Incorporating Time-Series Price Prediction and Quantitative Trading Strategy

Yuzhe Jin

Brasenose College

University of Oxford

Supervised by Professor Wesley Armour

Submitted in partial fulfilment of the

MEng in Engineering Science

Trinity Term 2025

Abstract

This project investigates the end-to-end application of machine learning for financial time-series forecasting and the corresponding automated trading strategy implementations. We develop and evaluate a progression of predictive models, starting from a classical ARIMA baseline through to a diversified ensemble of machine learning models, a Mixture of Experts (MoE) gating architecture, and finally a hybrid model combining MoE with the other best-performing selected models. In parallel, we design and refine trading strategies that convert these forecasts into buy or sell decisions. Naive baseline strategies are first implemented in conservative or optimistic philosophies. Then, a confidence-weighted voting mechanism was adopted in the trading strategy to dynamically learn the performance of each model over the past days. Across a multi-asset portfolio, we demonstrated that advanced machine learning models integrated with ensemble trading strategies can significantly enhance price prediction accuracy and trading performance. The encouraging final results resonate with our project objectives to enable individual investors to participate effectively in quantitative algorithmic trading within the financial industry.

Acknowledgments

I would like to express my sincere gratitude to my project supervisor, Professor Wesley Armour, for his invaluable guidance, support and encouragement throughout this research project. His patience for illustration and expertise in machine learning and time series analysis greatly steered my project direction when I encountered obstacles in my work. Besides, I am equally thankful to Dr Yishun Lu at the Oxford e-Research Centre for his technical insights and assistance with remote server configurations.

Finally, I wish to express my deep and heartfelt appreciation to my family for their unwavering support and encouragement not only throughout the degree but also throughout my entire life. Additionally, I want to say huge thanks to my friends for their camaraderie and dedicated support, including my peers from Brasenose College, cohorts at University of Oxford and friends beyond the university. Without you, the project might not be flourished as it represents today.

—— *"We used to look up at the sky and wonder at our place in the stars.*
Now we just look down and worry about our place in the dirt."

Contents

1 Introduction

1.1 General

Financial markets produce vast amounts of time-series data every day, such as stock, commodity and bond prices. Prices are influenced by a complex combination of factors – macro-economic indicators, geopolitical events, market sentiments, especially for the emergence of black swan events. All the above reasons make it a long-standing challenge for financial market predictions [1]. Even marginal improvements in forecasting accuracy could result in considerable financial gains when incorporated with trading strategies. According to the Efficient Market Hypothesis (EMH), asset class prices should reflect all available information for all investors in the market. This implies that price movements are expected to respond quickly to market changes, leaving future predictions technically infeasible [2]. Despite this theoretical challenge, researchers and industry professionals continue to explore new methodologies to capture subtle patterns or inefficiencies in financial markets.

So, as we have seen, throughout the evolution of the quantitative finance industry, dominant institutions and companies could leverage their substantial resources to develop advanced price prediction models and sophisticated trading algorithms. Consequently, it appears that only these well-established entities can secure a competitive edge in financial markets, whereas uninformed individual investors remain disadvantaged due to disparities in computational power and access to hidden information behind market data.

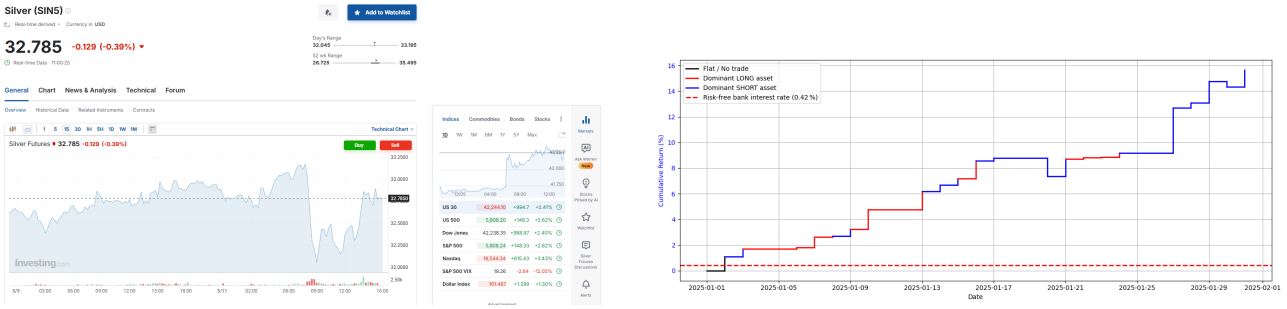
However, the story might be changed over the recent years. The advent of machine learning (ML) models has revolutionised financial market forecasting. The ML-based methods outperformed traditional statistical approaches in capturing intricate patterns within a large time-series dataset [3]. Moreover, deep learning (DL), a subset of the ML field, has shown great potential in modelling complex and non-linear relationships inherent in financial time series data points [4] [[5].

1.2 Project Overview

Despite these developments mentioned above, individual investors may still find it challenging when seeking to harness Artificial Intelligence (AI) for trading. Implementing DL models effectively not only requires quality datasets and computational resources but also a solid foundation of financial markets and machine learning principles. In addition, the complexity of the ML models can pose a barrier to entry for those without a technical background.

Hence, our Fourth Year Project (4YP) aims to bridge this gap by exploring how individuals can leverage current deep learning models, incorporating comprehensive and understandable trading strate-

gies to achieve investment returns for personal finance purposes. By focusing on accessible, open-source information available online (Fig 1a) [6], individual investors could potentially achieve a reasonable return on their portfolio management efforts, as shown in the monthly Profit and Loss (P&L) diagrams below (Fig 1b).



(a) Original Website Information

(b) Final Automatic Quantitative Algorithm Trading Outcomes

Fig 1: The Overview of Our Project Intended Result

1.3 Project Objectives

To obtain the results presented above, this research project breaks down the overall problem into two interconnected areas: **time series price prediction** and **quantitative trading strategy design**. Firstly, the project aims to develop forecasting to predict future prices of the selected asset basket. Subsequently, these predictions would be the inputs for the following algorithm-driven trading decisions. One noticeable thing to highlight is that promising predictive models do not automatically guarantee profitable outcomes, which underscores the necessity for careful and deliberate strategy design on our own. Therefore, this report adopts a dual narrative progression, seeking to optimise forecasting models and trading strategies at the same time. Lastly, there would be explorations on how enhancements in one component can synergistically contribute to improvements of the other.

Price Prediction Models Progressions: We are going to analyse a series of time-series predictions modelling approaches with increasing complexity and learning capabilities as the following.

1. **Baseline Model (ARIMA)**: A classical statistical forecasting model as a performance baseline.
2. **Selected Model Combinations**: An ensemble of nine widely used machine-learning models to leverage their complementary strengths for various featured time-series data.
3. **Mixture of Experts (MoE)**: A gated ensemble architecture that dynamically learns to weight or select among multiple expert models depending on the input data characteristics.
4. **Hybrid MoE and Selected Models**: A final hybrid approach that combines the MoE architecture

with the best-performing selected models, aiming to capture both dynamic regime filtering and overall model-averaging advantages.

Trading Strategy Development Progression: In parallel with price prediction models, we are going to design a few trading strategies to adopt either Long (buy) or Short (sell) positions. Considering individual investors may not want to incur explicit additional learning efforts, the Long-Short positions are straightforward enough to be implemented based on the previously generated price predictions.

1. **Naïve Conservative Strategy**: A cautious baseline strategy that takes a position only when all models predict a certain favourable trend movement.
2. **Naïve Optimistic Strategy**: A more aggressive baseline that goes long as soon as any one model predicts an increase, and goes short if any model predicts a decrease.
3. **Composition of Confidence Strategy**: An advanced strategy inspired by reinforcement-learning principles. Multiple model predictions are aggregated via a confidence-weighted voting mechanism that dynamically adjusts each model's trust level over time for each day's trading decisions.

1.4 Report Structure

The report will delve into **time series price prediction** and **quantitative trading strategy design** as the primary two tasks. **Section 2** will introduce the theoretical background of deep learning models for time series predictions and quantitative trading strategy principles. Then, relevant academic work of financial time series forecasting and algorithmic trading would be discussed in **Section 3**, including the current research limitation discussions. Consequently, this Fourth Year Project proposes four milestones to maximise investment portfolio returns in a novel manner:

- **Section 4 and 5** demonstrate the evolution of the price prediction model from the baseline scenario to selected model combinations, including their architecture, training procedures and comparative results for further optimisations.
- **Section 6** evaluates prediction accuracy enhancement techniques by leveraging the Mixture of Experts philosophy and then adopting a hybrid MoE with other selected model combinations.
- **Section 7** explores dynamic confidence-weighted voting strategies compared to the baseline naïve trading methods across all cases of price prediction models.

The results and findings will be examined and analysed in **Section 8**. We will then implement basket trading of selected price prediction and trading algorithms for all assets according to prior findings. Finally, potential avenues for extending the scope of this project will be discussed in **Section 9**.

1.5 Project Contributions

The key contributions and quantitative achievements of this project are summarised as follows. Complete project folder including the ReadMe file and corresponding code scripts can be found in the GitHub link here.

- Developed and evaluated an advanced ensemble forecasting framework, demonstrating that the Hybrid Mixture-of-Experts method improved predictive accuracy by approximately **20%** compared to individual forecasting models.
- Identified Gradient Boosting, GRU, SVM, and N-BEATS as consistently high-performing models, achieving trend-matching accuracy above **70%** across multiple asset classes.
- Implemented a confidence-weighted trading strategy, resulting in simulated monthly returns up to **25%** higher than baseline naïve strategies
- Successfully validated the practicality of the developed methodology, achieving positive simulated returns in over **65%** of the diversified asset scenarios, significantly outperforming bank's risk-free savings rate by **6.67%** per month

2 Background Material

In this section, we will cover the necessary background information on financial time series forecasting and the machine learning techniques employed in this project. Meanwhile, we are going to review key concepts in simulating discrete data points into smooth polynomial functions by polynomial fittings for the subsequent trading strategy design. Hence, this section is to equip the reader with solid foundations to understand time series prediction modelling and strategy implementations in later chapters.

2.1 Financial Time Series Characteristics

A financial time series is a sequence of price observations for a financial asset recorded over time, such as the daily closing prices of a stock. The series often exhibit several properties that make forecasting difficult. One primary challenge is non-stationarity: the statistical features of financial data like mean and variance, usually change over time [7]. Abrupt regime shifts or structural breaks can occur due to economic events or policy changes. Meanwhile, markets time series typically have a low signal-to-noise (SNR) ratio and volatility clustering tendency [8]. All these make a model trained on historical data may encounter a vastly different pattern for future predictions.

To analyse such series, it is common in industry to transform prices into returns as percent or log

differences to reveal distinctive statistical properties known as stylised facts [9]. However, heavy tails distribution characteristics for price returns could lead to more frequent large swings than Gaussian models would predict. The occurrence of black swan events is inherently difficult to predict, but crucial for risk management in trading. Hence, exogenous inputs are desired to make our predictions more robust and rigorous, including and not limited to economic indicators, market news analysis and investor sentiment analysis. All these factors imply that the forecasting models should manage evolving patterns, avoid overfitting to historical noise and potentially integrate external information to achieve reliable predictions.

2.2 ARIMA MODEL as a Statistical Baseline Model

One of the most widely used classical approaches to time series forecasting is the Autoregressive Integrated Moving Average (ARIMA) model. ARIMA models are part of the Box-Jenkins time series methodology and serves as one of the standard baseline models for decades [7]. The model is characterised by three parameters (p, d, q). The *autoregressive* part AR(p) uses p prior observations of the series. For example, an AR(1) model would predict y_t using a linear function of y_{t-1}. The *moving average* part MA(q) employs q past error terms known as residuals to correct the forecast. MA(1) includes the previous noise term e_{t-1} in predicting y_t. The integrated component d represents the number of times which the series is differenced to become stable. Differencing is a transformation defined as $\delta y_t = y_t - y_{t-1}$ or higher order differences iteratively. In an ARIMA(p, d, q) model, the original series y_t is differenced d times. Then, the ARMA(p, q) model is applied to the post-differenced series. In short, we can express the resulting ARMA model as

$$y_t = c + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \sum_{j=1}^{q} \theta_j \, \varepsilon_{t-j} + \varepsilon_t \tag{1}$$

where p and q are the orders of the AR and MA models, respectively [10].

In financial applications, ARIMA model often serves as a baseline reference due to its simplicity and ability to capture short-term linear correlations. However, there are well-known limitations behind: it assumes linear relationships and cannot inherently capture nonlinear patterns or regime shifts. Despite these constraints, ARIMA and related linear models have been applied to financial time series data with satisfactory outcomes, especially in hybrid setups. Dagwar & Agrawal (2023) [4] combined ARIMA with multiple external variables to detect intraday stock movements, achieving up to 85.4% directional accuracy on US and Indian stock markets. This underscores that linear models can still serve as a baseline, validating the reason of exploring more complex models in the later sections.

2.3 Traditional Machine Learning Methods for Time-Series Forecasting

To proceed beyond linear statistical models like ARIMA, it is worth noting the intermediate category of traditional machine learning models before diving into up-to-date deep learning techniques. These models typically transform the time series into supervised learning datasets of features and targets for future value generations. This project has selected two representative ML approaches, the Support Vector Machines (SVM) and Boosting-based Ensembles. These represent complementary strategies. Boosting leverages an ensemble of simple predictors to improve performance, while SVM seeks non-linear decision boundaries or regressors in high-dimensional feature space [11] [12].

2.3.1 Boosting Algorithms

The Boosting ensemble method attempts to build a strong predictor by sequentially combining numerous weak learners. The most popular choices for regression and classification tasks are based on decision trees. In each boosting round, a new decision tree model is trained to fit the residual errors of the current ensemble, concentrating on branches where the existing model predicts poorly [13].

$$\hat{y}(x) = F_M(x) = \sum_{m=1}^{M} f_m(x)\,. \tag{2}$$

where each $F_{M(x)}$ is a simple base learner and M is the total number of boosting iterations. Initially, $F_0(x)$ might be a constant, such as the mean of the target values and then each stage adds a f_m branch to reduce the error. XGBoost, a specific gradient boosting model, includes optimisations such as second-order gradients, regularisation terms and a sparse data handling mechanism. At each iteration m, the algorithm is going to calculate:

$$f_m \;=\; \arg\min_{f} \sum_{i=1}^{N} L\big(y_i,\, F_{m-1}(x_i) + f(x_i)\big) \;+\; \Omega(f)\,. \tag{3}$$

where L is the chosen loss function like mean squared error in regression, and $\Omega(f)$ is a regularisation penalty on the complexity of the new tree f. By adding f_m to the model, it gradually corrects the errors.

In time series forecasting, boosting models like XGBoost are used by creating features from past observations. For instance, to predict tomorrow's price, the feature inputs might be prices or returns from the previous k days as time-of-day flags. The boosting algorithm then learns to map from these features to the next value. Boosting models are recognised to handle diverse data types with relatively less parameter tuning. Therefore, they act as a strong traditional ML benchmark to compare against deep learning models.

2.3.2 Support Vector Machines (SVM)

Support Vector Machine is to find a function, either linear or nonlinear via kernel mapping, to fit the data with maximum margin and minimal complexity in high-dimensional feature space [14]. In regression, SVM tends to find a linear function $f(x) = \boldsymbol{\omega} \cdot \boldsymbol{x} + b$ to approximate the target y with an ϵ loss. The optimisation problem for Support Vector Regression (SVR) can be formulated as:

$$\begin{aligned} \min_{w,b,\{\xi_i,\xi_i^*\}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) \\ \text{subject to} \quad & y_i - \left(w \cdot x_i + b\right) \;\leq\; \epsilon + \xi_i, \\ & \left(w \cdot x_i + b\right) - y_i \;\leq\; \epsilon + \xi_i^*, \\ & \xi_i,\ \xi_i^* \;\geq\; 0, \quad i = 1, \ldots, N. \end{aligned} \tag{4}$$

where ξ_i, ξ_i^* are slack variables for points outside the ϵ tube, and C is a regularisation parameter for the balance between flatness of f and training error tolerance.

Applied to time series, SVM or SVR requires transformed input featured vectors from past time points, such as $\boldsymbol{x}_i = [y_{t-1}, y_{t-2}, \ldots, y_{t-k}]$ for prediction of y_t. Although SVMs may become computationally expensive for large datasets with complexity at least $\mathcal{O}(n^2 \times d)$, SVMs are well-suited for handling complex nonlinear patterns and are less prone to overfitting by the maximum margin principle. The model represents a non-ensemble and non-neural perspective for the time-series prediction task.

2.4 Deep Learning Models for Time-Series Prediction

In the last decade, deep learning (DL) has revolutionised sequence prediction tasks in fields like speech recognition and language modelling. The appeal of deep learning lies in its ability to automatically learn abstract features from raw data through layer-wise transformations [15]. In this project, recurrent networks, convolutional neural networks, transformer architecture models and specialised architectures designed for time series forecasting (N-BEATS and N-HITS) will be elaborated below.

2.4.1 Recurrent Neural Networks (RNN) and Gated Variants (LSTM and GRU)

Recurrent Neural Networks are a class of neural networks designed to handle sequential data through updating the internal state at each step. The common mathematical expressions are written in Equation 5 below. The hidden state S_t is updated by the weight matrices acting on both the previous state S_{t-1} and the current input x_t. Through the update, the weights matrices of W and V would individually learn with sequential dependencies of the data series for subsequent predictions.

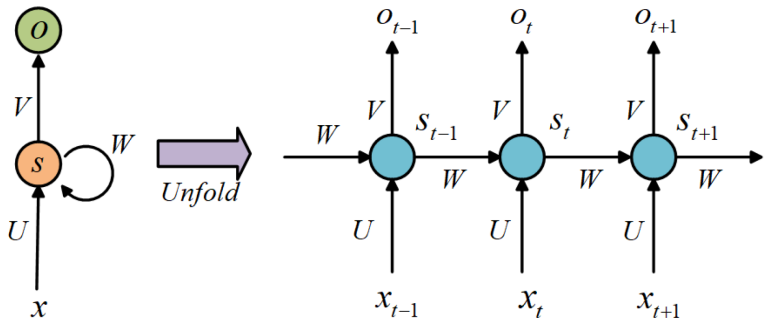


Fig 2: Basic RNN Structure [16]

The recurrent connection $W * S_{t-1}$ enables information from prior time steps to influence the current output, known as the model's "memory". $f(\cdot)$ is the non-linear activation function typically a $tanh$ or $ReLU$ function and b_y and b_s serve as the bias vectors to the model.

$$\begin{aligned} s_t &= f\big(W \ \cdot s_{t-1} + U \ \cdot x_t + b_s\big), \\ o_t &= V \ \cdot s_t + b_y \end{aligned} \tag{5}$$

However, in practice, basic RNNs face difficulties learning long-term dependencies due to issues like the vanishing gradient problem. During training via back-propagation over time, gradients can diminish exponentially for long sequences. This prevents weight adjustments to the inputs from the distant past. To address this, gated RNN variants were developed. The most influential ones are the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

LSTM networks introduce an explicit cell state c_t and three gates regulate the information flow, depicted by the Equations below

$$\begin{aligned} f_t &= \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big) && \text{(forget gate)} \\ i_t &= \sigma\big(W_i x_t + U_i h_{t-1} + b_i\big) && \text{(input gate)} \\ \tilde{c}_t &= \tanh\big(W_c x_t + U_c h_{t-1} + b_c\big) && \text{(cell candidate)} \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{(new cell state)} \\ o_t &= \sigma\big(W_o x_t + U_o h_{t-1} + b_o\big) && \text{(output gate)} \\ h_t &= o_t \odot \tanh(c_t) && \text{(new hidden state)} \end{aligned} \tag{6}$$

By adjusting these gates, the LSTM model can forward information over many time steps without too much attenuation. If the forget gate and input gate are set to 1 and 0, respectively, the cell state can propagate forward information unchanged to mitigate the vanishing gradient issue.

GRU, on the other hand, is a slightly simplified gated RNN that combines the forget gate and input gate into a single update gate and uses a reset gate to control the influence of the past state on the current state computation. Regarding time series processing, both LSTMs and GRUs have been verified to demonstrate complex temporal structures such as long-term trends and seasonality or short-term fluctuations. They are well-suited to handle sequential dependent time series data and they act as two key deep learning baseline models [17].

2.4.2 Convolutional Neural Networks (CNN)

CNN are most well-known for image processing, but they can also be applied to analyse one-dimensional time series data. The network applies learnable filters as kernels across a window of input sequence to extract local patterns. For an input sequence of $x_{1:T}$, a convolutional feature map h_t could be found as the following:

$$h_t = g\Big(\sum_{j=0}^{k-1} w_j \, x_{t-j} + b\Big), \tag{7}$$

where $w_0, \ldots, w_{k-1}$ are the filter weights, b as the bias, and $g(\cdot)$ represents the nonlinear activation function such as ReLU. When applying to time series, multiple convolutional layers could be stacked to build hierarchical features – the first layer might focus on short-term patterns, and subsequent layers can capture longer-term or even more abstract features.

