note that while all forecasting workflow architectures were evaluated over the same calendar period using identical accuracy and bias metrics, the forecasting horizons and update mechanisms differed by design across the workflow architectures. The fixed-horizon rolling-origin workflow and block-based workflows utilized fixed multi-day forecast blocks with periodic resetting, while LLM approaches generated forecasts autoregressively without correction methods. These differences, however, were intentionally preserved in order to reflect the natural operational behavior of each forecasting workflow, including their underlying model assumptions.

Additionally, the LLM-based generative workflow wasn’t fine-tuned for numeric forecasting, and hybrid statistical-LLM models weren’t explored. The results in this study shouldn’t be interpreted as a definitive assessment of all possible AI-based forecasting workflows. Nevertheless, the controlled use of a common dataset, evaluation period, and metrics

allows this study to isolate the effects of workflow design and model structure. It provides a clear characterization of the strengths and weaknesses of unconstrained forecasting workflows.

6. Conclusion

This study provides an empirical evaluation of LLM-based autoregressive generative workflows for numeric financial time-series forecasting by comparing their performance to structured forecasting workflow architectures, including fixed-horizon rolling-origin statistical workflows and a block-based forecasting workflow with periodic resetting, within a standardized out-of-sample evaluation design.

Autoregressive generative workflows consistently underperform structured forecasting workflows in measures of accuracy, bias, and long-horizon stability. LLM-based workflows are able to capture short-term patterns and directional movement in certain cases, however, their reliance on recursive prediction without any external re-anchoring leads to regular error accumulation and drift. The results of the LLM configurations also indicate that increasing prompt complexity, including heuristic-guided structure implemented in Version 3, doesn't reliably boost performance in autoregressive generative workflows.

A central insight of this study is that forecasting performance is driven mainly by workflow design as opposed to model complexity. Structured workflows utilize periodic re-estimation and re-anchoring to realized values, effectively limiting error propagation, and play a crucial role in maintaining long-run alignment with the data. Conversely, autoregressive generative workflows aren't equipped with these stabilizing mechanisms, leading to persistent bias and reduced stability over extended horizons.

These findings are supported by existing forecasting literature, which provides evidence that simple, well-structured statistical workflows serve as strong baselines in financial

forecasting, as the fixed-horizon rolling origin statistical workflows yielded high accuracy and consistency across all metrics.

The results imply that LLMs, as they stand, are not inherently unsuitable for numerical forecasting but that their effectiveness largely depends on their integration within structured forecasting workflows. Pure autoregressive generative implementations aren't fit for long-horizon numeric forecasting. While this may be the case, hybrid or structured approaches that utilize external anchoring or recalibration in their workflows may show more promising performance in this area.

Overall, this work contributes empirical evidence that workflow design, especially mechanisms for error control and re-anchoring, is a key driver of forecasting performance. These results reaffirm the persistent relevance of classical forecasting principles, while drawing attention to the importance of integrating generative models within structured forecasting workflows for practical financial applications.

References

Ansari, A. F., Stella, L., Turkmen, A. C., Zhang, X., Mercado, P., Shen, H., … Wang, Y. (2024). *Chronos: Learning the language of time series*. arXiv. https://arxiv.org/abs/2403.07815

Chatfield, C., & Yar, M. (1988). Holt–Winters forecasting: Some practical issues. *The Statistician, 37*(2), 129–140. https://doi.org/10.2307/2348257

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, Ł., … Schulman, J. (2021). *Training verifiers to solve math word problems*. arXiv. https://arxiv.org/abs/2110.14168

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417. https://doi.org/10.2307/2325486

Frieder, S., Pinchetti, L., Chevalier, A., Grinberg, N., Joseph, M., & Basu, S. (2023). *Mathematical capabilities of ChatGPT*. arXiv. https://arxiv.org/abs/2301.13867

Gardner, E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting, 4*(1), 1–28. https://doi.org/10.1002/for.3980040103

Gelper, S., Fried, R., & Croux, C. (2007). Robust forecasting with exponential and Holt–Winters smoothing. *Journal of Forecasting, 26*(3), 145–157. https://doi.org/10.1002/for.1032

Goodwin, P. (2010). The Holt–Winters approach to exponential smoothing: 50 years old and going strong. *Foresight: The International Journal of Applied Forecasting, 19*, 30–33.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts. https://otexts.com/fpp3/

Jin, M., Liu, Y., Zhang, Y., & Chen, H. (2023). *Time-LLM: Time series forecasting by reprogramming large language models*. OpenReview. https://openreview.net/forum?id=Unb5CVPtae