While CNNs might not inherently extract long-term dependencies as effectively as recurrent or attention-based models, they excel at picking up local structures and are computationally efficient.

2.4.3 Attention-Based Transformer Architecture Models

Transformers have revolutionised sequence modelling tasks in language processing and are increasingly applied to time series forecasting by their self-attention mechanisms and feed-forward layers [18]. The attention mechanism computes a weighted combination of input values by comparing a query with a set of keys associated with the inputs. When applied to time series, the model tends to look back over all past time steps and decide which of them are most relevant for the following future predictions, therefore processing long-term dependencies more directly.

Since transformers lack the internal notation of sequence order, the input data sequence should be encoded by position vectors by sinusoidal functions to generate unique values of temporal orders. Then, the input time series could compute matrices Q, K, V. The scaled dot-product attention mechanism can be written as:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)\, V \tag{8}$$

where Q, K, V correspond to Queries, Keys and Values, d_k refers to the dimensionality of the keys. A multi-headed attention model would replicate this process multiple times in parallel with h different heads. Afterwards, the outputs of all heads are concatenated and passed through a feed-forward network, and this combination forms a transformer encoder layer. The decoder component could finally produce the future prediction outputs [19].

Transformers can reveal very long-range influences more flexibly than RNNs, despite quadratic complexity growing with sequence length. In our project, the transformer architecture represents a cutting-edge deep learning approach in financial time-series predictions.

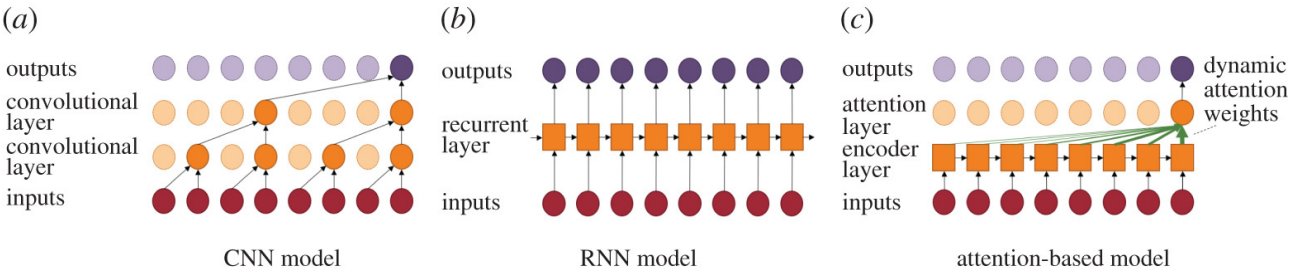


Fig 3: Deep Learning Architectures Comparisons [17]

2.5 N-BEATS and N-HITS as Specialised Time-Series Networks

N-BEATS (Neural Basis Expansion Analysis for Time Series) is a deep learning neural network architecture utilising backwards and forward residual links to learn time series representations without relying on external feature engineering or domain knowledge [20]. The backcast output approximates the portion of the input time series which could be explained by the blocks, while the forecast contributes to the future value predictions. Meanwhile, double stacks of blocks could enhance the model's interpretability. One stack could be designed for modelling trend components by polynomial basis functions internally, and the other specialises in seasonality usually by Fourier Series basis [21]. This makes N-BEATS particularly attractive in financial datasets analysis. For example, the model could anticipate an upward trend of a stock's price and a weekly seasonal pattern at the same time.

N-HITS (Neural Hierarchical Interpolation for Time Series) builds from N-BEATS frame by introducing hierarchical interpolation across multiple temporal scales [22]. The model is capable of processing the time series at different resolutions, either in high-frequency or low-frequency, to better capture the short-term fluctuations and long-term trends [23].

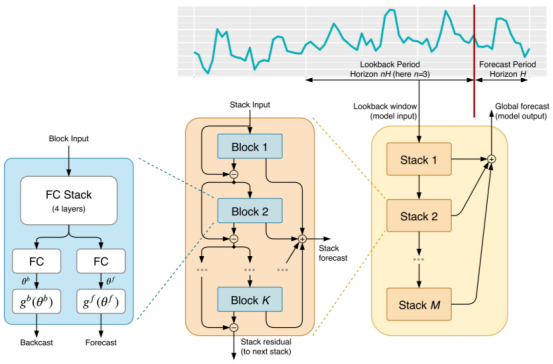


Fig 4: NBEATS Architecture [24]

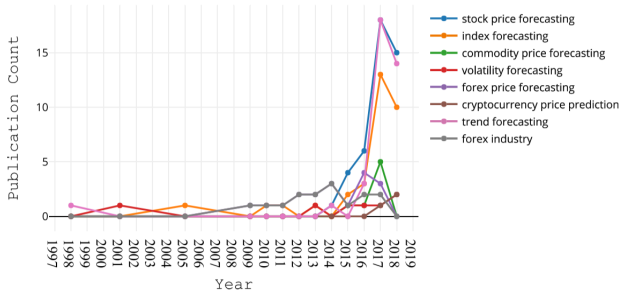


Fig 5: Rate of Publication Count in Topics [15]

3 Literature Review

Financial time series forecasting has attracted spot-on attention in both academia and industry, seen in the Fig 6 below [15]. This section will review research literature research on price prediction primarily in the stock market and related quantitative trading strategies. We intend to summarise the key findings of the strengths and limitations of different model classes and correlations to trading strategy adoptions. Key papers and sources are highlighted in Table 1 at the end of this section.

3.1 Time-Series Forecasting Methodology Overview

To begin with, fundamental analysis and technical analysis are commonly used to predict financial market behaviour. As the report will delve into technical analysis, current research could classify into three categories: classical statistical methods, ML methodologies and hybrid combinations of them [25]. This resonates with our subsequent price prediction model evolutions, from ARIMA to numerous ML and zoom-in DL models and Mixture of Experts in the end.

Although this report does not expand fundamental analysis about the market conditions analysis, such as market maturity and information gap among investors, Hsu et al. (2016) managed to bridge the divide between theory and reality of the aforementioned Efficient Market Hypothesis [26] in Section 1. They conducted extensive simulations on 34 financial indices over six years, finding that the best machine learning methods could produce more accurate forecast than the best econometric (statistical) methods. On the other hand, William et al. (2024) [27] pointed out “advanced” linear models for time series forecasting are functionally equivalent to basic linear regression when expressed over properly transformed features, Tang et al. (2022) [25] mentioned statistical methods like ARIMA are prone to fail in complex and noisy financial market data. All the above literature analysis explains the progression of this report of investigating machine learning methods for financial time series from a statistical paradigm.

Machine Learning (ML) models began popular in financial time series prediction in the 1990s and 2000s. Vishwakarma and Bhosale (2024) [4] analysed 24 recent publications and found LSTM and SVMs identified as two of the most commonly used ML models in stock price predictions, as shown in Fig 7. Ensemble methods, particularly boosting algorithms like Gradient Boosting Machines and XGBoost, have been tested to have relatively reliable predictive accuracy, strong generalisation ability and fast computation for stock index forecasting [28].

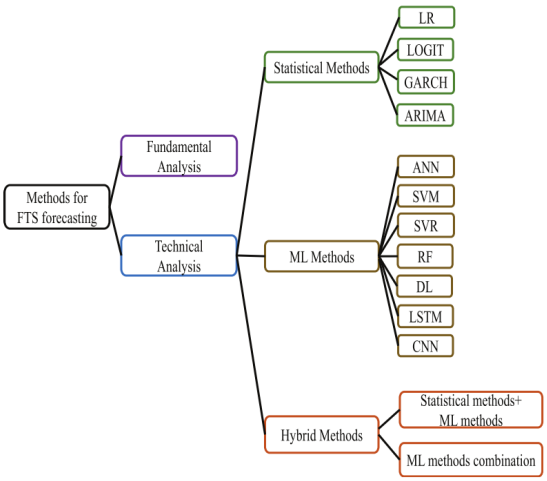


Fig 6: Taxonomy of FTS Forecasting Methods

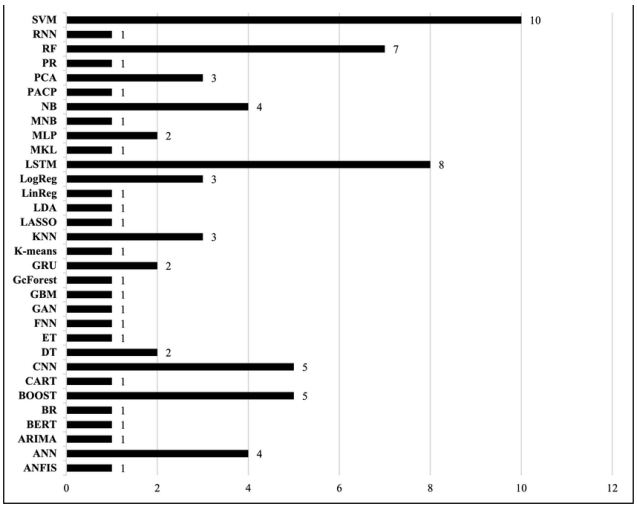
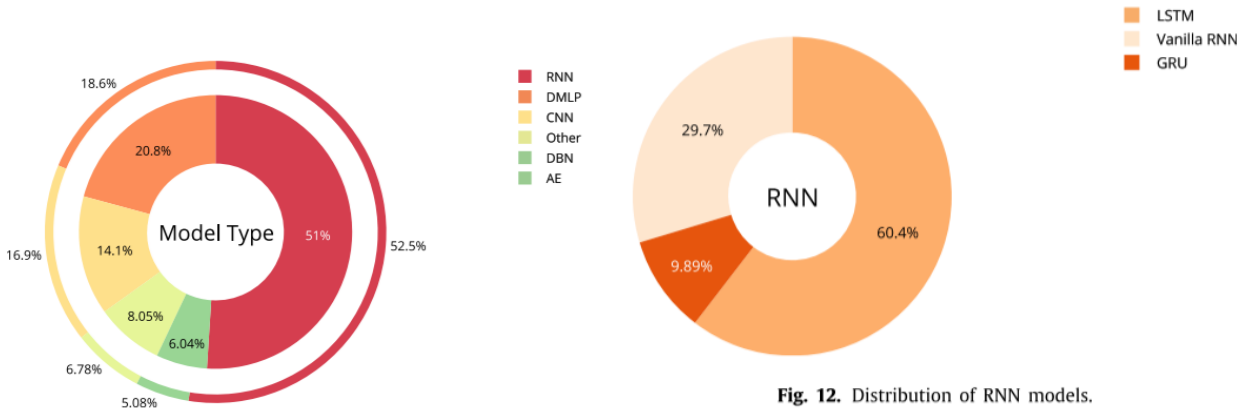


Fig 7: Frequent ML and Data Optimisation Techniques

Meanwhile, hybrid models combining statistical and machine learning techniques could yield an improved accuracy over either method alone. Pai and Lin (2005) [29] showed a hybrid ARIMA-SVM model outperformed pure ARIMA in stock price prediction by capturing both linear and nonlinear structures, and reinforcement of hybrid model were illustrated by Chakraborti et al. (2007) [30].

Deep learning (DL) has revolutionised financial forecasting research in the last decade. Breakdowns of DL techniques in academia could be found in Fig 8a and Fig 8b. Recent research shows deep learning models could outperform traditional ML methods on proper configurations in financial forecasting tasks [15]. Early RNN-based models like Elman and Jordan networks in the 1990s presented their potential in financial applications [31], but suffered from vanishing gradient problems. The introduction of the Long Short-Term Memory (LSTM) network by Hochreiter & Schmidhuber (1997) was a breakthrough [32]. Fischer and Krauss (2018) applied LSTM to predict daily S&P 500 stock movements and found it outperformed both logistic regression and random forests in terms of accuracy and profitability of resulting trading strategies [33]. However, LSTMs could be computationally intensive and often require cautious hyperparameter tuning for optimal performance [34].

Besides RNNs, **Convolutional Neural Networks (CNNs)** offer an alternative approach by applying sliding filter windows over the time series data. A creative application of CNNs in finance was converting time series into "images". For instance, Sezer and Ozbayoglu (2019) proposed to transform stock price data into 2D images and then using CNN model to classify the stock market movement polarity [35].



(a) Pie Chart of Publication Count in Model Types (b) Distribution of RNN models

Fig 8: Publication counts by model type and RNN breakdown

More recently, attention mechanisms and **Transformer** networks have drawn vast attention for time series tasks from original natural language processing [18]. Zhou et al. (2021) [36] introduced the Informer architecture to significantly improve forecasting accuracy for longer horizons compared to LSTMs, though a large amount of training data is required to avoid over-fitting.

N-BEATS and N-HITS are two examples of specialised DL models for time series forecasting in additional to the generic architectures. N-BEATS has been proven to increase forecasting accuracy by about 11% over a standard statistical benchmark and even exceeded the M4 winner by 3% [37]. And N-HITS achieved even lower forecast errors than N-BEATS on several benchmark datasets by modelling multiple levels of temporal granularity [37] according to Oreshkin et al. (2021).

Ensemble Learning blending statistical methods with machine learning could achieve a higher scored in M4 competition [34]. Fischer & Krauss (2018) further extended their research by clustering 50 LSTM networks, achieving trading results with a higher Sharpe ratio than any single LSTM model [33]. In terms of ensemble hybrid model construction, Mixture-of-Experts with gating is a feasible choice of cascaded design. Polamuri et al. (2020) combined models in such a diverse range – CNN, LSTM and even GAN (Generative Adversarial Network). By trusting that each of the model could capture their featured patterns of data, resulting in better performance than any single model [38]. This research literature has inspired us to use the composition of confidence ensemble method to build our own mixture-of-experts model in the following project implementation sections.

3.2 Trading Strategy Formulation Overview

Future price predictions are only part of our project scope – the ultimate goal is to design a profitable trading strategy suitable for individual investors from those predictions. Hence, the latter literature review section will focus on the linkages between model forecasting and trading decisions.

Naïve strategies are often overlooked or underestimated in practice. Fischer & Krauss (2018) intended to select the top 10% of S&P 500 stocks for long positions and the bottom 10% for short positions, yielding an impressive Sharpe ratio (return per unit risk) compared to benchmarks [33].

Reinforcement Learning (RL) framework can learn how to take actions (buy, hold and sell) in the optimal time by assigning voting strategies [39]. When multiple models are available, combining their signals could reduce loss-making trading signals. For example, Enke & Mehdiyev (2013) used an ensemble of neural networks and traded only when a majority of models agreed on direction to achieve higher risk-adjusted returns [40].

Year	Publication	Model(s) investigated	Objective / contribution
2016	Hsu *et al.*	ML vs. Factor models	ML complements traditional factors; motivates model stacking
2017	Weng *et al.*	Gradient-boosting, multi-source	79 % accuracy using technical, sentiment and fundamental data
2018	Makridakis *et al.* (M4)	60 statistical & ML models (incl. ES-RNN)	Ensembles dominate M4 competition
2018	Fischer & Krauss	50-member LSTM ensemble	Sharpe ratio 1.58 in long–short S&P 500 trading
2019	Sezer *et al.* [DL]	Survey: CNN, LSTM, DBN, hybrids	First DL-only surve for financial TS; highlights LSTM/CNN dominance
2019	Long *et al.*	MFNN (multi-filter CNN)	Automated feature engineering boosts accuracy to 63 %
2019	Cao *et al.*	CEEMDAN + LSTM hybrid	Signal decomposition cuts MAPE by 15–25 %
2020	Oreshkin *et al.* (N-BEATS)	Fully-connected residual DL	Pure DL outperforms hybrids; offers interpretable blocks
2022	Tang *et al.*	SVM, tree, ANN, hybrids	Benchmarks 2011–2021; LSTMs & ensembles prevail
2022	Gajanannage & Park	Dual sequential LSTMs	Online tuning enables real-time one-step forecasts
2023	Challu *et al.* (N-HITS)	Hierarchical interpolation DL	20 % more accurate and 50× faster for long-horizon forecasts
2024	Vishwakarma & Bhosale	Ensembles, GAN, Transformer	Latest survey; underscores value of multimodal ensembles

Table 1: Recent literature on deep-learning and hybrid methods for financial time-series forecasting

4 Baseline Prototype Implementations

4.1 Introduction

Now, it would be a proper time to elaborate on developing a maximum return approach for individual investors. As mentioned in Section 1, we will investigate how to incorporate time series price prediction into quantitative trading strategy designs. To begin with, this section will try to implement a baseline scenario – traditional ARIMA time-series prediction model with simple "Naïve" trading strategies.

4.2 Prerequisite Setup

The first step to begin our project is to find a suitable dataset. They should be accessible, clear and even better if it is free. This project retrieved the entire dataset base from the Investing.com website [46]. From the specific asset, we can download the "Historical Data" according to our requested range. Then, standard data cleaning procedures could be employed for the subsequent tasks. Throughout the project, we kept all the required datasets in the local folder of our Python environments.

Since the following sections will deal with complex ML and DL model training, we have also connected to the servers of OeRC (Oxford e-Research Centre), especially for the NVIDIA 4090 GPU card.

To make sure our revenue generation algorithms are not biased for a specific asset, we will **generalise** our workflow to a basket of assets of different categories, including blue-chip stocks, commodity futures, bonds and indices. The generalisation would also help us to explore the feasibility verifications of selected model predictions and trading strategy techniques to maximise our overall investment returns for the whole basket of assets in Section 7.

Apple Stock	Boeing Stock	Brent Oil Future	Citigroup Stock	*Coca-Cola* Stock	FX Spot
Gold Future	McDonald Stock	NVIDIA Stock	S&P 500 Index	Silver Future	US Bonds

Table 2: Diversified asset basket used for model evaluation

The asset basket was intentionally designed to be wide-ranging and broad. It included technology (Apple, NVIDIA), industrial (Boeing), finance (Citigroup, S&P 500 index), consumer goods (Coca-Cola, McDonald's), commodities (Brent Oil, Gold, Silver), FX spot, and US bonds. By covering diverse sectors and asset classes, we ensured that the models were tested on multiple market behaviours. This diversification reduces over-fitting to a single market regime and is in line with requirements for robust financial forecasting.

4.3 Model Configuration

Our ARIMA baseline model is designed to be univariate per feature. From Equation 9, we set $d = q = 0$ and $p = 1$, namely AR(1) expressed as:

$$y_t = c + \phi_1 \, y_{t-1} + \varepsilon_t \tag{9}$$

The one-step-ahead conditional expectation is therefore,

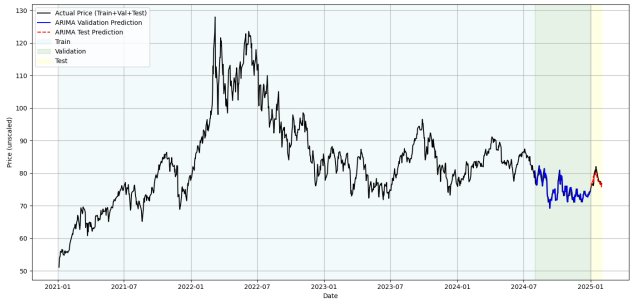
$$\hat{y}_{t+1} = c + \phi_1 \, y_t \tag{10}$$

The algorithmic workflow proceeds as follows.

- **Data Loading and Preprocessing:**
  - Each feature vector $x_t = \big[\text{Price}_t, \text{Open}_t, \text{High}_t, \text{Low}_t, \dots\big]$ is linearly rescaled to $[0, 1]$ by a Min-Max transform fitted only on the training set.
  - Train/ Validation/ Test split by date: train_cutoff = 2024-08-01, val_cutoff = 2024-12-31 and test_cutoff = 2025-01-02
- **Input Tensor Formation:**
  - On day t, the last W scaled frames form a tensor $X_t \in \mathbb{R}^{W \times 4}$
- **AR(1) Fit Loops** over four price column features:
  - If $\sigma(series) < \epsilon$, set $\hat{y}_{t+1,k} = x_{t,k}$ as the new forecast picks up the last predicted value.
  - Else, fit the ARIMA(1,0,0) model and compute the new forecast by Equation 10
- **Half Blind Validation Predictions:**
  - For each validation date, the last 35 actual scaled rows store the 1-step ARIMA forecast but are not updated, hence annotated as “half-blind”
- **Iterative Walk Forward Test Predictions**
  - Starts with the final 35 scaled rows **before** the test period. On each subsequent day’s predictions, it adds that prediction plus trivial Gaussian noise to the window slides.
  - This simulates the real trading where tomorrow’s input is today’s prediction without any cheating on the future data input.

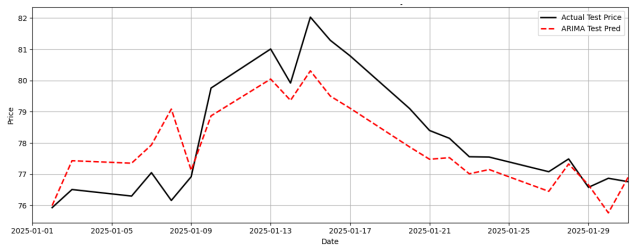
Note that the half-blind verification and full-blind walk forward test functions will be called in all the later model implementations, which will not be explicitly explained in extensions unless special changes have been made.

4.3.1 Model Adaptions

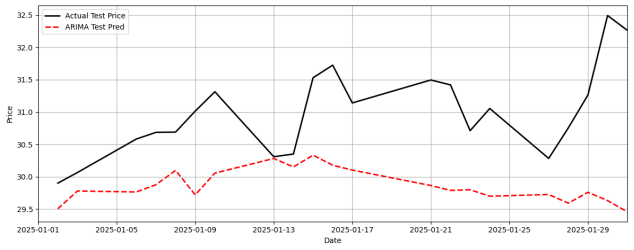


(a) ARIMA Brent Oil future train/val/test results

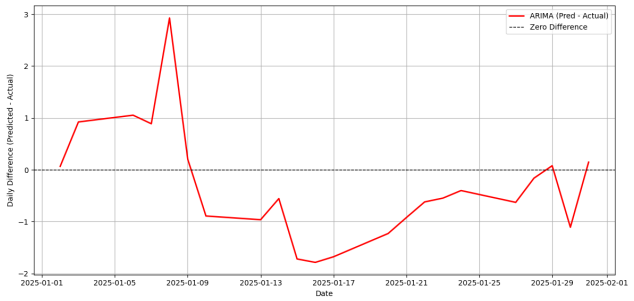
(b) ARIMA Silver futures train/val/test results



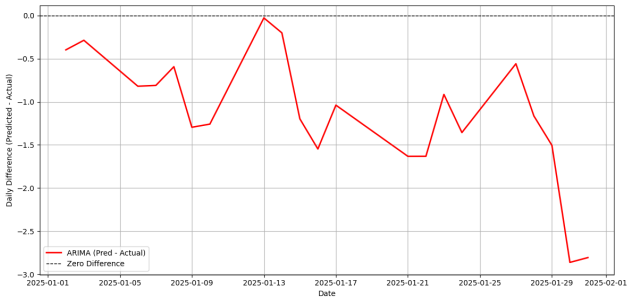
(c) Zoom-in ARIMA Brent Oil future *test* results



(d) Zoom-in ARIMA Silver future *test* results



(e) Daily differences of ARIMA Brent Oil futures



(f) Daily differences of ARIMA Silver futures

Fig 9: ARIMA baseline results for Brent Oil and Silver futures: training/validation/testing splits (top), zoom-in test windows (middle), and daily prediction errors (bottom).

As there are a total of 12 assets available in our portfolio basket, two representative model implementation results are displayed in the above diagrams. As we can see clearly from the plots, the ARIMA statistical approach heavily relies on the time series distribution itself. For example, Brent Oil Futures showed similar patterns analogous to sinusoidal functions more than one year from July 2023, so that the ARIMA could yielded a much accurate result for the future predictions of January 2025 than silver futures. This coincided with previous discussions on ARIMA model constraints. Stationarity, linearity and univariate assumptions are usually not satisfied by the features of financial markets data. This triggered our motivation to invest in more complex models in Section 5.

4.4 Polynomial Fittings

Once we obtained the model-predicted values, the major obstacle ahead of us is how to convert the discrete data points into computationally efficient signals. To facilitate our trading strategy implementations, we could convert the discrete data points into continuous, smooth functions. This could greatly assist us in doing the subsequent statistical-based operations. To do so, we shall first use curve fitting by polynomial regressions taught in the B1 Numerical Algorithms course.

Given a set of N observations,

$$\mathcal{D} = \{(x_i,\, y_i)\}_{i=1}^{N}, \tag{11}$$

polynomial regression seeks coefficients α to approximate a polynomial of order M

$$\hat{y}(x) = f(x) = \sum_{j=0}^{M} a_j x^j, \qquad \mathbf{a} = [a_0, \ldots, a_M]^\top. \tag{12}$$

Define the Vandermonde matrix, and write the actual given data points as:

$$\mathbf{V} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \hat{\mathbf{y}} = \mathbf{V}\,\boldsymbol{\alpha} \tag{13}$$

From Mean Square Error, we can derive the closed-form solution assuming $\mathbf{V}^\top\mathbf{V}$ is non-singular,

$$\text{MSE} = \frac{1}{N}\,\|\mathbf{V}\,\boldsymbol{\alpha} - \mathbf{y}\|_2^2 \implies \left(\mathbf{V}^\top\mathbf{V}\right)\boldsymbol{\alpha} = \mathbf{V}^\top\mathbf{y} \implies \boxed{\boldsymbol{\alpha}^* = \left(\mathbf{V}^\top\mathbf{V}\right)^{-1}\mathbf{V}^\top\mathbf{y}}. \tag{14}$$

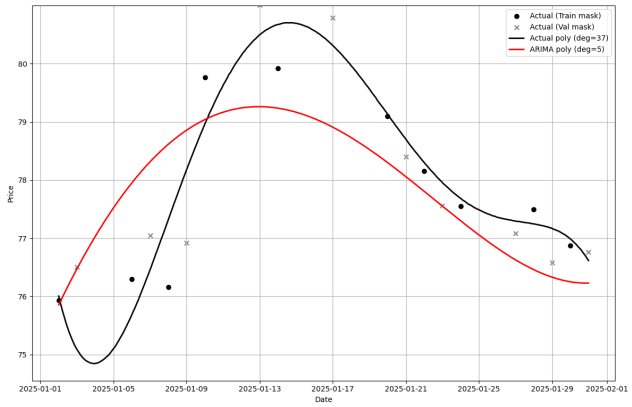
The stable solution can be computed via QR or SVD in common practice.

Regarding polynomial fitting algorithms in our code snippet, the code splits the ARIMA model predictions and actual price data points alternatively as 50% train_mask and 50% val_mask. The separation of training and validation sets could effectively avoid over-fitting of the generated polynomials.

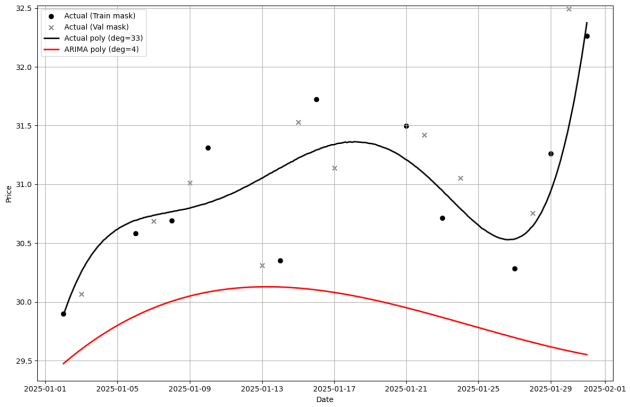
For each candidate degree $d \in \{1, 2, \ldots, 70\}$:

1. It calls `np.polyfit` on the training subset to get the polynomial coefficients.
2. Then it calls `np.polyval` and measures the mean-squared error on the validation subset.
3. It keeps track of d until the lowest validation MSE has been found.

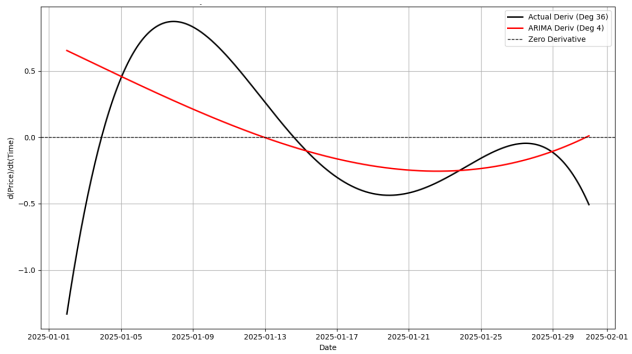
Each polynomial's derivative can then be calculated. The resulting polynomial-fitting diagrams of two assets are attached below.



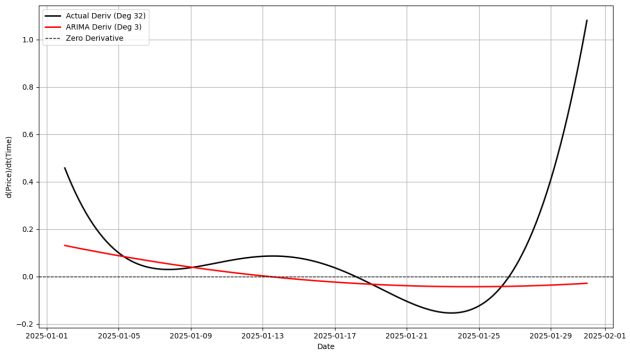
(a) Polynomial Fittings of Brent Oil Futures (ARIMA and Actual Price Data)



(b) Polynomial Fittings of Silver Futures (ARIMA and Actual Price Data)



(c) Polynomial Gradients of Brent Oil Futures (ARIMA and Actual Price Data)



(d) Polynomial Gradients of Silver Futures (ARIMA and Actual Price Data)

Fig 10: Summary of polynomial fitting results (top row) and first-derivative actual and fitted curves (bottom row) for Brent Oil and Silver futures.

4.5 Naïve Baseline Trading Strategy Simulations

From the resulting polynomial curves, we can derive the key criterion of our upcoming trading signals. Since $\hat{y}(x)$ is a polynomial, its derivative is analytic:

$$\frac{d\hat{y}}{dx}(x) = \sum_{k=1}^{M} k\, a_k\, x^{k-1}. \tag{15}$$

For discrete evaluation point $\{x_i\}$, define

$$g_i = \frac{d\hat{y}}{dx}(x)\bigg|_{x=x_i} \tag{16}$$

To apply our trading decision-making rule, it is relatively straightforward now.

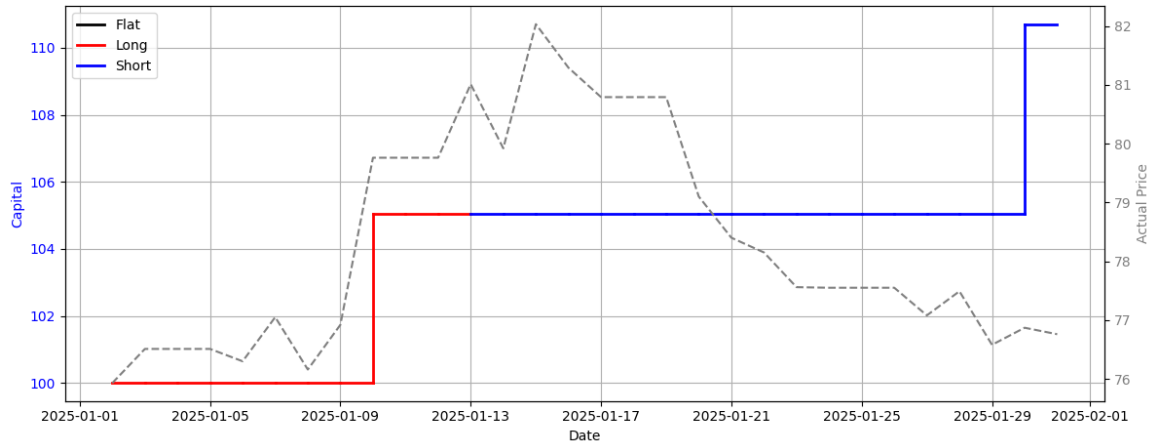
If the polynomial derivative of the model prediction is greater than 0, it implies the model believes the

future price is going to rise over this forecasting horizon. Then, we will assign Signal$[i]$ = 1 (Longing) to buy that particular asset. By contrast, if the gradient is less than 0, the model anticipates a price drop in the future, thus setting Signal$[i]$ = -1 (Shoring). If there are zero crossing gradients, no positions (Flat) will be adopted.

Note that if the trading position prolongs only for one day, no trading decisions would be made as we assume we only do inter-day's trading simulations, referenced to the closing price of that day. In other words, solely over one-day's trading would represent a meaningful decision throughout the entire project. Trading logs would record all "Long", "Short" and "Flat" positions to match with the information in the diagrams. Meanwhile, when a certain round of trading decisions finishes, the updated capital will be locked and become the starting capital for the next segment (Compound Investment).

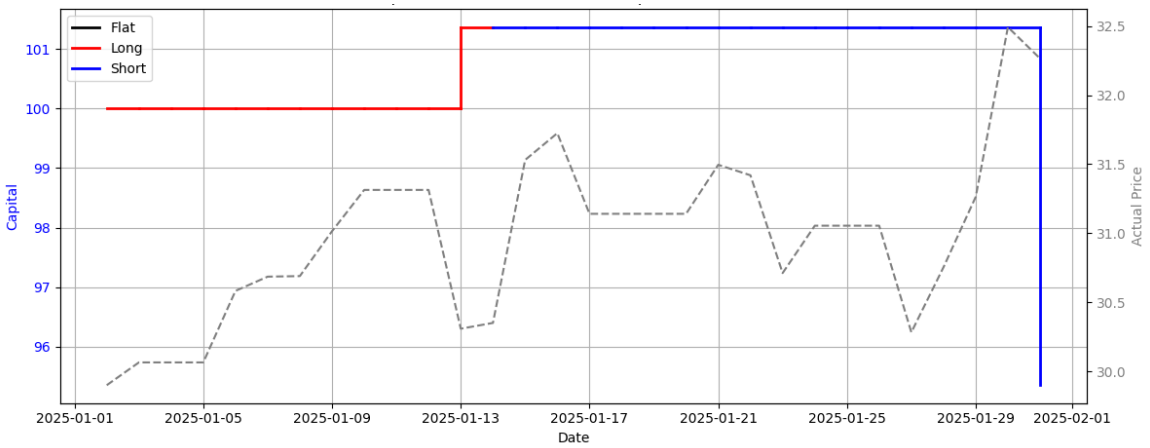
For more convenient comparison, we will assume the initial investment portfolio is 100 US Dollars unless otherwise stated. The results of trading simulations for Brent Oil Futures and Silver Futures are shown in the following diagrams.

As we can see from Fig 10c, the ARIMA model of Brent Oil future anticipated a turning point from positive gradient to negative, hence from 13th January 2025, the trading simulation algorithms would choose to stop longing and start shorting the future. This coincides with what we have seen in Fig 11a. Similarly, the ARIMA forecasting model shifted from long to short for silver futures from January 14th 2025, until the end of January.



(a) Brent Oil Future P&L Investment Returns

Fig 11: Naïve conservative/optimistic (ARIMA–gradient) step-wise capital curves for the two commodities. The dashed grey line is the actual price (right axis); coloured stair-steps show trading positions and the cumulative capital trajectory (left axis)



(b) Silver Future P&L Investment Returns

In terms of the investment portfolio returns, as we can see from the Jupyter Notebook script output below, trading simulation on brent oil futures produced a rather encouraging outcome of 10.70% monthly return, while resulting in a monthly loss of -4.65% for silver futures.

```
            ===== Naive Conservative/Optimistic (ARIMA Gradient) of Brent Oil Futures Trade Log ====
Long           | 2025-01-02 -> 2025-01-10 (7 days) | Entry=75.93, Exit=79.76 (Diff=3.83) |Return=5.04%| EndCap=105.04
Short          | 2025-01-13 -> 2025-01-30 (14 days)| Entry=81.01, Exit=76.87 (Diff=4.14) |Return=5.39%| EndCap=110.70
Flat (ignored) | 2025-01-31 -> 2025-01-31 ( 1 days)| Entry=76.76, Exit=76.76 (Diff=0.00)|Return=0.00% | EndCap=110.70
=== Final Results of Brent Oil Future Tradings: Naive Conservative/Optimistic (ARIMA Gradient) ===
Final Capital = 110.70,  PnL = 10.70,  Return = 10.70%

            ===== Naive Conservative/Optimistic (ARIMA Gradient) of Silver Futures Trade Log ====
Long  | 2025-01-02 -> 2025-01-13 (8 days)  | Entry=29.90, Exit=30.31 (Diff=0.41)  | Return=1.37%   | EndCap=101.37
Short | 2025-01-14 -> 2025-01-31 (13 days) | Entry=30.35, Exit=32.27 (Diff=1.91)  | Return=-5.93%  | EndCap= 95.35
=== Final Results of Silver Future Tradings: Naive Conservative/Optimistic (ARIMA Gradient) ===
Final Capital = 95.35,  PnL = -4.65,  Return = -4.65%
```

4.6 Baseline Prototype Generalisation

The above performance results showed coherence with what we discussed in Section 4.3.1, the relatively similar past patterns of brent oil future time-series could lead to an unusually reliable prediction from ARIMA model, thus an uncommon high investment return. To check whether our methodology and workflow work for the other 10 assets, we have compiled the overall trading simulation yield in the following diagram. The overall procedure to replicate these deliverable results could be found in the GitHub and its corresponding ReadMe file here.

As we can see, for the majority of the chosen investment classes, the ARIMA model could lead to an enormous deviation from the actual price trend. And the trading rule will not adjust its strategy according to the continuous errors over time. Therefore, most asset investment turned out to be a loss, except Citigroup and NVIDIA stocks and the aforementioned Brent Oil Futures. To make the

result analysis more reasonable, we retrieved the 2025 UK Bank Interest Rate as the reference line if we save our money in the bank alternatively [41].

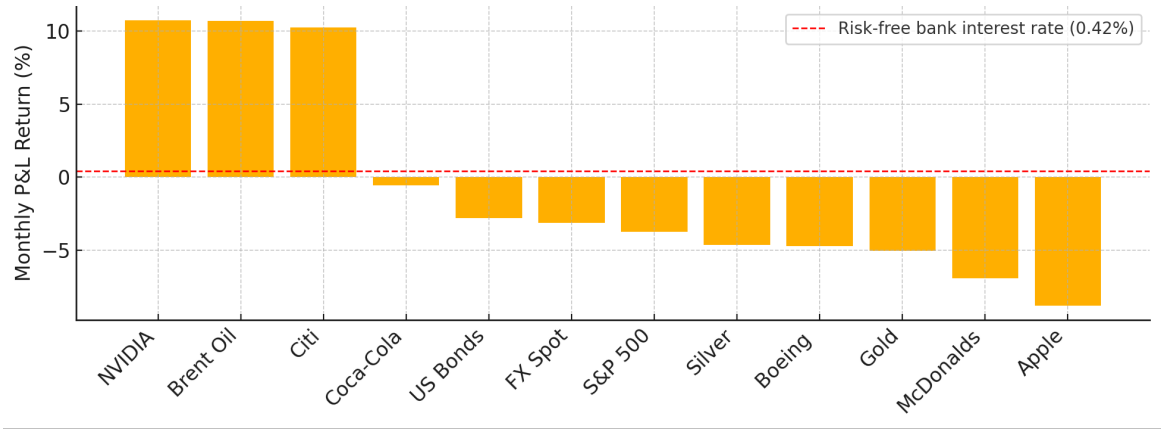


Fig 12: ARIMA model prediction & naïve trading strategy simulation over an entire basket of assets. The red dashed line shows the 2025 UK risk-free bank interest rate (0.42%) for comparison.

Consequently, the above result (Fig 12 brings us the curiosity of how we can improve the trading simulation performance in two ways, one from price prediction accuracy and the other from dynamically changing trading mythologies outlined in Section 1.3.

5 Price Prediction Model Progression

5.1 Introduction

In this section, we will examine the evolution of our asset price forecasting models through three progressively advanced cases: (1) Selected Model Combinations, (2) Mixture of Experts (MoE) and (3) Hybrid MoE with Selected Models. Each stage will build upon the previous configurations, aiming to improve prediction accuracy and convert the refinements into better trading performance. All models are then evaluated by the baseline trading strategies introduced earlier.

From the generated trading simulation results, comparative discussions will be provided to identify which model performed best, why certain models excelled, and what the outcomes imply about time-series behaviour and the corresponding model generalizability.

5.2 Case I: Selected Model Combinations

5.2.1 Model Setup

To begin with, the data preprocessing stage will choose the four feature groups as we did before in Section 4 (Price, Open, High, and Low). These four features will be scaled, sliced into windows, and predicted all at once. All the neural network-based models (LSTM, CNN, GRU, RNN, Transformer,

N-BTEATS, and N-HITS) shared a common interface in a single class, BaseModel. To provide an overview of the internal architecture of the complete model class, aligned with the foundational knowledge presented in Section 2:

1. CNN:
   - Uses two 1D convolutional layers (`conv1 and conv2`) with ReLU activation.
   - The input format `(batch_size, sequence_length, features)` will be permuted to `(batch_size, features, sequence_length)` before applying `Conv1D`.
   - The output will be flattened to 4 predicted feature values by a fully-connected layer.
2. LSTM:
   - A two-layer LSTM with `hidden_size=128`, `dropout = 0.1`
   - The final hidden state from the last time-step is used for the forward-connected layer to get the next day's feature values (`[Price, Open, High, Low]`).
3. GRU:
   - Similar to the LSTM architecture but uses a Gated Recurrent Unit from the `torch.nn` library.
4. RNN:
   - A vanilla RNN with 2 layers, `hidden_size=128`, ReLU non-linearity, and `dropout=0.1`.
   - The final time-step's hidden state is mapped to 4 outputs via a linear layer.
5. Transformer:
   - Maps each 4D input vector to a 128-dimensional embedding via a linear layer.
   - Then uses a `TransformerEncoder` with `nhead=8`, `dropout=0.1`, and 3 encoder layers.
   - After the Transformer encoder, the final time step is passed through a linear layer to produce the 4 feature outputs.
6. N-BEATS:
   - Simplified architecture inspired from N-BEATS literature for uni- and multivariate forecasting.
   - Flattens the $35 \times 4 = 140$ inputs, then passes through a few fully connected layers (FC → ReLU → FC → ReLU → FC).
7. N-HITS:
   - Similar ideas to N-BEATS. Each block is a multi-layer perceptron producing a 4D forecast.
   - The block outputs are summed across all blocks to form the final predictions.
   - Two classical regression baselines are also included via scikit-learn [48].
8. SVM:
   - Uses an RBF (Radial Basis Function) kernel with $C = 10.0$ and $\epsilon = 0.01$.
   - Wrapped in a MultiOutputRegressor so it can predict [Price, Open, High, Low] simultaneously.