Lo, A. W., & MacKinlay, A. C. (1988). Stock market prices do not follow random walks: Evidence from a simple specification test. *The Review of Financial Studies, 1*(1), 41–66. https://doi.org/10.1093/rfs/1.1.41

Makridakis, S., Petropoulos, F., & Kang, Y. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE, 13*(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting, 36*(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives, 17*(1), 59–82. https://doi.org/10.1257/089533003321164958

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., & Vollgraf, R. (2023). *Lag-Llama: Towards foundation models for probabilistic time series forecasting*. arXiv. https://arxiv.org/abs/2310.08278

Tan, M., et al. (2024). *Are language models actually useful for time series forecasting?* Advances in Neural Information Processing Systems. https://proceedings.neurips.cc/paper_files/paper/2024/file/6ed5bf446f59e2c6646d23058c86424b-Paper-Conference.pdf

Taylor, J. W. (2003). Exponential smoothing with a damped multiplicative trend. *International Journal of Forecasting, 19*(4), 715–725. https://doi.org/10.1016/S0169-2070(03)00004-5

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2022). *Self-consistency improves chain-of-thought reasoning in language models*. arXiv. https://arxiv.org/abs/2203.11171

APPENDIX TABLE OF CONTENTS

Appendix A: ChatGPT Prompts, Exact Responses, Model And Date Accessed…….….33

Appendix B: Paired T-Tests…….....……...……..43

Appendix C: Top 3 Performing Workflows……...……...49

Appendix A- ChatGPT Prompts, Exact Responses, Model, And Date Accessed

Version: LLM Version 1

ChatGPT Model: 4o

Date Accessed: May 22nd, 2025

Please note the responses are direct wording taken from ChatGPT as a reply to my prompts.

Prompt:

I gave you an excel file "Capstone Project database", please refer to the sheet titled "2023 Daily S&P 500 Jan-Nov". These are the actual daily closing prices of the S&P 500 from January to November 2023. I want you to predict the closing prices of December 2023 to December 2024 using an LLM-style sequence continuation approach with a 10-day sliding window. For each trading day, infer the next value based purely on pattern recognition and contextual continuation, don't use any statistical models, regression, or ML. After each prediction, append the predicted value to the dataset and include it in the input window for subsequent predictions. Please generate the predictions sequentially, while ensuring the outputs reflect realistic market behavior and mimic pure LLM token-style continuation instead of numeric optimization. First start with predicting December 2023. Align your predictions with the trading day calendar I've given you in the sheet titled "2024 Daily actual".

Response:

Got it — you want to use LLM-style prediction logic, but still apply it in a structured 10-day sliding window approach, like a time series. Here's how this will work: We'll simulate how a language model (like me) would continue a sequence of S&P 500 closing prices based on recent

values. This means: I won’t fit a statistical model (like linear regression). I will "predict the next number" based purely on pattern recognition, intuition, and linguistic continuation — just as if I were generating the next number in a text. After each prediction, it becomes part of the next window — following your autoregressive format. Example of LLM-style reasoning: Given the last 10 closing prices like: 4530, 4552, 4561, 4557, 4570, 4582, 4591, 4589, 4594, 4600 An LLM might say: “There’s an upward momentum with slight noise; let’s continue that rhythm.” Predicted next value: 4607 Then it would shift the window and use: 4552, 4561, ..., 4600, 4607 for the next day. Let’s Proceed: I'll now generate LLM-style daily predictions for December 2023 using this logic, one by one, using a 10-day sliding window starting from the final actual values of Nov 2023. Let’s begin. Here is a step-by-step LLM-style prediction for each trading day in December 2023, using a 10-day sliding window and linguistic/pattern-based intuition — not statistical models. Last 10 real values from November 2023: From your dataset, the final 10 trading days are: 4530.75, 4552.35, 4561.58, 4557.57, 4570.30, 4582.00, 4591.70, 4589.75, 4594.43, 4600.31. An LLM might say: “There’s an upward momentum with slight noise; let’s continue that rhythm.” Predicted next value: 4607. Then it would shift the window and use: 4552, 4561, ..., 4600, 4607 for the next day.

Version: LLM Version 2

ChatGPT Model: 5.2

Date Accessed: January 2nd, 2026

Prompt:

I gave you an excel file titled “Capstone Project database”, please refer to the sheet titled “2023 Daily S&P 500 Jan-Nov”. These are the actual daily closing prices of the S&P 500 from January to November 2023. I want you to predict the closing prices of December 2023 to December 2024 using an LLM-style sequence continuation approach with a 10-day sliding window. For each trading day, infer the next value based purely on pattern recognition and contextual continuation, don’t use any statistical models, regression, or ML. After each prediction, append the predicted value to the dataset and include it in the input window for subsequent predictions. Please generate the predictions sequentially, while ensuring the outputs reflect realistic market behavior and mimic pure LLM token-style continuation instead of numeric optimization. First start with predicting December 2023. Align your predictions with the trading day calendar I’ve given you in the sheet titled “2024 Daily actual”.