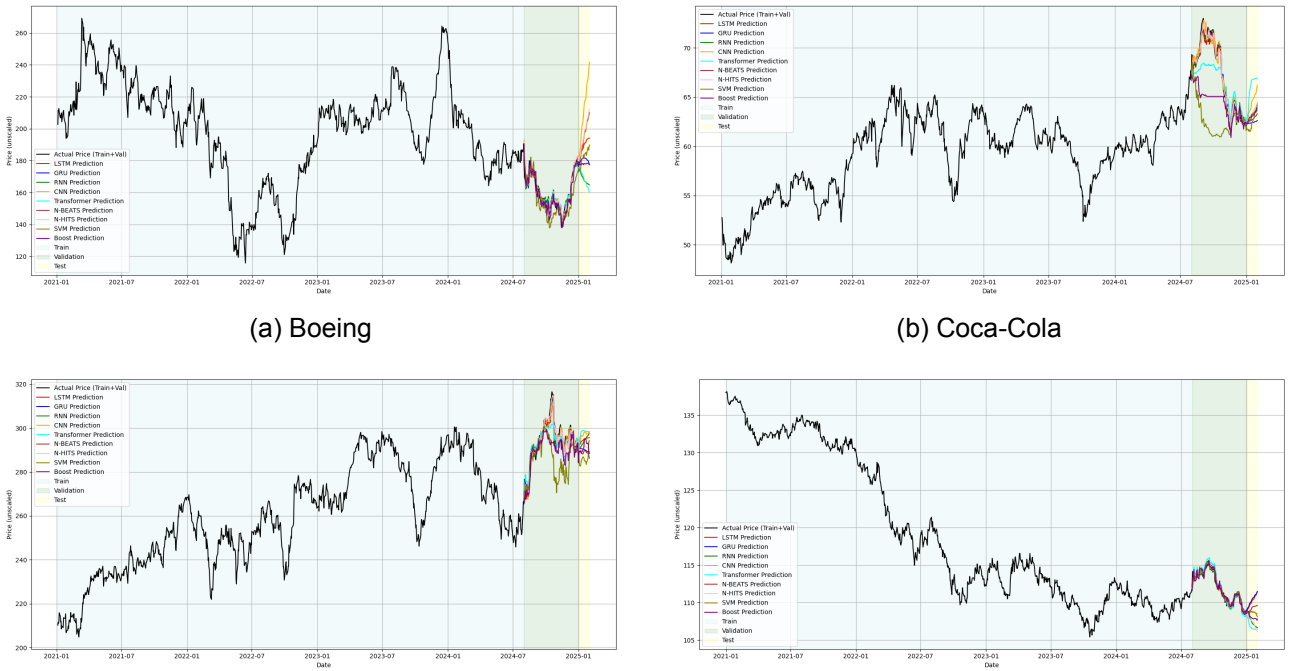
9. Boost:
    - The gradient boosting regressor is configured with `n_estimators=200`, `max_depth=3`.
    - Each 35-day × 4-feature window is reshaped to a 140-dimensional vector before fitting.

5.2.2 Model Training

Once the model functions have been defined, we are going to train all the above nine models consecutively. For the Pytorch models, we would like to select AdamW optimiser with the learning rate (LR) of 0.0005 and ReduceLROnPlateau scheduler to monitor LR based on the MSELoss criterion.

Then, an early stopping mechanism compares the half-blind validation loss (Section 4.3) for each epoch. After continuously failing to improve the modelling performance at patience of 150, the model training process stops, and the best model weights and bias are saved to a .pt file for future use without retraining. To train the sklearn models, they are fit once on the sets of `(X_train_flat, y_train_all)`. Similarly, by computing half-blind iterative validation loss, the resulting best model parameters are stored in a pickle file (.pkl) for future direct deployment.

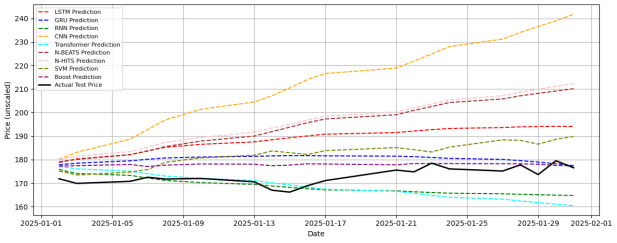
To evaluate the selected model methodology for time-series prediction, we have implemented it across 4 out of 12 assets in our portfolio basket. The results will be referenced for the subsequent polynomial fitting, trading simulations and the result discussions.



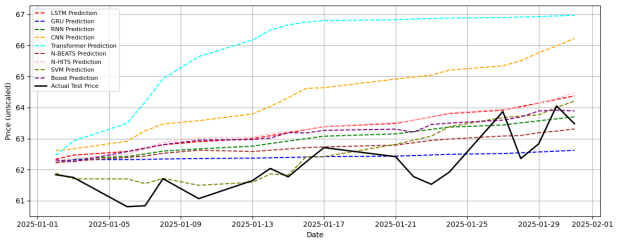
(a) Boeing (b) Coca-Cola

(c) McDonald's (d) US Bonds

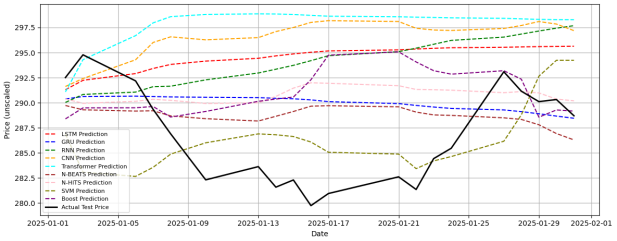
Fig 13: (a–d) Full-period forecasts by all candidate models for four assets.



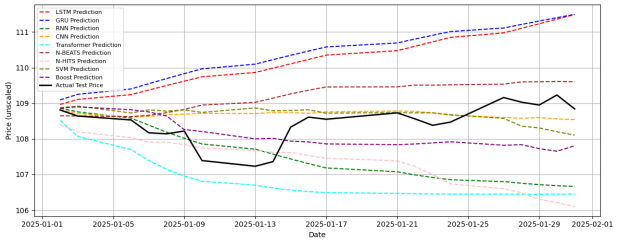
(a) Boeing zoom-in



(b) Coca-Cola zoom-in



(c) McDonald's zoom-in



(d) US Bonds zoom-in

Fig 14: (e–h) Corresponding zoom-in test period predictions.

5.3 Model Ranking by Performance Metrics

After training, all the validation and test predictions from each model are stored in the structure of `results_dic`, sorted by the model's name keys. These predictions are going to be compared against the ground truth actual prices retrieved from the website [6] on the test window. We will assign the model confidence score according to two metrics:

1. Cumulative Absolute Error over the entire test interval as $\sum_t \|\hat{y}_t - y_t\|_1$
2. Trend Matching Rate as fraction of days where $\text{sign}(\hat{C}_t - \hat{C}_{t-1}) = \text{sign}(C_t - C_{t-1})$

For each day d, if the sign of the actual price difference between the days matches that of the predicted price difference, we describe the model as matching the trend of the actual price trend, either in upward or downward movement over the two days.

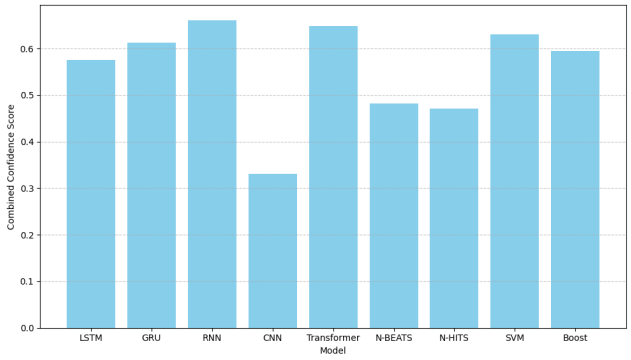
Therefore, the trend matching rate is then the proportion of days in the test window on which the model's predicted price movement aligns with the actual movement.

To finalise the allocation of combined model confidence, we shall first normalise the error score. That is, the lowest error among all models is mapped to a score of 1, and the highest error is mapped to 0. The model with errors in between would be linearly interpolated. Then, with the calculated normalised error score, the trend matching rate could be incorporated to give us the final confidence value. In this section, the weighting comprises 70% of the trend matching rate and 30% of the normalised error score. The models are then sorted by their "combined_confidence" in descending order to find the top 5 performance predictions. The selected 5 models will re-normalise their confidence score, summing

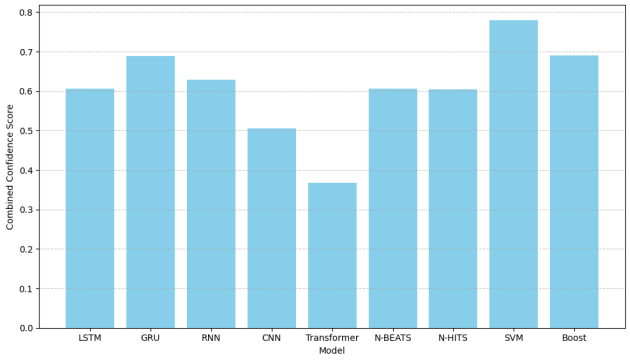
to 1 as an ensemble basis used in Section 6.

To give the reason why we would like to select 5 models out of 9 in this section, the top five models are intended to underpin the Naïve Conservative and Naïve Optimistic baseline trading algorithms. This is because some of the model forecasts among the nine models can be unreliable and large deviations can occur for certain asset time series.

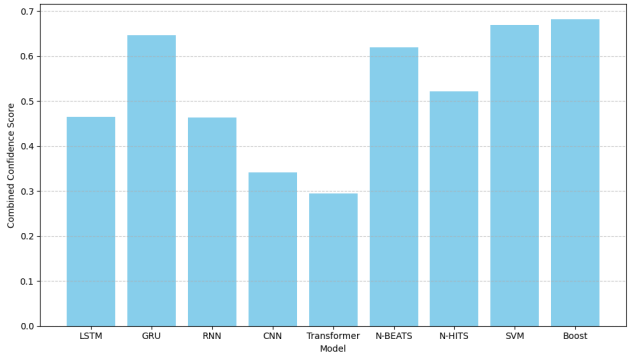
Fig 15: Pre-normalisation combined confidence scores of each forecasting model across four assets.



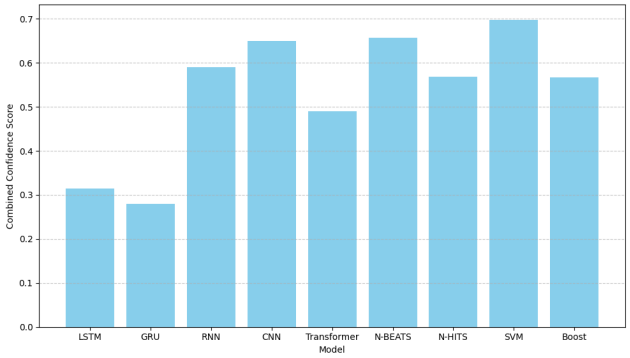
(a) Boeing (pre-normalisation).



(b) Coca-Cola (pre-normalisation).



(c) McDonald's (pre-normalisation).



(d) US bonds (pre-normalisation).

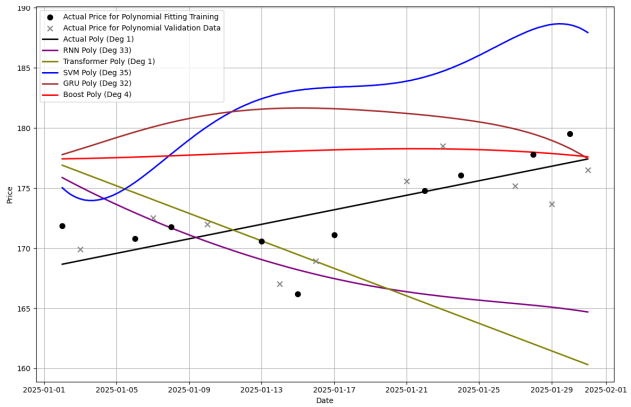
Index	Model Name	`normalized_confidence`
8	Boost	0.217176
7	SVM	0.213072
1	GRU	0.206192
5	N-BEATS	0.197225
6	N-HITS	0.166334

Table 3: Normalised confidence scores for the top 5 models of McDonald's stock

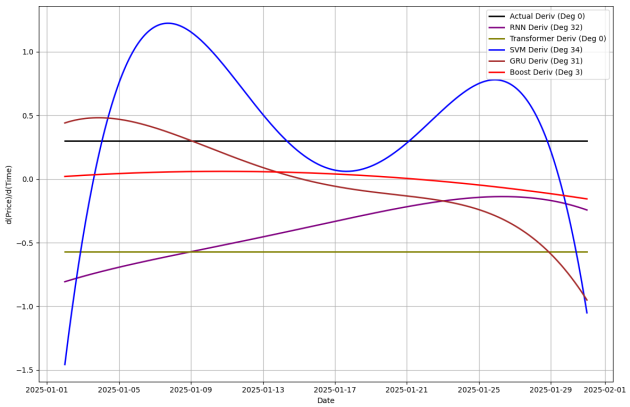
5.4 Polynomial Fitting and Trading Algorithm Executions

Polynomial fitting, transforming discrete data points of predicted and actual prices, is essentially the same as what we did in Section 4.4. The only difference is that the polynomial fitting should deal with the 5 top-performing models' predictions instead of the previous model.

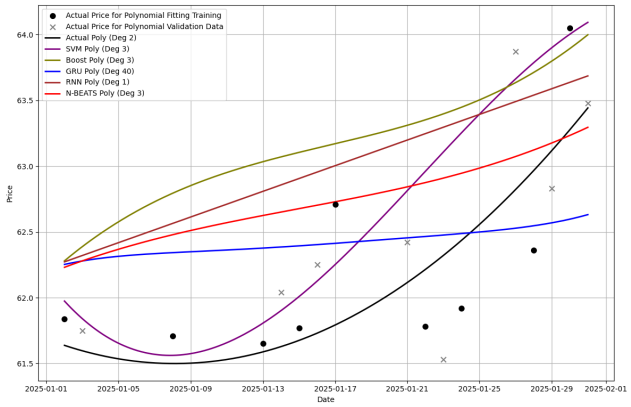
Similarly, the baseline trading strategy is implemented according to the polynomial derivatives. However, one key refinement of the algorithm is the classification of either “Naïve Conservative” or **“Naïve Optimistic”** decision-making approach. **Naïve Conservative** tactic implies the trading signal would be triggered ONLY when all the model’s predictive polynomial derivatives change polarity.



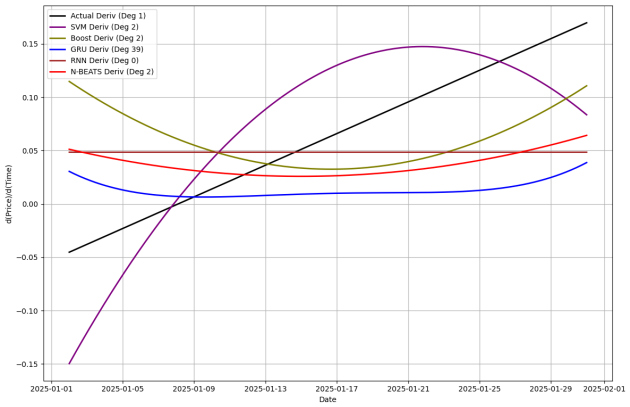
(a) Polynomial fitting for Selected 5 Model Predictions of Boeing Stock



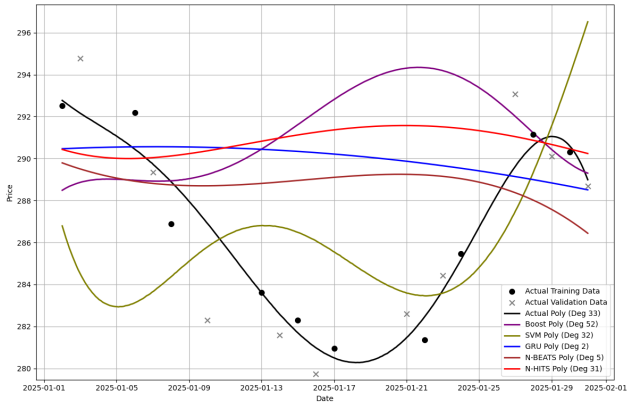
(b) Polynomial derivatives for Selected 5 Model Predictions of Boeing Stock



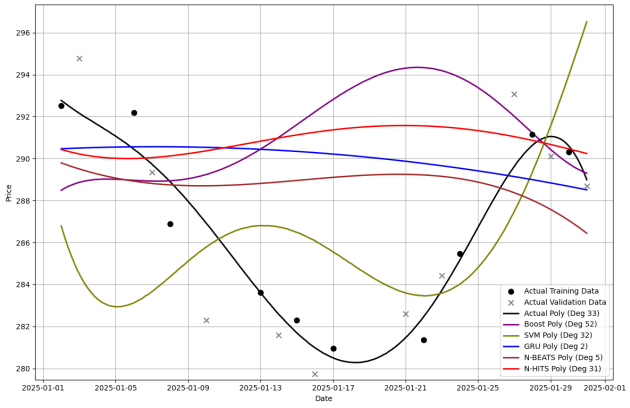
(c) Polynomial fitting for Selected 5 Model Predictions of Coca-Cola Stock



(d) Polynomial derivatives for Selected 5 Model Predictions of Coca-Cola Stock

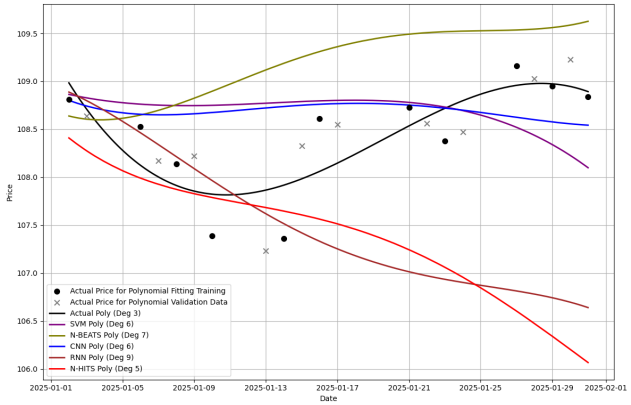


(e) Polynomial fitting for Selected 5 Model Predictions of McDonald’s Stock

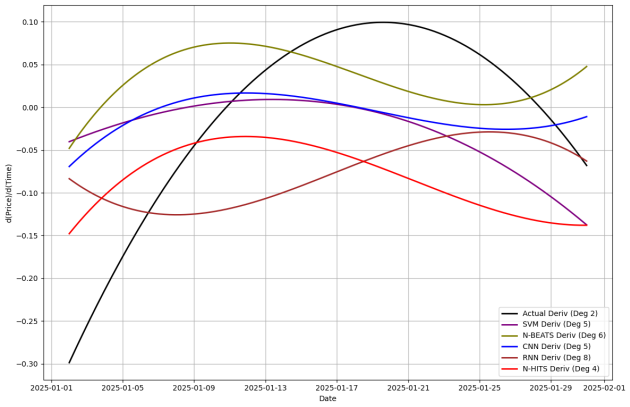


(f) Polynomial derivatives for Selected 5 Model Predictions of McDonald’s Stock

Fig 16: (a)-(f) Polynomial fits and their derivatives for the five selected forecasting models on Boeing, Coca-Cola and McDonald’s stock.



(a) Polynomial fitting for Selected 5 Model Predictions of US Bonds

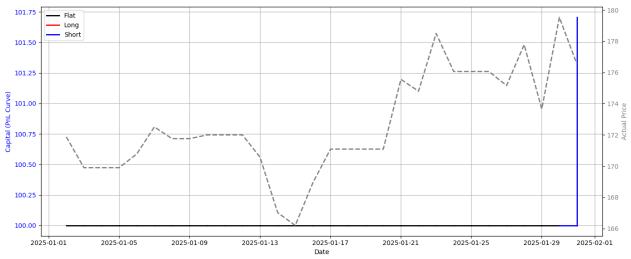


(b) Polynomial derivatives for Selected 5 Model Predictions of US Bonds

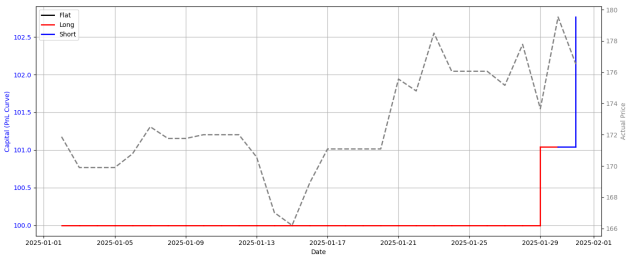
Fig 17: Polynomial fits and their derivatives for the five selected forecasting models on US bonds.

A long decision is made only if all derivatives are greater than 0, and shorting is made when all derivatives are less than 0. Naïve Optimistic strategy means the trading decision would be made AS LONG AS one model's predictive polynomial derivatives change polarity, long when positive gradients and short for negative gradients.

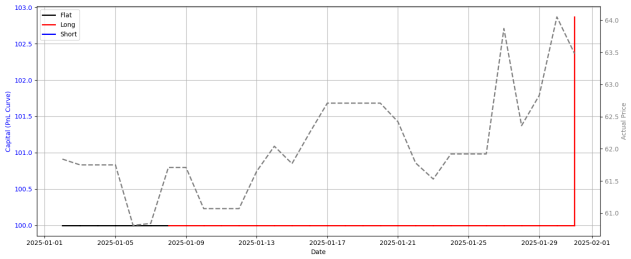
As we noted before in Section 4.5, we only account for the trading decisions exceeding one day. Otherwise, the segment would remain flat indicated in the diagram.



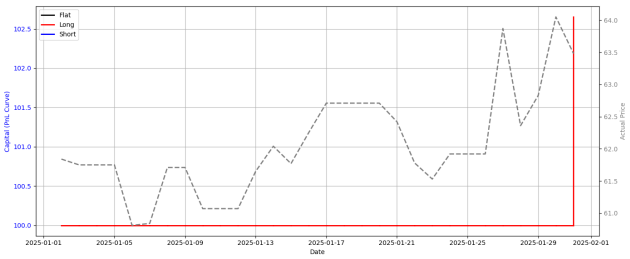
(a) Naïve conservative stepwise P&L curve for Boeing stock (offset settlement).



(b) Naïve optimistic stepwise P&L curve for Boeing stock (offset settlement).

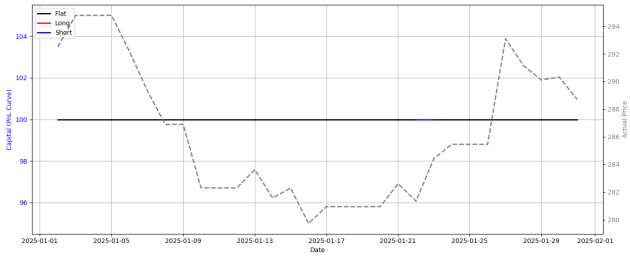


(c) Naïve conservative stepwise P&L curve for Coca-Cola stock (offset settlement).

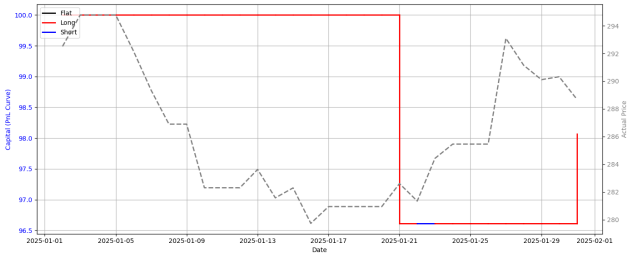


(d) Naïve optimistic stepwise P&L curve for Coca-Cola stock (offset settlement).

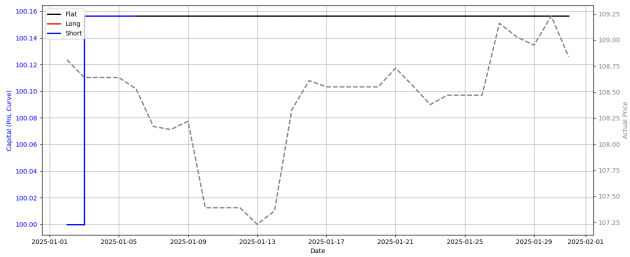
Fig 18: Naïve conservative or optimistic trading returns for Boeing stock and Coca-Cola stock;



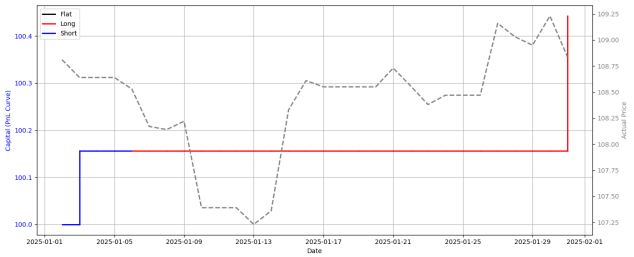
(a) Naïve conservative stepwise P&L curve for McDonald's stock (offset settlement).



(b) Naïve optimistic stepwise P&L curve for McDonald's stock (offset settlement).



(c) Naïve conservative stepwise P&L curve for US bonds (offset settlement).



(d) Naïve optimistic stepwise P&L curve for US bonds (offset settlement).

Fig 19: Naïve conservative or optimistic trading returns for McDonald's stock and US Bonds

5.5 Selected Model Combinations Generalisation Results Discussion

Extending the model prediction philosophy of selected model combinations to the rest 8 asset classes, we can simulate the overall investment return from the resulting trading decisions.

So, it would be a valuable investigation of how the chosen 9 models dealing with the various time-series datasets. Under which circumstances would these models perform well and are there any close correlations between our theoretical anticipation and the generated results?

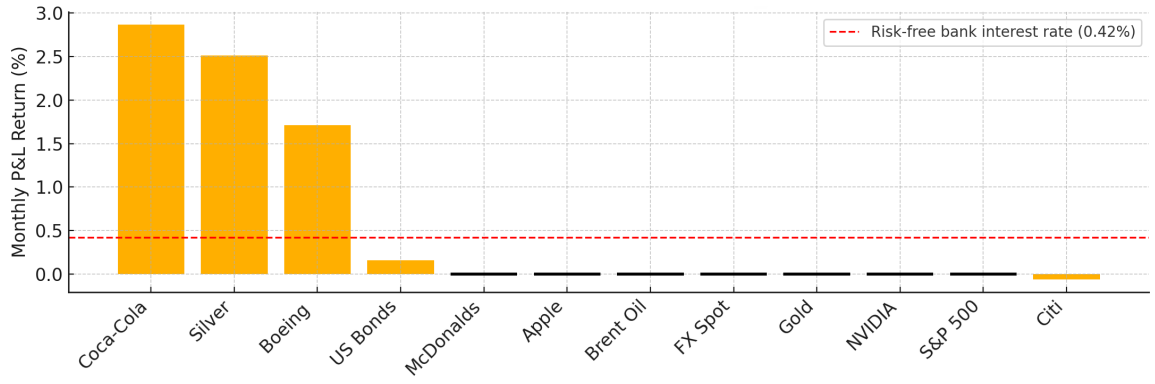


Fig 20: Normal model selection with naïve conservative trading simulation over the entire basket of assets (short horizontal black line represents zero).

As we can see from the right-hand-side compilation of the number of occurrences of certain models in the top-5 performance combination for the above four stocks. It is surprising to see only LSTM had not been chosen, maybe due to its tendency to over-smooth sudden turning points and burdens of long

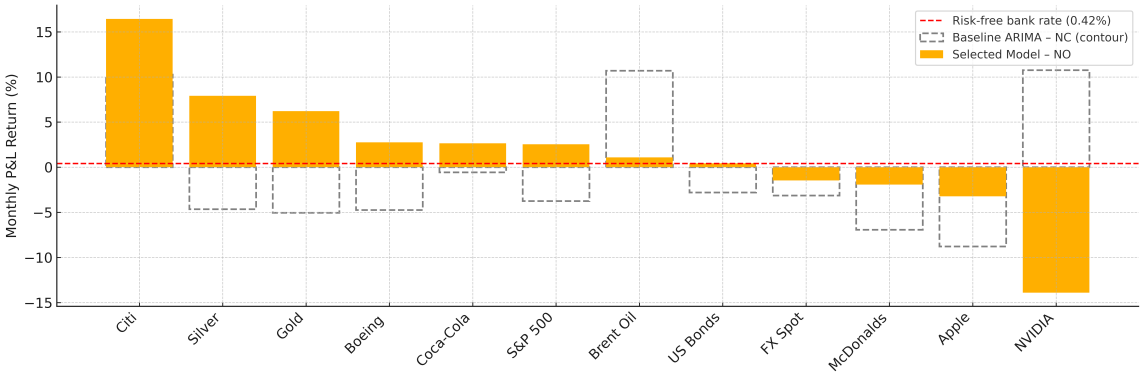


Fig 21: Normal model selection with naïve optimistic trading simulation over the entire basket of assets, in comparison with baseline model scenario

memory in memory. Besides, CNN requires a clear and repeating local motifs whereas one month prediction perhaps lack the seasonality. Meanwhile, data-hungry attention mechanisms demand a much richer context compared to the current around 750 training points.

To estimate why some models dominated these four assets, we can analyse the model's characteristics and the time-series dataset features. SVM did not auto-regress on its forecasts, so that its prediction error does not compound. Also, the chosen stock or bond price movement was close to linear plus white noise combinations, aligning with SVM's best weights and bias.

Model	Selections
SVM	4
GRU	3
Boost	3
RNN	3
N-BEATS	3
N-HITS	1
CNN	1
Transformer	1

Table 4: Number of times each model appears in the top-5 selection

GRU, on the other hand, had fewer parameters than LSTM, making it less tendency to overfitting on modest-sized training sets. Additionally, boost algorithms could average many weak signals, even if a few lags are spurious noise, the ensemble output is still likely to be stable. Lastly, RNN were capable of track day-to-day shifts without the need to model deeper structures, and a one-month window might be too short to cause the vanishing gradient harm.

Returning back to the overall trading simulations over the whole asset classes for the selected model combination price prediction philosophy, Naïve Conservative strategy indeed led to a more cautious behaviour – 7 assets turned to have no trading at throughout the time window because no polynomial gradients from model prediction could agree on the same trend, similar to Fig 18c. While the Naïve Optimistic approach can yield higher returns, investors should still be mindful, as it has resulted in losses for four assets. However, the overall performance seemed significantly improved from the baseline scenario shown in Fig 12. This motivated us to investigate whether we could further innovate our price prediction models for better forecasting accuracy.

6 Further Price Prediction Model Refinement

6.1 Introduction

From the last section, we have already seen selection of well-matched models could critically improve the overall trading simulation returns. So, a natural question comes up as to whether we could extend this idea by incorporating the strength of the model according to the past performance for the real, unseen future price prediction? Mixture of Experts (MoE) model philosophy has been proven as an effective method for substantially scaling up the model capacity with minimum computation overhead [42]. It introduces a gated ensemble architecture to dynamically learns to weigh the trust across the available models according to their complementary strengths for various time series [43].

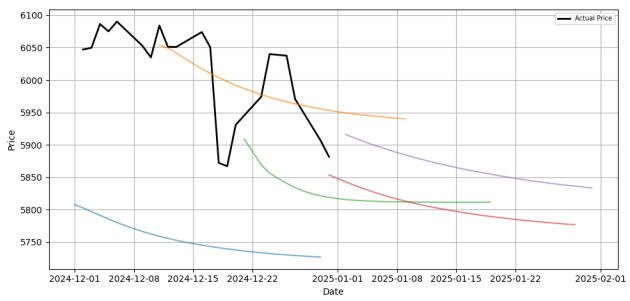
6.2 Case II: Mixture of Experts

6.2.1 Model Setup

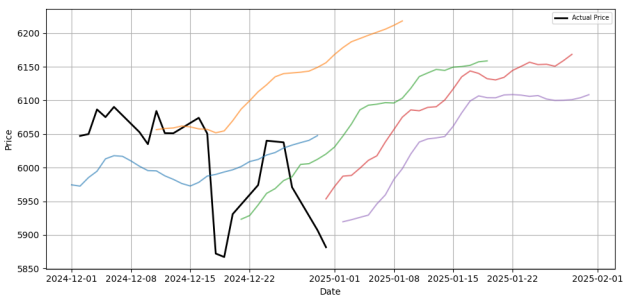
The guidance for building up our own Mixture-of-Expert model is to reflect the current available models' advantages in different data regimes. For example, CNN is capable of filtering local sub-patterns, LSTM tends to capture long-term dependencies and N-HITS for simulating hierarchical or high-granularity data, elaborated in the last sub-section 5.5.

To begin with, we will do a rolling forward approach to each base model to generate the `(Model, BaseDate)` pairs stored in the *final_rolling_fcst* table. This is because we want the models to "self-learn", which models should be assigned higher confidence by revealing actual data for previous time-series predictions. Then, the experience could be extrapolated to the upcoming unknown datasets. For example, the LSTM model gave a relatively more accurate trend prediction than N-HITS throughout the total-blind rolling window. We might trust more in the LSTM model over the next entire unknown period (January 2025) according to such prior information, assuming that the asset would not encounter significant external influential factors.

Fig 22: Rolling forecasts (every 10th base date) of the S&P 500 index by LSTM and N-HITS model



(a) LSTM rolling forecast of S&P 500 Index



(b) N-HITS rolling forecast of S&P 500 Index

To transform the above inspirations into a quantitative way, we are going to score the mini model on the same reference window of "previous" known December 2024 on three performance metrics:

- **Raw Accuracy metrics** (`baseAcc`): Mean absolute error to the actual price as $1 - \frac{\text{MAE}_{\text{Dec}} - \text{MAE}_{\text{min}}}{\text{MAE}_{\text{max}} - \text{MAE}_{\text{min}}}$
- **Trend–shape Fit** (`slopeAlign`): Fit a quadratic $p(t)$ for the predictions of the model. Score its sum-squared error (SSE) and e^{-SSE} against actual price data
- **Volatility Match** (`volAlign`): let σ_{pred} and σ_{real} be the standard deviations of daily price changes. Measured by $e^{-(\sigma_{\text{pred}}/\sigma_{\text{real}})} - 1$

Three criteria were chosen because MAE alone does not retrieve the price movement shape (slope) or risk intensity (volatility) of the real market. Then, a combined score and its normalisation result as weights to the expert model were derived as:

$$w_e = \frac{\text{MAE}_e^* \cdot \text{Slope}_e \cdot \text{Vol}_e}{\sum_j \text{MAE}_j^* \cdot \text{Slope}_j \cdot \text{Vol}_j} \quad \text{where} \quad \text{MAE}_e^* = 1 - \frac{\text{MAE}_e - \min(\text{MAE})}{\max(\text{MAE}) - \min(\text{MAE})}. \tag{17}$$

The static gating behaves like a naïve Bayesian posterior. Every independent likelihood component (the score) comprises how much belief we would impose on expert *e* for every calendar day d in the forecasting horizon.

Afterwards, we can calculate the initial `MoE_price(d)` by

$$\widehat{p}(d) = \frac{\sum_e w_e \cdot \text{price}_e(d)}{\sum_e w_e}. \tag{18}$$

To dynamically incorporate the information of all experts throughout the testing window, the rise and fall voting will be counted. If over 70% independent models agree with a price up on the following day, the MoE price prediction will be increased by 1%, otherwise a decrease of 1% if the majority of the model experts voting a price fall. A directional factor will also be included to reward or penalise the specific model's weights for correct or incorrect predictions when actual price data is revealed after yesterday's forecasting. Finally, a small Gaussian noise is added to mimic the market's random volatility and prevent the curve from becoming unnaturally smooth.

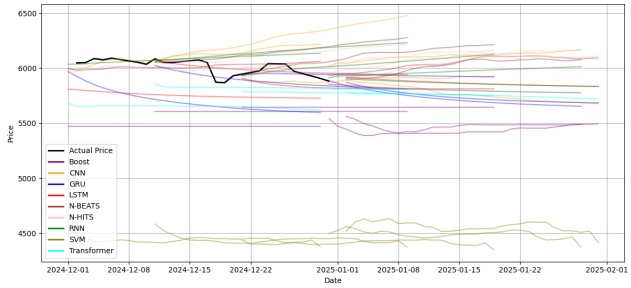
In summary, the continuous rolling updates for the Mixture-of-Experts price predictions could be derived as the following equation:

$$\hat{y}_t = \underbrace{\big(1 + b_t\big)\,\frac{\sum_e q_e\, d_{e,t}\, \hat{y}_{e,t}}{\sum_e q_e\, d_{e,t}}}_{\text{expert-weighted forecast}} + \mathcal{N}\big(0,\ \epsilon^2\, \hat{y}_t^2\big), \tag{19}$$

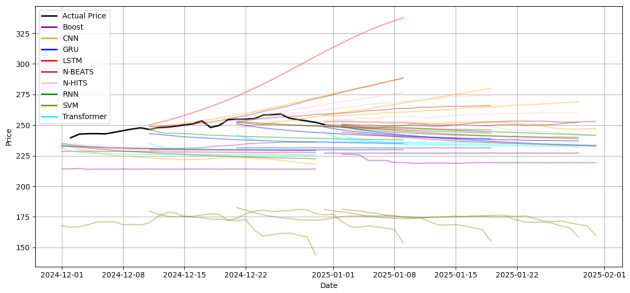
where $\hat{y}_{e,t}$ is the prediction of the expert e on day t, $q_e = (\text{baseAcc}_e) \cdot (\text{slopeAlign}_e) \cdot (\text{volAlign}_e)$ represents composite weight and $d_{e,t} \in \{0.9, 1, 1.1\}$ refers to the direction factor at time t.

Finally, $b_t \in \{+0.01, -0.01, 0\}$ is the cumulative consensus bias, $+0.01$ each time if more than 70% of experts predict "up", or -0.01 given 70% predict "down", and 0 otherwise.

The resultant rolling forecast across all nine models and the corresponding Mixture-of-Experts prediction curves on the test windows are shown below.

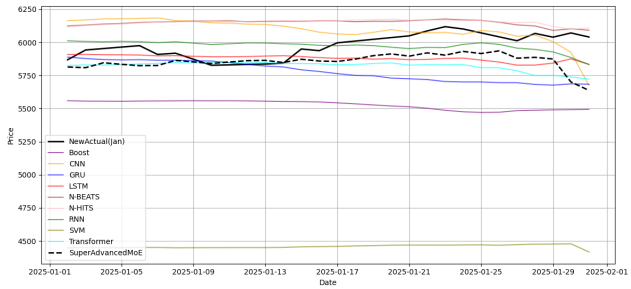


(a) Zoom-in joint forecast for S&P 500 (every 10th base date).

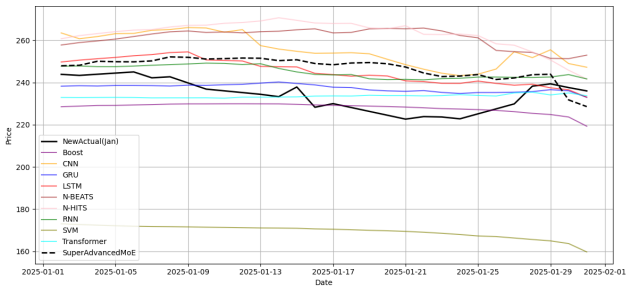


(b) Zoom-in joint forecast for Apple (every 10th base date).

Fig 23: Zoom-in Model Combination Rolling Forecast of S&P 500 Index and Apple Stocks



(a) Mixture-of-Experts forecasts (dashed) vs actual for S&P 500 over Jan 2–31, 2025.

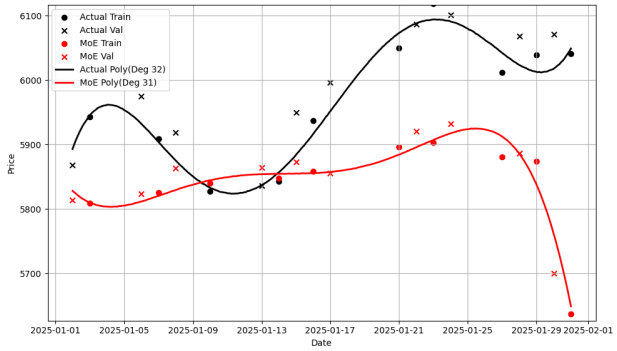


(b) Mixture-of-Experts forecasts (dashed) vs actual for Apple over Jan 2–31, 2025.