Response:

Absolutely — I can do that. Before I start: I’m going to keep the exact same mechanics you asked for (10-day sliding window + autoregressive use of prior predictions), and I’m going to write the reasoning in a deliberately “LLM-style,” long, drawn-out way: describing trend feel, rhythm, volatility, micro-structure-like “texture,” gentle mean reversion, and “what the next token in the sequence should look like,” without fitting a statistical model. I’ll do December 2023 trading days (excluding 12/25 market holiday).

Detailed LLM-style reasoning:

When I "read" this 10-number sequence like text, I see a narrative that starts with lower values (4508 → 4514), then a noticeable push upward (to 4547), a small dip (4538), and then a steady "climb-with-breathing" pattern (4556 → 4559), followed by a couple small oscillations around the mid-4550s, and then a final strong uptick to 4567.80.

What matters most in an LLM-style continuation is not a calculated slope, but the *rhythm of change*: you have a pattern of upward drift with intermittent pauses — like the series "wants" to go up, but it doesn't do it in a perfectly smooth way. The last value (4567.80) looks like a momentum punctuation mark — a strong final token that often invites either:

Version: LLM Version 3

ChatGPT Model: 5.2

Date Accessed: January 9th, 2026

Prompt:

I gave you an excel file titled “Capstone Project database”, please refer to the sheet titled “2023 Daily S&P 500 Jan-Nov”. These are the actual daily closing prices of the S&P 500 from January to November 2023. I want you to predict the daily closing prices of December 2023 to December 2024. I want you to implement an autoregressive forecasting model for daily S&P 500 closing prices, where each closing value is regarded as a sequential token. At each step operate using only the most recent 10 observed or generated closing prices. For the rolling window estimate these state variables: the current level which is defined as the 10-day mean, the short-term drift, and the volatility. I want you to classify the current volatility regime based on the estimated volatility. Please generate the next day’s closing price by combining the following: a drift component adapted to the current regime, a mean-reversion term pulling the price toward the current estimated level, a long-horizon stability control to prevent divergence, and a small stochastic innovation term scaled based on the estimated volatility. After you generate each prediction, please append it to the sequence so it becomes a part of the input for subsequent steps, and maintain a fully self-referential process based on a rolling 10-day window. I want to recalibrate the drift scaling factor every 20 predictions in order to prevent growth or stagnation. Align the predictions with the trading day calendar I’ve attached.

Response:

Done — I implemented a fully mechanical, day-by-day 10-day sliding-window algorithm (each prediction becomes part of the next 10-day window), generated all predictions from 12/1/2023 through 12/31/2024, and recomputed all metrics against the "2024 Daily actual" sheet in your Excel file.

For each trading day from 12/1/23 → 12/31/24:

1. Windowing
    - Take the last 10 closes available (from history + previous predictions).

2. Estimate internal state
    - Level = mean of last 10 closes
    - Short-term drift = mean of log returns over last 9 intervals
    - Volatility = std dev of log returns over last 9 intervals
    - Volatility regime = calm / moderate / expanding (based on volatility thresholds)

3. Generate the next-day close
    - Predicted next-day return is constructed from:
        - drift component (trend following)
        - mean reversion toward level
        - volatility-responsive "shock" (deterministic, via a sine function so it's reproducible)

        - scheduled recalibration every 20 steps adjusting drift scaling to prevent runaway or stagnation
    - Convert predicted return into predicted price and append.

This is true autoregressive generation: every prediction is reused as input.

Version: LLM Version 4

ChatGPT Model: 5.2

Date Accessed: January 13th, 2026

Prompt:

I have an Excel file called “Capstone Project Database”. Use the sheet titled “2023 Daily S&P 500 Jan-Nov” as historical context and the sheet “2024 trading days” as the list of dates to forecast. Please generate daily S&P 500 closing price predictions for December 2023 through December 2024 using a strict autoregressive numeric token process, similar to how a language model would predicts tokens sequentially. The generation should operate step-by-step, one day at a time. At each step, only use the historical data provided, plus all the previously generated predictions as the full context. Don’t under any circumstances, generate the entire sequence at once or plan ahead, as each prediction should be based only on information that is available up to that specific point in time. Treat each predicted closing price as a new data point and append it to the sequence, so that it becomes part of the context for the next prediction. Have an internal record of all the generated values and make sure each prediction depends directly on earlier values in the sequence. Don’t fit statistical models, estimate parameters, or apply external optimization. Don’t use any future information or global trajectory planning in the process at all. I want you to simulate true autoregressive token generation. Make sure the generated values maintain a realistic scale, magnitude, and decimal precision that is consistent with S&P 500 daily closing prices.