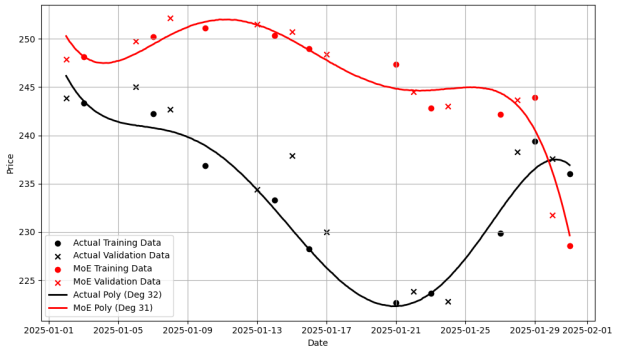
Fig 24: Mixture-of-Expert Model Predictions of S&P 500 Index and Apple Stocks

6.2.2 Overall Trading Simulation Progressions

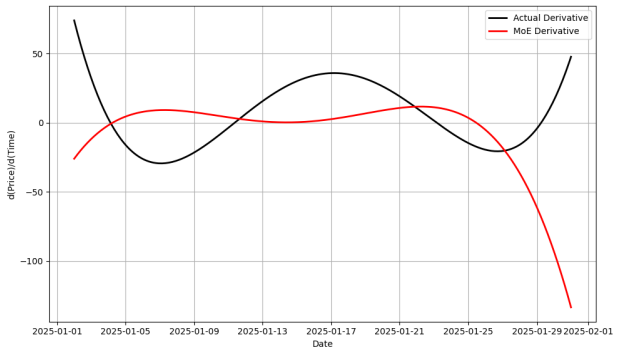
As we did in baseline prototype model and selected model combination scenarios, the trading decisions will be made according to the baseline trading strategy (Naïve Conservative and Naïve Optimistic). To avoid redundant repetitions, we are going to represent the results without delving into detailed operation explanations (demonstrated in Sections 4.4, 4.5 and 5.4). Note that since there is one curve (MoE) of polynomial derives of interest, the Naïve Conservative and Naïve Optimistic strategies will produce the same result.



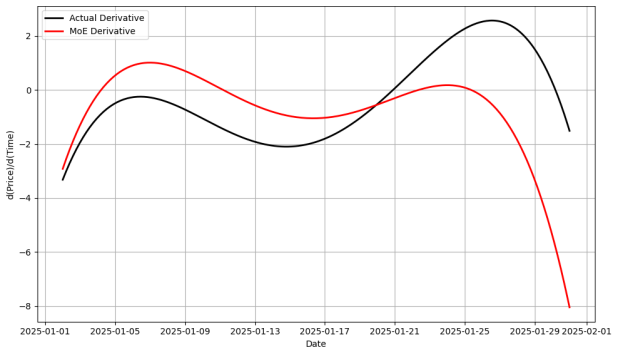
(a) Polynomial fitting of MoE predictions of S&P 500



(b) Polynomial fitting for MoE on Apple Stocks

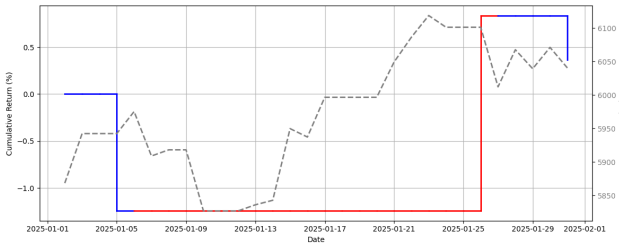


(c) Polynomial derivative curves for MoE on S&P 500.

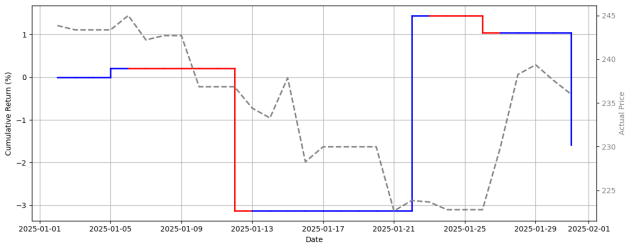


(d) Polynomial derivatives for MoE on Apple stocks

Fig 25: (a),(b) Best-fit degree of M polynomials for the MoE ensemble's predictions compared to actual prices; (c),(d) their corresponding first-derivative curves



(a) Baseline P&L for S&P 500



(b) Baseline P&L for Apple

Fig 26: Cumulative P&L returns of the MoE model with baseline trading strategies across two assets

6.2.3 Mixture-of-Experts Methodology Generalisation

To demonstrate the MoE philosophy is migratable and replicable to various asset classes, similar to what we did for the selected model combination generalisations, we expanded the above trading simulations over the rest 10 investment categories.

The above results over the totalling 12 asset classes were a little surprising as the same trading algorithms did not yield an anticipated higher return. Actually, there were 6 commodities ending in loss from investment, poorer than both Naïve Conservative and Naïve Optimistic trading strategies for the selected model combinations. Therefore, we were motivated to investigate whether we were able to integrate the strengths and capabilities of Mixture-of-Experts with Selected Model Combinations

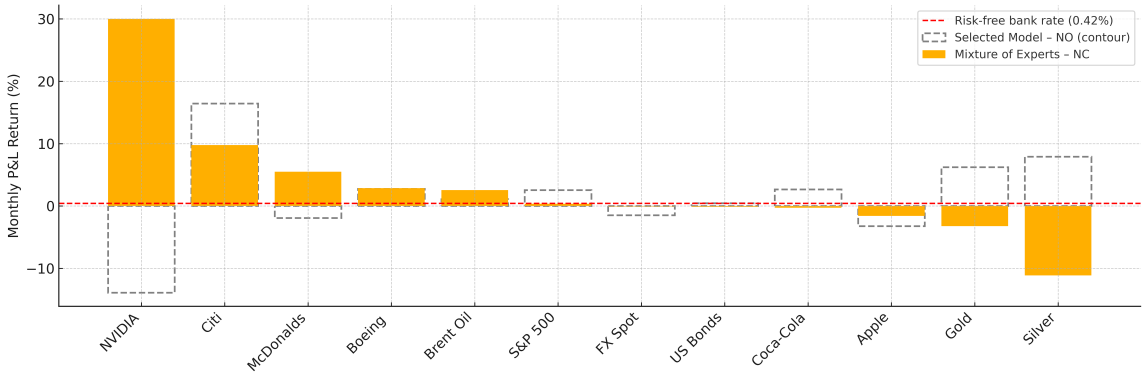


Fig 27: Mixture-of-Experts Model with baseline trading simulation over entire basket of assets, in comparison to Selected Model scenarios

approaches in the next sub-subsection.

6.3 Case III: Hybrid MoE with Selected Model Combinations

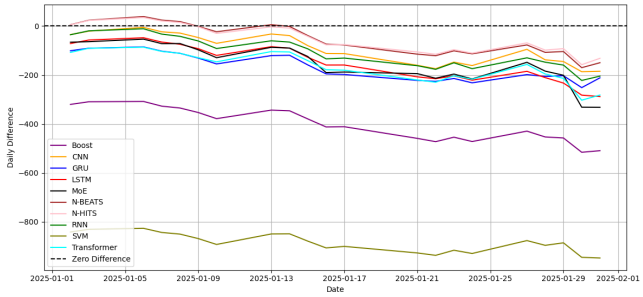
6.3.1 Model Adaptions

Following the skeleton of choosing the top 5 performed models illustrated in Section 5, the key difference is that we intend to enforce the synthesised MoE model into the top 5 models, regardless of whether the MoE model ranked among the best 5 models. This assumption is to ensure we can make a difference in the overall workflow to all cases in Section 5, facilitating the subsequent result implications and implications.

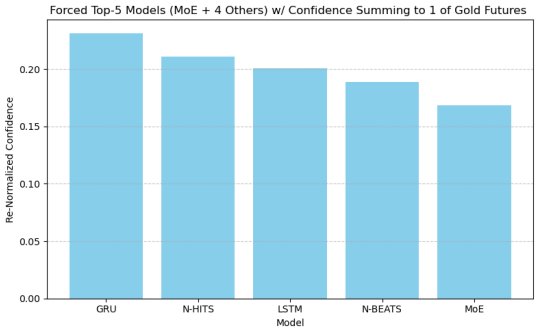
To include the MoE model in the top 5 short-list by compulsory, we will follow the following logic: (1) in the `select_top5_including_moe` function, no matter whether the MoE is already ranked in the first five, we always extract the MoE row and select the four highest-scoring non-MoE models. (2) We concatenate MoE with the other top 4 models in descending order. (3) Renormalise their confidence weights so that the weight sum still equals 1.

6.3.2 Overall Trading Simulation Progressions

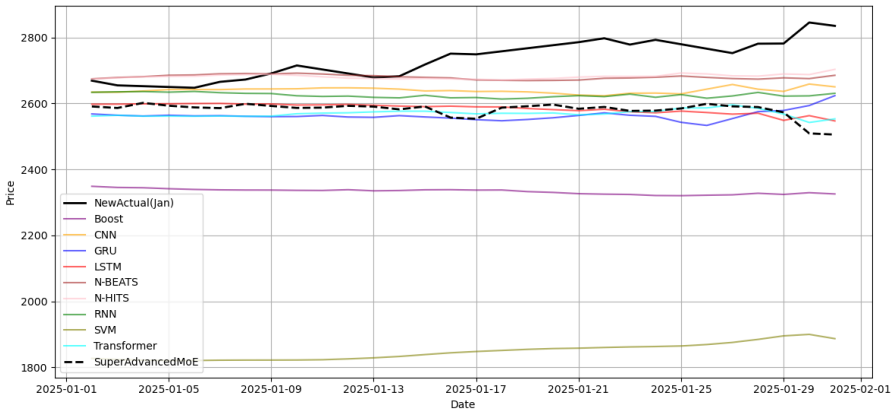
Similar to the previous sections, the trading decisions will be made according to the baseline trading strategy, either in Naïve Conservative or Naïve Optimistic philosophy. Without duplicated comments, we will attach all the generated diagrams of gold futures sequentially as a concrete example.



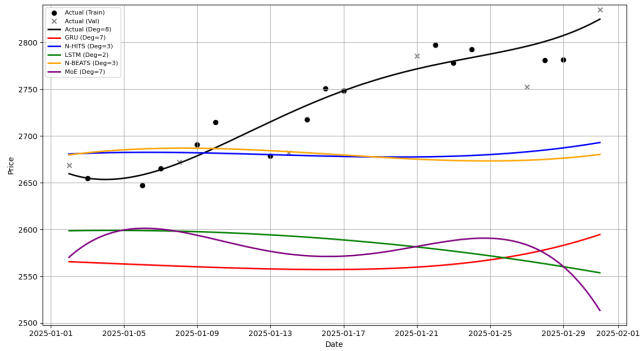
(a) Daily differences for all models vs. actual gold price



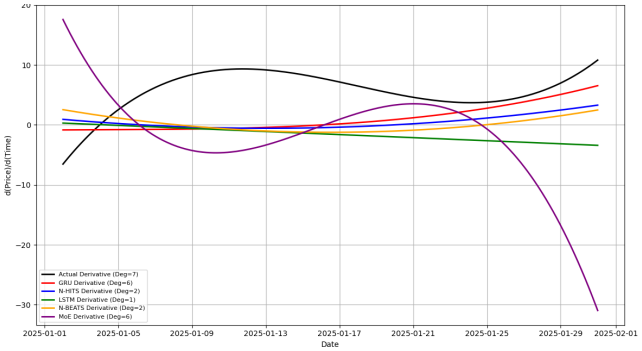
(b) Re-normalised confidence scores for top-5 models



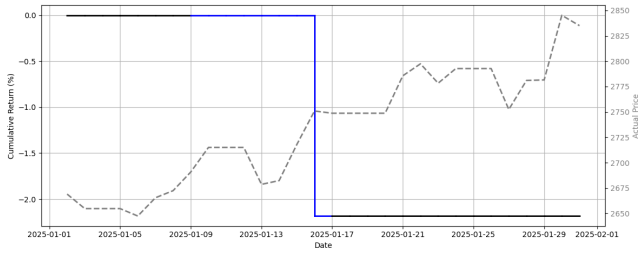
(c) Super-MoE & model-combination forecasts



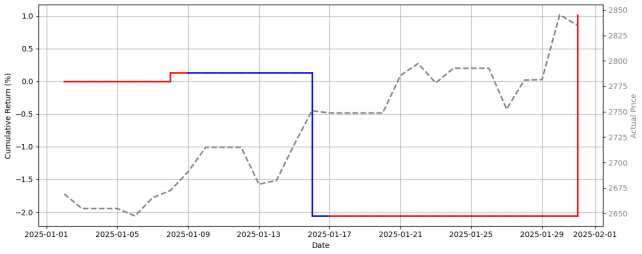
(d) Polynomial fits (MoE enforced)



(e) Polynomial derivatives



(f) Naïve-conservative trading sim.



(g) Naïve-optimistic trading sim.

Fig 28: (a) Daily prediction errors, (b) confidence weights, (c) Super-MoE forecasts, (d–e) polynomial fits/derivatives, and (f–g) naïve trading simulations for gold futures.

6.3.3 Generalisation of Hybrid MoE with Selected Model Combinations

To implement the workflow over the available asset classes, the trading simulations over the rest 10 investment commodities were shown below.

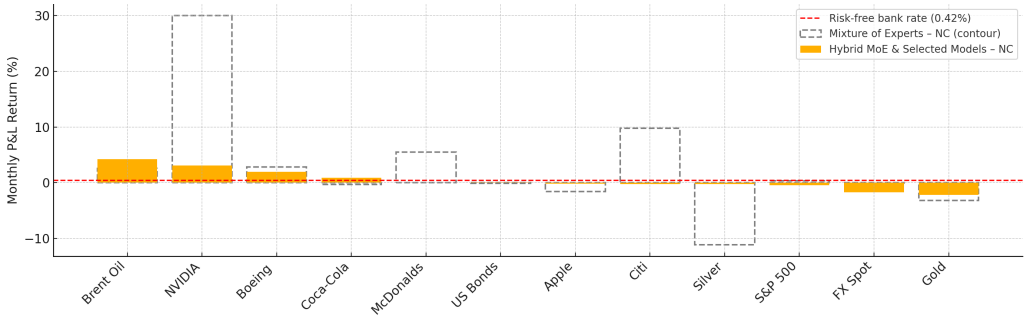


Fig 29: Baseline Naïve Conservative trading for Hybrid MoE & Selected Model over asset basket, in comparison to MoE Model scenarios

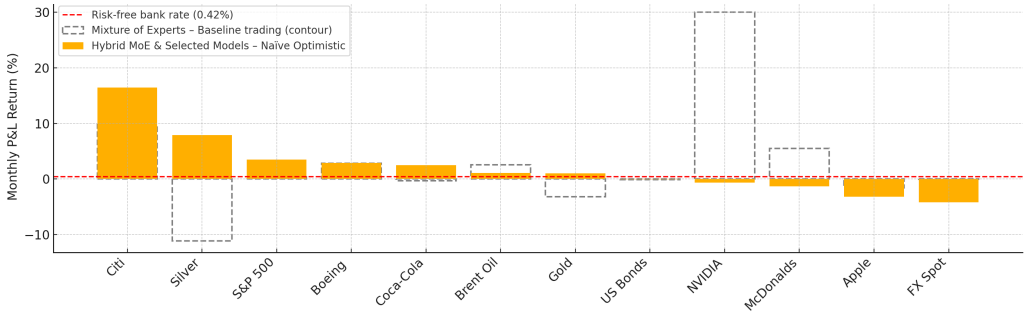


Fig 30: Baseline Naïve Optimistic trading for Hybrid MoE & Selected Model over asset basket, in comparison to MoE Model scenarios

6.4 Result Discussion and Evaluation

It is interesting to note that even the most complex model prediction techniques so far do NOT promise a definite enhancement for the overall investment returns. Compared to selected model baseline trading results, Naïve Conservative adoptions become less conservative, as only McDonald's stock did not involve any transactions compared to the previous 7 in total without actual trading. Naïve Optimistic decisions seemed not to make a significant difference, mainly with a reduction in loss for the negative return asset categories.

Delving into the reasons behind, the primary deductions might be that the forecasting windows for training the MoE model took place in December 2024, while the real adoption of the MoE model in testing equipped with previous configurations was in January 2025. That could imply that the tuned weights w_e could fail to cope with the upcoming unknown month, especially when market conditions vastly change [51]. Meanwhile, the 70% voting mechanism for the next-day's MoE price to increase or decrease could lead to a directional bias, particularly when most models' confidence weighting was

close to the binary threshold. The MoE model's prediction might be erratically jagged up-and-down, performing poorly in relatively stable time series. Finally, our innovative mixture-of-experts still only takes account of price as 1D factor optimisation, constrained to external macro-economic factors.

In terms of the practicality considerations of executing the overall workflows, machine learning training cost, both monetarily and environmentally, should be considered to strive for professional engineering principles taught in the B2 course. As mentioned in the prerequisite setup (Section 4.2), this project mainly leverages the resources of OeRC Skynet hardware and software environments with gratitude. From the NVIDIA-SMI monitoring interface, the average GPU power consumption was recorded as 150 W. Assuming similar power consumption levels for other hardware components (CPU, RAM, and networking devices), and considering an overall system electrical efficiency of approximately 60%, the total system power draw is estimated at around 500 W.

Item	**Assumptions / Formula**	**Result**
Energy used	$500\,\text{W} \times 7.5\,\text{h}$	3.75 kWh
Electricity cost (domestic rate) [44]	$3.75\,\text{kWh} \times £0.2486/\text{kWh}$	£0.93
Electricity cost incl. datacentre PUE 1.4 [45]	$1.125\,\text{kWh} \times 1.4 \times £0.2486/\text{kWh}$	£1.31
GPU depreciation ("wear") [46]	(£1,870 / 11,680 h) $\times$ 7.5 h	£1.20
Total monetary cost	£1.31 + £1.20	**£2.51**
Operational CO_2 (low grid factor 124 g/kWh)	$3.75\,\text{kWh} \times 0.124\,\text{kg/kWh}$	0.47 kg
Operational CO_2 (high grid factor 207 g/kWh) [47]	$1.125\,\text{kWh} \times 0.207\,\text{kg/kWh}$	0.78 kg
Embodied CO_2 allocation	$(130\,\text{kg}$ / 11,680h) $\times$ 7.5 h [48]	0.08 kg
Total CO_2 emissions (low/high)	0.47 + 0.08 / 0.78 + 0.08	**0.55–0.86 kg**

Table 5: Summary of costs for one 7.5-hour RTX 4090 training run (150 W average draw)

7 Trading Strategy Refinement

7.1 Introduction

So far, we were principally concentrating on future price prediction technique innovations. We progressed from ARIMA baseline model to selected model combinations, the construction of Mixture-of-Experts and eventually the hybrid model combinations of MoE and remaining chosen models. All the above could be considered as an Engineering side driven breakthrough.

However, the fintech evolutions, such as limited-order books, have greatly driven the modern market microstructure and therefore the subsequent high-frequency and low-latency quantitative strategies [52] [53]. Inspired by the relevant literature [44], we intended to explore whether we might refine simple polynomial-derived signal trading to a more adaptive and dynamic strategy.

7.2 Confidence-Weighted Voting Strategy

When developing the Mixture-of-Experts model by various model ensembles, we leveraged the idea of weighted voting among each mini-expert model with a confidence score. Motivated by this philosophy, our selected model ensemble generates a daily composite trading signal. Each model votes whether to take a position today, and each vote is weighted by the model's normalised confidence. Key components of the confidence formulations are:

Then, we are going to compute each model's composite confidence at day t by weighted sums of the above confidence components:

- **Static Confidence**: Use `combined_confidence` scores in the best-five model section as the baseline. Adjusted by $\alpha_{\text{fit\_up}}$ or $\alpha_{\text{fit\_down}}$ depending on whether the absolute prediction error of raw prices exceeds a predefined threshold.
- **Dynamic Fit Confidence**: The polynomial-fitting score measuring how well price changes align with the fitted polynomials, drifted by $\alpha_{\text{fit\_up}}$ or $\alpha_{\text{fit\_down}}$ whenever the absolute polynomial-fitting error is above the threshold.
- **Dynamic Gradient Confidence**: The polynomial-derivative direction accuracy tracker between today and yesterday, with the largest adjustment step among the three indicators.

Then, we are going to compute each model's composite confidence at day t by weighted sums of the above confidence components:

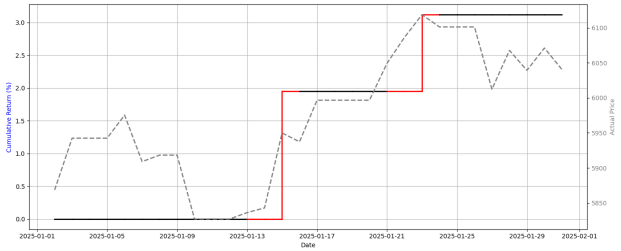
$$C_m(t) = w_{\text{static}} \cdot C_m^{\text{static}} + w_{\text{fit}} \cdot C_m^{\text{fit}}(t) + w_{\text{grad}} \cdot C_m^{\text{grad}}(t) \tag{20}$$

where are manually defined as weights $w_{\text{static}} = 0.2$, $w_{\text{fit}} = 0.3$, and $w_{\text{grad}} = 0.5$ to emphasise importance of recent directional predictive accuracy. After renormalisation, this would yield a composite confidence score of $\hat{C}_m(t) \in [0, 1]$ for each model. For the next trading day, we aggregated each model's signed confidence vote to determine the corresponding position signals.

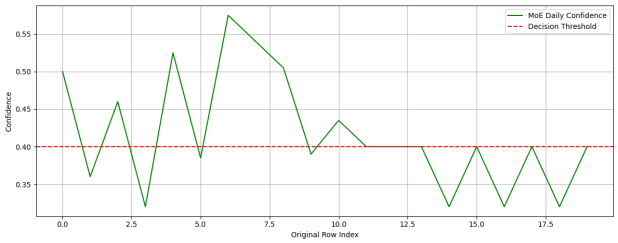
The ensemble issues a *Buy* signal $(+1)$ when $V(t) \geq \tau$, a *Sell* signal (-1) when $V(t) \leq -\tau$, and a *Flat* signal (0) otherwise, where τ is the manually set threshold value. Finally, we implemented an additional safeguard: if *all* models have voted for a loss-making trading decision, we optionally flip the signal to avoid repeatedly compounding losses.

7.3 Overall Trading Simulation Progress

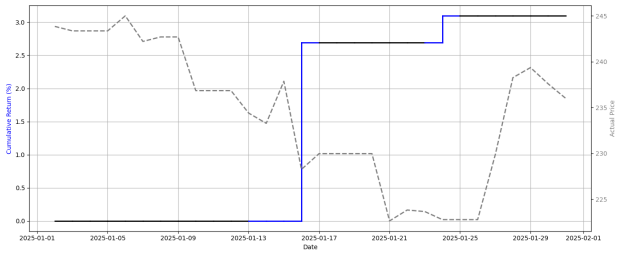
To ensure a clear and unbiased comparison, we would adopt the same price prediction technique applied to the same set of assets in Section 6.2 and 6.3.



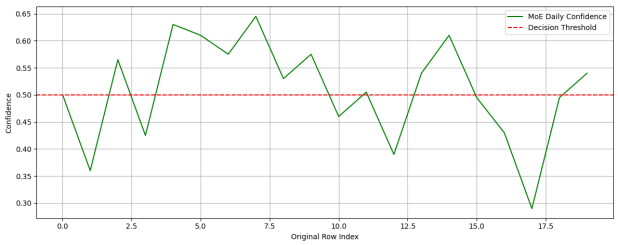
(a) Confidence Vote Trading for MoE Model Prediction of S&P 500 Index



(b) Daily Confidence Values of MoE Model Prediction of S&P 500 Index

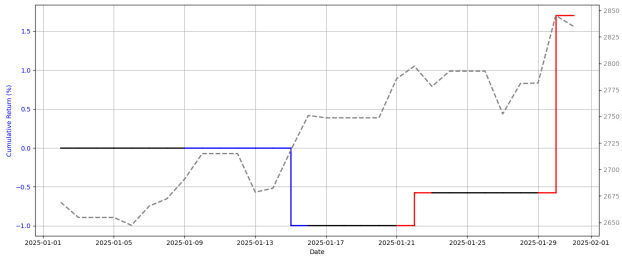


(c) Confidence Vote Trading for MoE Model Prediction of Apple Stock

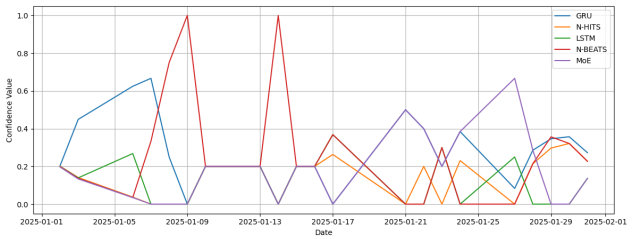


(d) Daily Confidence Values of MoE Model Prediction of Apple Stock

Fig 31: MoE-based trading signals and daily confidence for two assets: (a)–(b) S&P 500 Index, (c)–(d) Apple Stock.



(a) Confidence Vote Trading for Hybrid MoE & Selected Model Prediction of Gold Futures



(b) Daily Confidence Values of Hybrid MoE & Selected Model Prediction of Gold Futures

Fig 32: Hybrid MoE trading signals and confidence for gold futures: (a) PnL stepwise returns via confidence voting, (b) day-by-day confidence values from the ensemble.

As we can see in the following Table 6, for the selected assets in MoE model prediction and the Hybrid MoE and selected models, the population voting strategy in confidence weighting ensemble produce rather encouraging results. The monthly P&L return percentage all experienced a moderate increase after the new trading strategy adoptions.

Monthly P&L Return Percentage (%)			
Price Prediction Technique	Baseline Trading Strategy		Population Voting Strategy
	Naïve Conservative	Naïve Optimistic	Confidence-Weighted Vote
Mixture of Experts Model (Apple)	-1.59		3.10
Mixture of Experts Model (S&P)	0.36		3.12
Hybrid MoE and Selected Models (Gold)	-2.18	1.01	1.70

Table 6: Monthly P&L Return Percentage (%) for Trading Strategy Evolution of the MoE model and Hybrid MoE and selected model combinations

7.4 Generalisations and Results Discussions

Now, we are keen to evaluate the overall effectiveness of the new trading approach across all previously discussed price prediction techniques over the 12 chosen investment asset baskets.

Compared to Figure 12, the introduction of a confidence-weighted voting strategy allowed the Baseline ARIMA Prediction Model to achieve positive monthly returns for four additional assets. In comparison to Figures 20 and 21, trading simulations resulted in profitable outcomes for all assets, whereas previously, seven assets had no trades and four were in deficit for the baseline trading method.

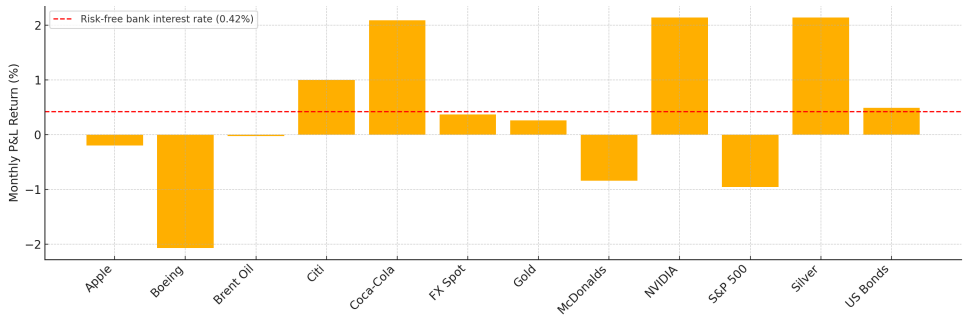


Fig 33: Confidence-Weighted Vote Trading for ARIMA Baseline Model Over Asset Basket

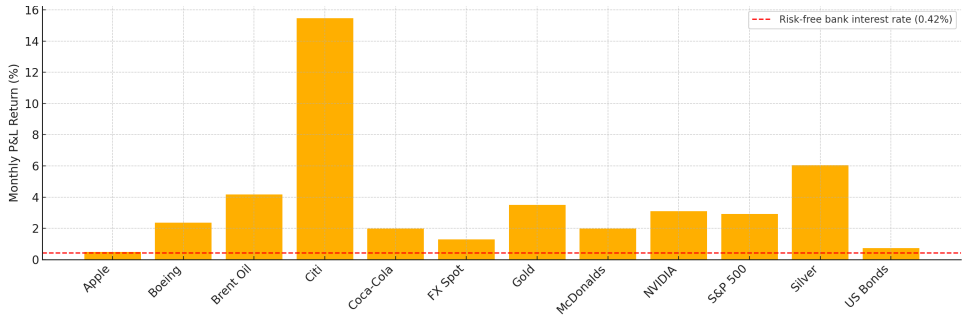


Fig 34: Confidence-Weighted Vote Trading for Selected Model Combinations Over Asset Basket

Referring to Figure 27, all asset simulations achieved returns exceeding the risk-free bank deposit interest, an improvement over the previous scenario where six commodities incurred losses. Finally,

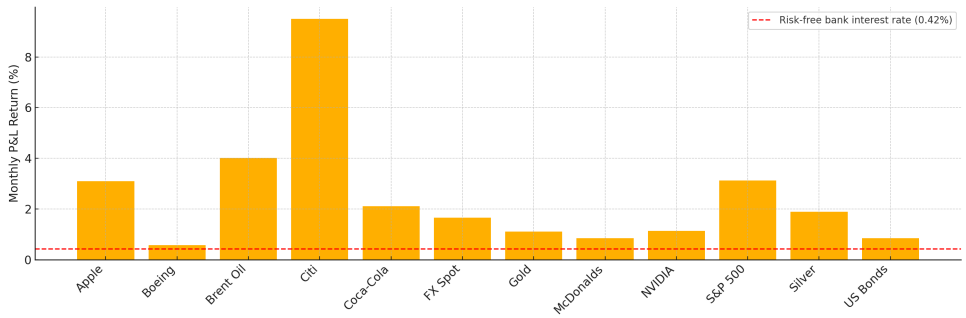


Fig 35: Confidence-Weighted Vote Trading for the MoE Model Over Asset Basket

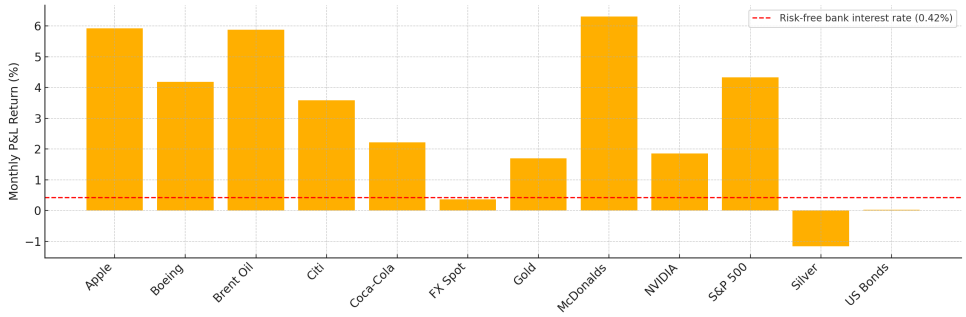


Fig 36: Confidence-Weighted Vote Trading for Hybrid MoE and Selected Models Over Asset Basket

as illustrated in Figures 29 and 30, nearly all assets achieved returns exceeding a satisfactory 2% threshold, with only one asset showing a deficit, compared to the earlier cases of six and four assets incurring losses under the naïve baseline trading strategy.

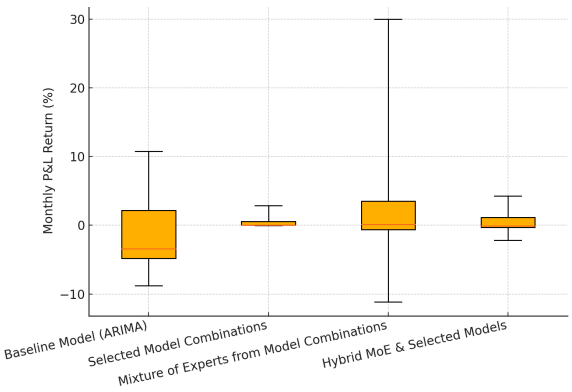
However, although the investment returns were competent and satisfactory for the new confidence-weighted vote trading strategy, it does NOT mean the improvement comes with no cost. For all baseline trading scenarios, the transaction simulations were completed in one setting. That is, we implemented the price prediction in any methodology and then generate the trading decisions for the whole subsequent future month. Therefore, individual investors could follow the instructions according to the trading log without the need to refer back to the script execution for the entire month. However, since confidence-weighted voting leverages the philosophy of reinforcement learning, such as Q-learning [49], the execution of the trading adoptions requires constant updates of each model's confidence. Essentially, it would incur the following additional operational procedures in practice. We might need to retrieve yesterday's asset information from the website [50] every day and insert that piece of information into the Excel worksheet in VSCode consecutively throughout the month. Although the update procedure does not consume much time, probably within 3 minutes, notably quicker than typical institutional investment methodology, the semi-automatic overall workflow could still be enhanced into a fully automated system. A more comprehensive suggestion of future work will be presented in the subsequent section.

8 Result Evaluation and Final Adoptions

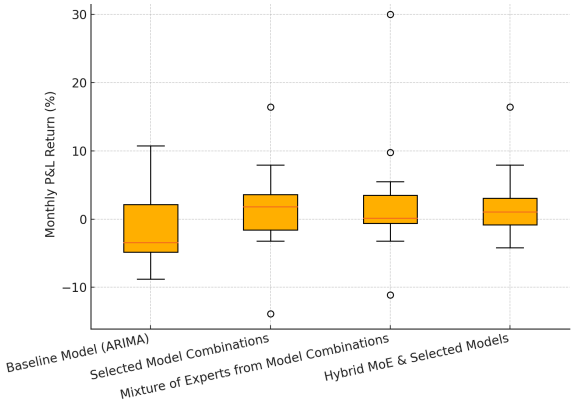
As described in Section 1.3, this project intends to explore how to maximise investment trading returns for Individual Investors along two storylines: time series price prediction and quantitative trading strategy design. To represent price prediction and trading strategy evolutions in a more comparable visualised way, we incorporate the variate controlling methodology along each dimension.

8.1 Price Prediction Dimension Evaluation

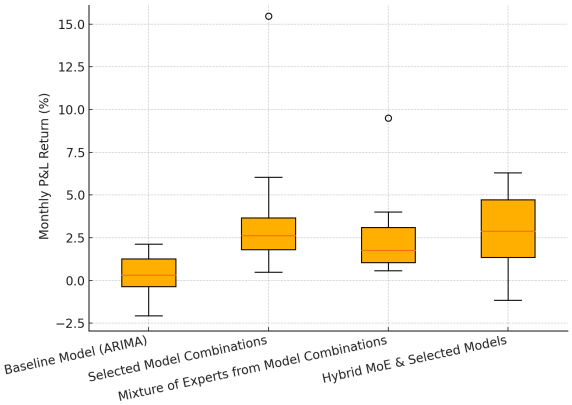
To investigate the effectiveness of price prediction technique improvements, we would like to compare them under the same available trading strategy settings. As we can see from the below Box-Whisker diagrams, increasing the complexity of price prediction models does not promise a rise in investment rewards. However, throughout all trading strategy adoptions, the selection of models, either the original version or the enforced inclusion of the MoE model, truly produces a more reliable yield with less variation across different assets.



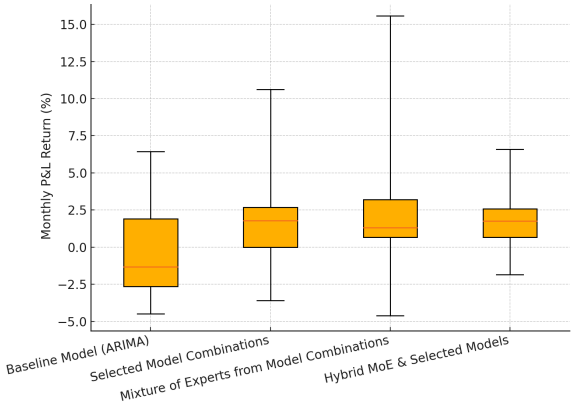
(a) Baseline (NC strategy)



(b) Baseline (NO strategy)



(c) Confidence-voting strategy



(d) Averaged over all strategies

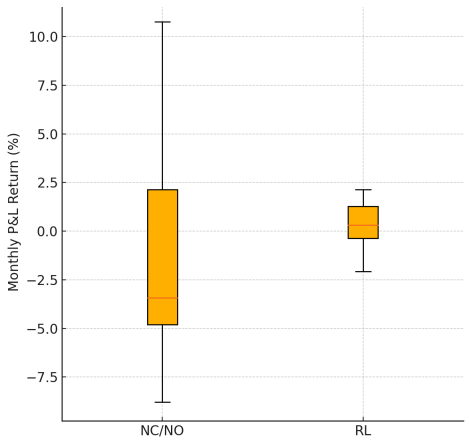
Fig 37: Prediction Technique Effectiveness on Monthly P&L Returns: (a–b) under baseline strategy (NC/NO), (c) under confidence-voting, and (d) averaged across all trading strategies.

8.2 Trading Strategy Dimension Evaluation

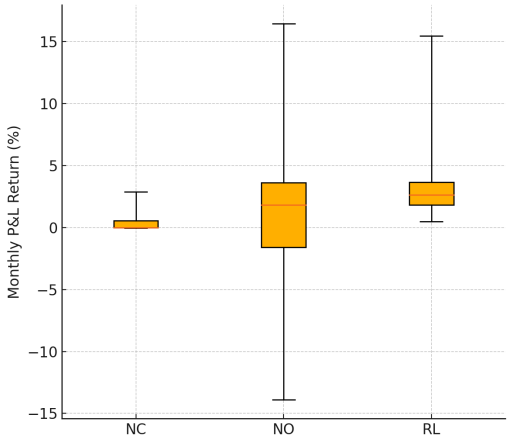
To assess the effectiveness of trading strategy innovations, we would like to compare them under the same price prediction model settings.

(Annotations: NC – Naïve Conservative, NO – Naïve Optimistic, RL – Confidence-Weighted Voting with Reinforcement Learning Analogy)

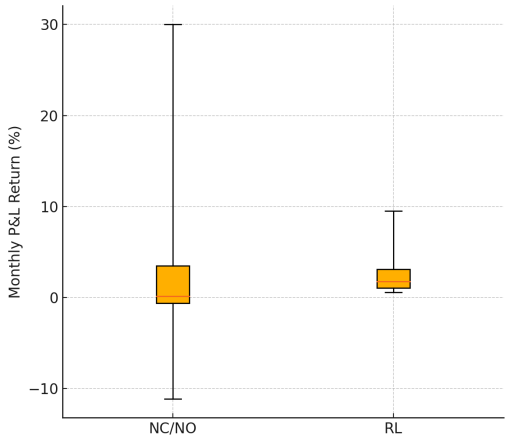
From the following Box-Whisker diagrams, adopting new trading strategy usually produce a slightly higher lower-quadrant investment return. More importantly, the population voting method almost guaranteed positive earnings with lower volatility than baseline strategy almost in ALL cases, validating the necessity of everyday information retrieval.



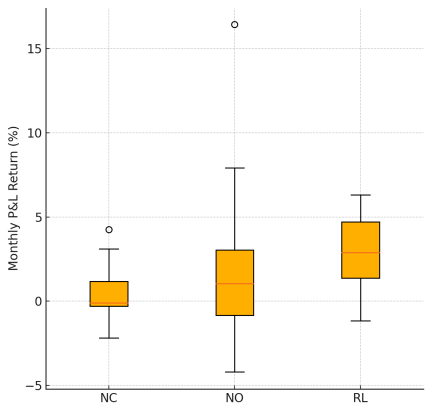
(a) ARIMA baseline model



(b) Selected model combinations



(c) MoE model



(d) Hybrid MoE & selected models

Fig 38: Trading-strategy innovation effectiveness on monthly P&L returns, under four model evolutions: (a) ARIMA baseline, (b) selected-model combinations, (c) MoE model, (d) hybrid MoE & selected models.

8.3 Final Results and Real Trading Adoptions

As illustrated in Figure 77, the top-right corner, representing the **Hybrid MoE and Selected Models with Dynamic Confidence Voting Trading Strategy**, indicates a balanced and practical combination suitable for actual trading across the basket of 12 assets. This strategy provides an attractive monthly yield of 2.93% (2.51% above the risk-free savings rate at banks) and maintains a relatively low risk profile, as nearly all underlying price models consistently generate positive returns for the selected assets.

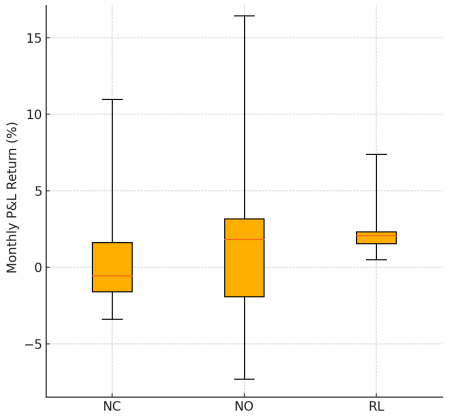


Fig 39: Trading Strategy Innovation Effectiveness on Investment Returns over ALL Models

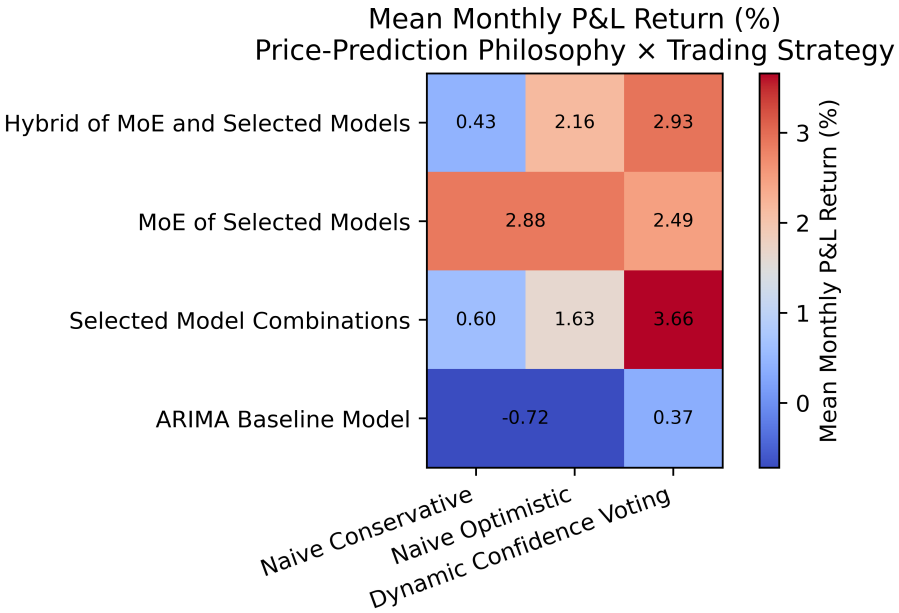


Fig 40: Overall Evaluation of Prediction Model and Trading Strategy Evolutions on a 2D Heat-Map

8.4 Basket Trading Verification

Therefore, we are now eligible to adopt the price prediction and trading strategy algorithms across our selected basket of assets mentioned in Table 2, demonstrating a fully integrated and automated algorithmic trading workflow directly deployable in real life.