Response:

Got it. I'll treat this as true step-by-step autoregressive numeric token generation, not a batch forecast.

Here's exactly how I'll proceed:

1. Initialize the context (the "prompt")
    - I'll load the sheet "2023 Daily S&P 500 Jan-Nov" and take its closing prices as the initial numeric token sequence.
    - The most recent portion of that sequence will implicitly carry the strongest influence, the same way the tail of a text prompt most strongly shapes next-token prediction.
2. Define the forecast timeline
    - I'll use the sheet "2024 trading days" strictly as the ordered list of trading dates to be generated (and I'll include December 2023 trading days first, then all of 2024 through December 2024).
3. Generate one value at a time (strict autoregression)
    - For the next trading day, I will generate exactly one closing price as a plausible continuation of the existing numeric sequence.
    - The prediction will be conditioned only on the sequence so far (historical + previously generated predictions).
    - I will not "look ahead," sketch a full path, or implicitly aim for an end-of-year target.
4. Append immediately and update context

- After producing that single predicted close, I will append it to the sequence as if it were an observed closing price.
- That appended value becomes part of the context used to generate the next day's prediction.

5. Maintain an internal running record
    - I will keep a complete ordered record of generated values so that each new prediction directly depends on prior predictions via the updated sequence.
6. No modeling or calibration
    - I will not fit ARIMA/regression/ML models, estimate drift/volatility parameters, optimize error, or apply smoothing constraints.
    - The only driver is *free-form numeric continuation* that preserves realistic scale, step size, and decimal precision typical of S&P 500 closes.
7. Output format
    - I'll output one predicted closing price per trading day for the required date range, matching realistic S&P 500 formatting (two decimals), with no extra commentary in the final prediction output.

Appendix B- Paired T-Tests

Note that Variable 1 represents realized closing prices and Variable 2 is forecasted closing prices.

Paired T-Test for LLM Version 1

LLM Version 1 (ChatGPT-4o)	*Variable 1*	*Variable 2*
Mean	5373.58926	5215.65224
Variance	163728.402	121586.737
Observations	272	272
Pearson Correlation	0.97319472	
Hypothesized Mean Difference	0	
df	271	
t Stat	25.1889293	
P(T<=t) one-tail	2.9559E-73	
t Critical one-tail	1.65049578	
P(T<=t) two-tail	5.9118E-73	
t Critical two-tail	1.96875631	

Paired T-Test for LLM Version 2

LLM Version 2 (ChatGPT-5.2)	*Variable 1*	*Variable 2*
Mean	5373.5893	4605.625
Variance	163728.4	269.59121
Observations	272	272
Pearson Correlation	0.9713534	
Hypothesized Mean Difference	0	
df	271	
t Stat	32.584123	
P(T<=t) one-tail	4.989E-96	
t Critical one-tail	1.6504958	
P(T<=t) two-tail	9.977E-96	
t Critical two-tail	1.9687563	

Paired T-Test for LLM Version 3

LLM Version 3 (ChatGPT-5.2)	*Variable 1*	*Variable 2*
Mean	5369.835699	5076.212404
Variance	164756.1576	95399.01683
Observations	272	272
Pearson Correlation	0.971082026	
Hypothesized Mean Difference	0	
df	271	
t Stat	37.51044296	
P(T<=t) one-tail	1.3453E-109	
t Critical one-tail	1.650495779	
P(T<=t) two-tail	2.6906E-109	
t Critical two-tail	1.968756314	

Paired T-Test for LLM Version 4

LLM Version 4 (ChatGPT-5.2)	*Variable 1*	*Variable 2*
Mean	5373.5893	5159.4494
Variance	163728.4	121302.18
Observations	272	272
Pearson Correlation	0.9711457	
Hypothesized Mean Difference	0	
df	271	
t Stat	33.211615	
P(T<=t) one-tail	7.961E-98	
t Critical one-tail	1.6504958	
P(T<=t) two-tail	1.592E-97	
t Critical two-tail	1.9687563	