The skeleton of the basket trading algorithm is to convert the assigned confidence of the MoE model of each asset into its corresponding weights for the asset basket, similar to the confidence weighting process for each prediction model for each asset. Then, our investment portfolio would allocate the investment fund to each asset according to the trading signal and the asset's confidence weight. The core principle of the basket trading algorithm is to exclusively trade assets with sufficiently high confidence scores, aiming to minimise idle periods without trading activity across different commodities.

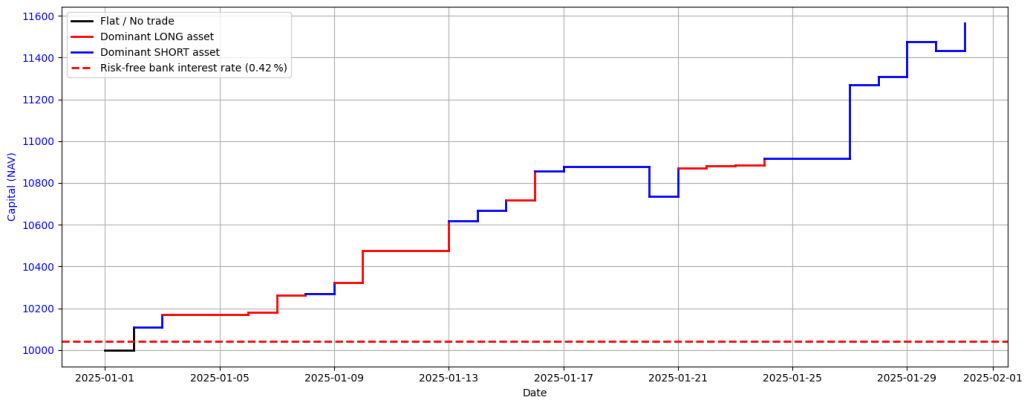


Fig 41: P&L raw revenue returns for basket trading of all assets, initial portfolio of $10,000 assumed

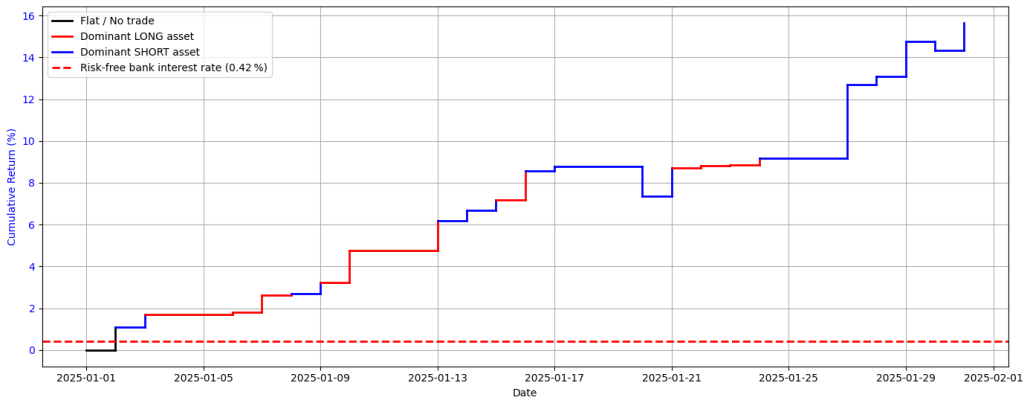


Fig 42: P&L percentage returns for basket trading of all assets

The above two diagrams represent the investment returns made by the trading decisions of the dominant asset on the corresponding transaction closing dates. A breakdown of each asset's trading execution numbers is shown in Fig 43. To have a spot-on look at the daily trading log for each asset on each day, we have compiled the record into a 2D heat map format, highlighting big gain transactions (more than 0.5%) of that asset and labelling the slashed tiles of loss-making decisions.

From Fig 44, 41, and 42, we can see that the ensemble trading may include loss-making decisions. For example, the losses observed on certain days in Fig 44 are reflected on the P&L curves as a slight drop in Fig 41 and 42. However, the basket algorithm successfully yielded promising and optimistic investment returns of 15.65% a month, equivalent to 187.8% annual profits, exceeding the industry-wide average (20-40%) by a significant amount [51] [52].

However, the above calculations did not take into account the transactional costs to calculate the real net profits of our investments. To begin with, we had a total number of 132 trading transactions throughout the test month (January 2025). As an individual investor participating in financial markets, there could be commission fees to the brokers for each transaction, breaking down as $£4 \times 76 + £3 \times 44 + £2 \times 12 = £460$, [53] [54]. Then, clearing & settlement admin fees are considered to be 76 stock trades $\times £0.10 = £7.60$ [55]. Finally, the estimated operational cost for January 2025 is $£460+$

£7.60 ≈ £468. Incorporating the model training cost mentioned in Table 5, the entire transactional cost is evaluated as £468 + £2.51 × 4 ≈ £478 every month, with ≈ £3.80 per trade.

In summary, the forecasted **monthly** net profit from our trading investments is approximately **$934** over an initial $10000 deposit, after deducting operational costs of roughly 40.2% to the revenue. The net monthly investment profits are anticipated at around **7.09%** per month and **85.1%** annually. The cost of automatic algorithm trading for individual investors seems to be significantly higher than for institutional participants [56]. The latter only incur about 3% to 4% operational cost in total, around 90% less than the cost of the individual stakeholder.

Meanwhile, we have to admit that our workflow currently only works for 12 assets in a particular month, and the annual performance might fluctuate. Moreover, we did not formally evaluate the risk-adjusted investment performance to assess our basket trading algorithm more comprehensively.

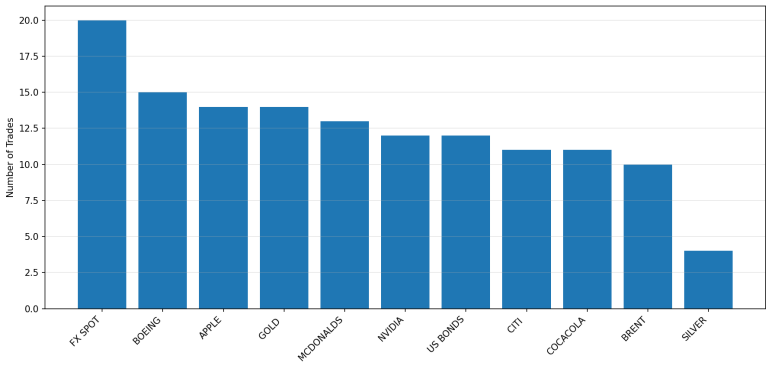


Fig 43: Number of Trading Executions for Each Asset Throughout the Unknown Testing January 2025

9 Conclusions and Future Work

9.1 Summary of Project

This project aims to develop a practical, transferable investment method tailored to individual investors' personal finance needs. We have presented a dual-track framework of price prediction and trading strategy improvement.

Proceeding on the price prediction technique, we began with the baseline prototype scenario, a simple ARIMA model with the naïve trading approach. Then, leveraging the emergence of various machine learning and deep learning model architectures, we intended to harness the complementary strengths of individual algorithms to yield lower forecasting error than the ARIMA model alone. Next, a Mixture-of-Experts (MoE) model was trained to dynamically gate and weight these experts. The further gains are adapted from each model's contributions to cope with different featured time-series datasets. Since the construction of the single Mixture-of-Expert model does not guarantee an im-

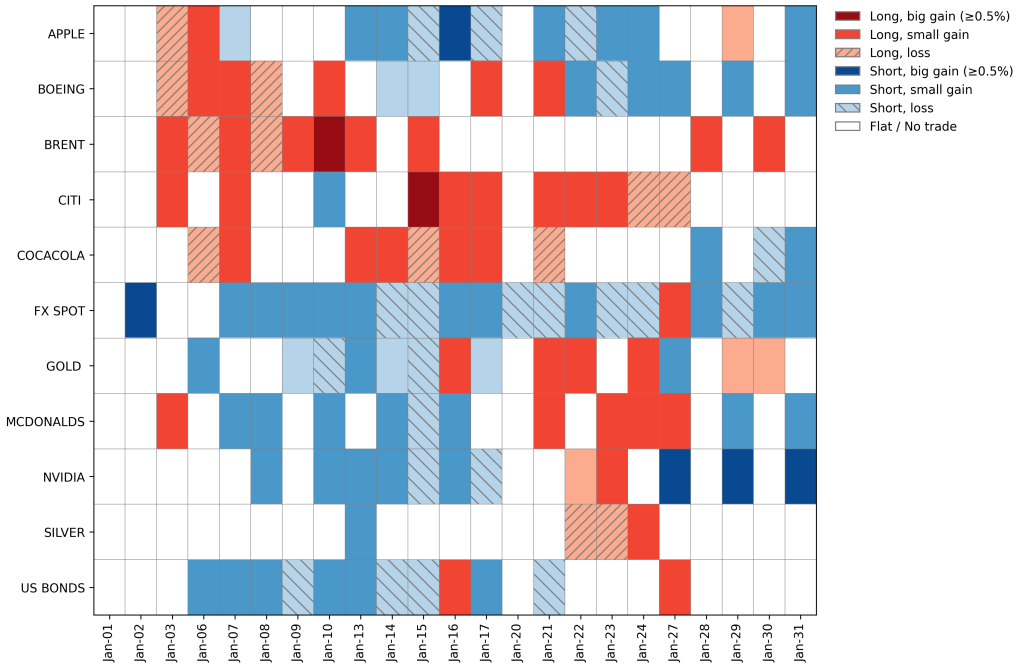


Fig 44: Visualised Representations of Daily Trading Logs Over Asset Baskets in January 2025

proved prediction accuracy, we were encouraged to incorporate the hybrid architecture of the MoE and the other four selected top-ranked models. Such combination has shown its potential to enhance predictive accuracy and price trend coincidence rate. These findings align with the literature review that deep learning and ensemble techniques tend to outperform traditional approaches in complex financial time-series forecasting [28] [38].

Regarding trading strategy innovations, we firstly implemented two baseline strategies. The conservative approach makes trading decisions when only when all models agree change of polarity of polynomial derivatives while the optimistic method adopts positions as long as any one model polynomial derivatives change their sign. Both baselines could generate positive profits in most of the cases on our asset basket, and the optimistic strategy usually yielded higher returns at the cost of more frequent trade positions and higher result volatility. Consequently, introducing the confidence-weighted voting strategy provide a smoother decision-making process. By weighting each model by its recent accuracy and trend confidence, the new approach avoided overreacting to loss-making signals. In practice, the new trading procedure produced a modest increase for investment returns.

9.2 Future Work

As we briefly mentioned the limitations of the workflow in each section, we have assumed a manual time-series retrieval for confidence-weighted vote trading strategy implementation, pre-defined hyperparameters for confidence weight values, both in the configuration of MoE model and population voting strategy development. Meanwhile, we only deploy price prediction models over price history

input as the only one source without external world information. Therefore, future work to extend this research's practicality and result robustness could include the following aspects:

- **Automate data and retraining pipeline.** Integrate financial data APIs such as Investing.com or Yahoo Finance to implement a scheduled process for fetching daily prices and updating trading positions. This evolves the confidence-weighted strategy into a fully automated "live" system.
- **Hyperparameter tuning for MoE confidences.** Systematically optimise the MoE model's and population-voting strategy's confidence-weight parameters (static, poly-fit, poly-gradient) and the voting threshold τ via grid search or Bayesian optimisation. Adaptive tuning may further enhance the gating mechanism's responsiveness to market shifts in time series.
- **Incorporate external information.** Extend the framework by including exogenous inputs such as news headlines or social-media sentiment analysis. Prior studies [57] have shown that adding news-derived sentiment features can improve forecast robustness, especially around macroeconomic events (e.g. COVID-19, trade wars). Integrating such signals may help the models anticipate and adapt to black-swan events or regime changes that pure historical prices cannot capture.

The ultimate goal is to establish a robust, resilient, and replicable framework for price prediction and associated trading strategies across diverse assets, enabling individual investors to effectively participate in quantitative algorithmic trading within the financial industry.

References

[1] David Reid, Abir Jaafar Hussain, and Hissam Tawfik. Financial time series prediction using spiking neural networks. *PLoS ONE*, 9(8):e103656, 2014.

[2] Investopedia. Efficient market hypothesis (emh). `https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp`. Accessed: 2025-04-29.

[3] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020.

[4] Vijay Kumar Vishwakarma and Narayan P. Bhosale. A survey of recent machine learning techniques for stock prediction methodologies. *Neural Computing and Applications*, 37(4):1951–1972, 2024.

[5] Investopedia. How more accessible ai could revolutionize wall street trading. `https://www.investopedia.com/open-source-ai-wall-street-8786858`. Published: 2025-03-06; Accessed: 2025-04-29.

[6] Investing.com. Silver futures price today. `https://www.investing.com/commodities/silver`. Accessed: 2025-04-29.

[7] IBM. What are arima models? `https://www.ibm.com/topics/arima`. Accessed: 2025-04-29.

[8] Eryk Lewinson. Introduction to quantitative finance: Stylized facts of asset returns. *TDS Archive, Medium*, n.d. Accessed: 2025-04-29.

[9] Stylized facts. `https://en.wikipedia.org/wiki/Stylized_fact`. Accessed: 2025-04-29.

[10] Lleyton Ariton. A thorough introduction to arima models. *Analytics Vidhya, Medium*, n.d. Accessed: 2025-04-29.

[11] Fraidoon Omarzai. Xgboost classification in depth: Master xgboost classification with …. Medium, `https://towardsdatascience.com/xgboost-classification-in-depth`. Accessed: 2025-04-29.

[12] scikit-learn Developers. *1.4 Support Vector Machines*. scikit-learn, 2024. Accessed: 2025-04-29.

[13] Brijesh Soni. Understanding boosting in machine learning: A comprehensive guide. *Medium*, n.d. Accessed: 2025-04-29.

[14] Mathworks. Understanding support vector machine regression. `https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html`. Accessed: 2025-04-29.

[15] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[16] Juncheng Zhu, Zhile Yang, Monjur Mourshed, Yuan Jun Guo, Yimin Zhou, Yan Chang, Yanjie Wei, and Shengzhong Feng. Electric vehicle charging load forecasting: A comparative study of deep learning approaches. *Energies*, 12(1):1–20, 2019.

[17] Unknown. Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[19] Glass Box. The transformer: Attention is all you need. https://glassbox.org/transformer-attention-is-all-you-need. Accessed: 2025-04-29.

[20] Nixtla. N-beats. https://nixtla.github.io/nbeats. Accessed: 2025-04-29.

[21] Anonymous. N-beats. https://openreview.net/attachment?id=r1ecqn4YwB&name=original_pdf. Accessed: 2025-04-29.

[22] Nixtla. N-hits. https://nixtla.github.io/nhits. Accessed: 2025-04-29.

[23] Tijs van der Velden. Nhits: Interpretable deep learning ts forecasting. *Medium*, n.d. Accessed: 2025-04-29.

[24] Oreshkin, Boris N. and Carpov, Nikolay and Chapados, Nicolas and Seeger, Matthias. N-beats: Neural basis expansion analysis for interpretable time series forecasting. https://arxiv.org/abs/1905.10437, 2019. Accessed: 2025-04-29.

[25] Yajiao Tang, Zhenyu Song, Yulin Zhu, Huaiyu Yuan, Maozhang Hou, Junkai Ji, Cheng Tang, and Jianqiang Li. A survey on machine learning models for financial time series forecasting. *Neurocomputing*, 512:363–380, 2022.

[26] Ming-Wei Hsu, Stefan Lessmann, Ming-Chien Sung, Tiejun Ma, and Johnnie E.V. Johnson. Bridging the divide in financial market forecasting: machine learners vs. financial economists. *Expert Systems with Applications*, 61:215–234, 2016.

[27] William Toner and Luke Darlow. An analysis of linear time series forecasting models, 2024.

[28] Shangkun Deng, Xiaoru Huang, Yingke Zhu, Zhihao Su, Zhe Fu, and Tatsuro Shimada. Stock index direction forecasting using an explainable extreme gradient boosting and investor sentiments. *The North American Journal of Economics and Finance*, 64:101848, 2023.

[29] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.

[30] A. Chakraborti, Marco Patriarca, and M. Santhanam. Financial time-series analysis: a brief overview. *arXiv.org, Quantitative Finance Papers*, 5, 04 2007.

[31] D.T Pham and D Karaboga. Training elman and jordan networks for system identification using genetic algorithms. *Elsevier*, 2010. Accessed: 2025-04-29.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[33] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[34] Mohit Apte and Yashodhara Haribhakta. Advancing financial forecasting: A comparative analysis of neural forecasting models n-hits and n-beats, 2024.

[35] Omer Berat Sezer and Ahmet Murat Ozbayoglu. Financial trading model with stock bar chart image time series with deep convolutional neural networks, 2019.

[36] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

[37] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2020.

[38] Subba Rao Polamuri, Dr. Kudipudi Srinivas, and Dr. A. Krishna Mohan. Multi-model generative adversarial network hybrid prediction algorithm (mmgan-hpa) for stock market prices prediction. *Journal of King Saud University - Computer and Information Sciences*, 34(9):7433–7444, 2022.

[39] Vu Minh Ngo, Huan Huu Nguyen, and Phuc Van Nguyen. Does reinforcement learning outperform deep learning and traditional portfolio optimization models in frontier and developed financial markets? *Research in International Business and Finance*, 65:101936, 2023.

[40] David Enke and Nijat Mehdiyev. Stock market prediction using a combination of stepwise regression analysis, differential evolution-based fuzzy clustering, and a fuzzy inference neural network. *Intelligent Automation & Soft Computing*, 19, 12 2013.

[41] Bank of England. Bank rate history and data. https://www.bankofengland.co.uk/boeapps/database/Bank-Rate.asp. Accessed: 2025-04-29.

[42] Omar Sanseviero, Lewis Tunstall, Philipp Schmid, Sourab Mangrulkar, Younes Belkada, and Pedro Cuenca. Mixture of experts explained, 2023. Accessed: 2025-04-29.

[43] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *arXiv preprint arXiv:2503.07137*, 2025. Accessed: 2025-04-29.

[44] Ofgem. Quarterly Domestic Tariffs in the UK: Q2 2025. https://www.ofgem.gov.uk/publications/quarterly-domestic-tariffs-q2-2025, April 2025. Accessed April 2025.

[45] Uptime Institute. 2024 Global Data Center Survey. Technical report, Uptime Institute, 2024.

[46] Scan Computers. NVIDIA GeForce RTX 4090 Graphics Card, 2025. Accessed April 2025.

[47] National Grid Electricity System Operator. Carbon Intensity API Documentation, 2024. Accessed April 2025.

[48] ASUS. Product Carbon Footprint Report: ROG Strix GeForce RTX 4090. Technical report, ASUS Sustainability, 2023.

[49] Yahui Bai, Yuhe Gao, Runzhe Wan, Sheng Zhang, and Rui Song. A review of reinforcement learning in financial applications. *arXiv preprint arXiv:2411.12746*, 2024. Accessed: 2025-04-29.

[50] Investing.com. Investing.com – stock market quotes & financial news. https://www.investing.com/. Accessed: 2025-04-29.

[51] Aurum Research. Hedge fund industry performance deep dive—full year 2023. https://www.aurum.com/hedge-fund-data/hedge-fund-industry-deep-dive, February 2024.

[52] Hank Tucker. Citadel's $16 billion gain in 2022 makes ken griffin's flagship the top□earning hedge fund ever. *Forbes*, 2023.

[53] IG Group. Our Commissions and Charges on Share Dealing, 2025. Retail tariff page – shows £3 commission once 3+ trades/quarter, £8 otherwise.

[54] Interactive Brokers U.K. Ltd. Trade Stocks Online – Western European Commissions, 2025. Flat £3 / €3 per trade for typical ticket sizes on UK & EU shares.

[55] Cboe Clear Europe. Regulation Fees & Penalties, 2024. Section 3.7 lists the €0.10 per-execution equity clearing fee.

[56] Vantage Markets. Ecn account trading fees, March 2025. Commission $2 per 100k lot round-turn on major FX pairs—representative of low-spread ECN brokers.

[57] B. Weng, M. A. Ahmed, and F. M. Megahed. Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications*, 79:153–163, 2017.