Paired T-Test for Random Walk

Random Walk (no drift)	*Variable 1*	*Variable 2*
Mean	5370.72654	5308.76617
Variance	165446.685	174088.591
Observations	272	272
Pearson Correlation	0.94922878	
Hypothesized Mean Difference	0	
df	271	
t Stat	7.67348011	
P(T<=t) one-tail	1.6063E-13	
t Critical one-tail	1.65062398	
P(T<=t) two-tail	3.2125E-13	
t Critical two-tail	1.96895628	

Paired T-Test for Random Walk with Drift

Random Walk with Drift	*Variable 1*	*Variable 2*
Mean	5370.7265	5353.2592
Variance	165446.69	177722.59
Observations	272	272
Pearson Correlation	0.9515888	
Hypothesized Mean Difference	0	
df	271	
t Stat	2.196478	
P(T<=t) one-tail	0.014462	
t Critical one-tail	1.650624	
P(T<=t) two-tail	0.0289241	
t Critical two-tail	1.9689563	

Paired T-Test for Simple Exponential Smoothing

Simple Exponential Smoothing (SES)	*Variable 1*	*Variable 2*
Mean	5371.7104	5306.0463
Variance	165109.9	173572.57
Observations	272	272
Pearson Correlation	0.9483201	
Hypothesized Mean Difference	0	
df	271	
t Stat	8.177326	
P(T<=t) one-tail	5.578E-15	
t Critical one-tail	1.650475	
P(T<=t) two-tail	1.116E-14	
t Critical two-tail	1.9687238	

Paired T-Test for Holt-Winters with Additive Trend

Holt-Winters (Additive Trend)	*Variable 1*	*Variable 2*
Mean	5371.7104	5364.4174
Variance	165109.9	173730.2
Observations	272	272
Pearson Correlation	0.9440237	
Hypothesized Mean Difference	0	
df	271	
t Stat	0.872581	
P(T<=t) one-tail	0.1918306	
t Critical one-tail	1.650475	
P(T<=t) two-tail	0.3836611	
t Critical two-tail	1.9687238	

Paired T-Test for Holt-Winters with Additive Trend and Damping

Holt-Winters (Additive + Damped)	*Variable 1*	*Variable 2*
Mean	5371.7104	5322.7396
Variance	165109.9	187097.03
Observations	272	272
Pearson Correlation	0.9448265	
Hypothesized Mean Difference	0	
df	271	
t Stat	5.7097908	
P(T<=t) one-tail	1.479E-08	
t Critical one-tail	1.650475	
P(T<=t) two-tail	2.958E-08	
t Critical two-tail	1.9687238	

Paired T-Test for Local Linear Trend

Local Linear Trend (State-Space)	*Variable 1*	*Variable 2*
Mean	5371.7104	5351.5433
Variance	165109.9	177067.92
Observations	272	272
Pearson Correlation	0.9512827	
Hypothesized Mean Difference	0	
df	271	
t Stat	2.5655695	
P(T<=t) one-tail	0.0054185	
t Critical one-tail	1.650475	
P(T<=t) two-tail	0.010837	
t Critical two-tail	1.9687238	

Paired T-Test for ARIMA (1,1,1)

ARIMA(1,1,1)	*Variable 1*	*Variable 2*
Mean	5371.7104	5306.2067
Variance	165109.9	173951.21
Observations	272	272
Pearson Correlation	0.9480849	
Hypothesized Mean Difference	0	
df	271	
t Stat	8.1323665	
P(T<=t) one-tail	7.522E-15	
t Critical one-tail	1.650475	
P(T<=t) two-tail	1.504E-14	
t Critical two-tail	1.9687238	

Paired T-Test for Block-Based Holt-Winters

Altair AI Studio (block-refit HW)	*Variable 1*	*Variable 2*
Mean	5373.5893	5380.2713
Variance	163728.4	167331.09
Observations	272	272
Pearson Correlation	0.9079265	
Hypothesized Mean Difference	0	
df	271	
t Stat	-0.63102	
P(T<=t) one-tail	0.2642796	
t Critical one-tail	1.6504958	
P(T<=t) two-tail	0.5285593	
t Critical two-tail	1.9687563	

Appendix C- Top 3 Performing Workflows

Appendix C. ***Displays the top-ranking workflows in terms of accuracy and bias metrics.***

Metrics	Random Walk with Drift	Holt–Winters (Additive Trend)	Altair AI Studio (Block-refit HW)
Overall Performance Rank	1	2	3
MAE	101.53	106.01	114.88
RMSE	130.63	138.04	174.45
MAPE(%)	1.90	1.97	2.11
R^2	0.89	0.88	0.81
Mean Error	-17.47	-7.29	6.68
Directional Accuracy (%)	50.19	49.08	50.92
Bias Statistically Significant?	Yes	No	No