A Foundational Deep Learning Approach to Cryptocurrency Market Forecasting

Cranfield University
Thesis April 2025– September 2025
MSc in Computational and Software Techniques in Engineering

Supervisor: Dr Jun LI
Student : Rodolphe Lucas

Project Reference: *JL-202X-CSTE-FFMD-V1*

Abstract

Cryptocurrencies have emerged as a major financial innovation, challenging traditional monetary systems by offering decentralized and transparent alternatives. Among the core challenges in this domain lies the accurate forecasting of cryptocurrency prices, particularly in the context of short-term, high-volatility environments.

This thesis investigates whether deep learning methods can provide a sustainable edge in forecasting cryptocurrency prices and managing trades on the Bitcoin (BTC/USDT) market. Motivated by the extreme volatility, non-linearity, and regime shifts that characterize digital assets, the work is divided into two parts. In Part I, we frame intra-day prediction as a three-class classification problem (Buy, Hold, Sell) on 4-hour data. An adaptive labelling scheme based on sliding-window distributions and class-imbalance–aware metrics is used to benchmark LSTM, GRU, CNN–RNN, and transformer-based architectures against an XGBoost baseline. While LSTM models achieve around 55 percent accuracy, they struggle to identify Sell signals, and neither binary nor threshold-adjusted labelling improves generalisation.

Part II reformulates the task as multi-horizon regression across three synchronized timeframes (H1, H4, D1) and five prediction horizons (1–48 h). Dedicated LSTM–GLU encoders extract features per timeframe, which are then fused through a weighted gating mechanism. Models are trained on temporal splits covering 2017–2022 and ensembled using validation-weighted averaging. On the unseen test split, the ensemble achieves a *Directional Accuracy* of approximately 89% and an R^2 of 0.873 at the 24h horizon. Forecast calibration proves stronger at the 12–24h horizons than at 1–4h, which can be explained by the 24h stride used in the training procedure that naturally aligns better with mid-range dynamics. A realistic trading simulation converts the probabilistic forecasts into position signals using consensus gating across horizons and evaluates them under downtrend, uptrend, and high-volatility regimes. Positive profit-and-loss (P&L), Sharpe, and Sortino ratios are achievable when execution thresholds and risk parameters are carefully tuned, but performance remains highly sensitive to market regime and trade management. Overall, the results underscore the promise of multi-timeframe deep learning while highlighting the importance of robust evaluation, adaptive labelling, and risk-aware decision rules for real-world deployment.

Keywords: cryptocurrency forecasting, multi-horizon regression, deep learning, multi-timeframe models, Bitcoin, trading strategies

Contents

Introduction

The traditional financial system relies on a complex network of institutions and state mechanisms, including central banks, public treasuries, and commercial banking institutions. These actors regulate the money supply through various monetary regimes, such as the gold standard, reserve currency-backed systems, or purely fiat regimes (Mishkin 2018). However, these frameworks exhibit several structural limitations: exposure to inflation, transaction delays, concentration of monetary power, high operational costs, and lack of transparency (Nakamoto 2008; Narayanan et al. 2016). These shortcomings have encouraged the emergence of digital alternatives, marking a significant disruption in the history of modern monetary systems.

In 2008, within this context, following the subprime crisis, *Bitcoin* emerged — the first functional cryptocurrency, introduced by Satoshi Nakamoto (Nakamoto 2008). It inaugurated a new monetary paradigm, based on a peer-to-peer exchange system, secured by cryptographic algorithms and free from any centralized intermediary. The objective was clear: to restore trust in money through transparency, security, speed, and disintermediation. Despite persistent volatility, the cryptocurrency ecosystem has experienced exponential growth. The global cryptocurrency market capitalization now exceeds 3 trillion dollars (CoinMarketCap 2025), with an expected average annual growth rate of over 11% in the coming years (Statista Research Department 2023).

Nevertheless, forecasting cryptocurrency prices remains a major challenge. Due to their extreme volatility and the ease of access offered by trading platforms such as Binance or Coinbase, cryptocurrencies have attracted a growing number of retail and institutional investors. This influx of participants has further fueled market dynamics and, in turn, motivated an increasing number of academic studies seeking to anticipate price movements. Cryptocurrency prices are influenced by a wide array of exogenous factors: economic policies, regulations, technological innovations, public perception, and sudden macroeconomic events (W. Chen, Xu, and Y. Zhang 2021; Murray and Thomas 2020). Their behavior is highly nonlinear, volatile, and dependent on real-time information. Within the specific field of Bitcoin pricing, several research strands coexist. Some works (Bolt and Oordt 2019; Pagnotta and Buraschi 2018; Biais et al. 2018; Schilling and Uhlig 2019) focus on developing and validating theoretical economic models, while others concentrate more on the empirical study of price formation.

Due to the still-emerging nature of cryptocurrencies, a detailed understanding of the mechanisms driving their price formation remains incomplete. Several studies further suggest that *Bitcoin*, the sector's flagship asset, may represent a distinct asset class, exhibiting hybrid characteristics between currency, commodity, and financial security (Glaser et al. 2014; Dyhrberg 2016; Burniske and White 2017). This uniqueness limits the applicability of traditional predictive signals from classical finance, such as those identified for publicly traded stocks (Green, Hand, and X. F. Zhang 2013).

In the face of such complexity, machine learning methods appear particularly well suited. They enable the exploitation of a wide array of explanatory variables, including technical indicators, volatility measures, sentiment data, and macroeconomic variables. Their ability to model nonlinear relationships and extract patterns in highly noisy environments makes them a powerful tool for analyzing cryptocurrency markets (Cybenko 1989; Goodfellow, Bengio, and Courville 2016). In the broader context of time series forecasting, these techniques have already demonstrated their effectiveness in complex and dynamic domains, such as climate modeling (Rasp and Thuerey 2021), residential energy consumption (Kong et al. 2019), and traditional financial markets (Kara, Boyacioglu, and Baykan 2011). In the case of cryptocurrencies, they enable the simultaneous integration of multivariate, often heterogeneous signals from both internal digital data (OHLCV, technical indicators) and external factors (news, social media, regulation, etc.). This flexibility is further reinforced by the increasing availability of real-time data and powerful processing libraries. However, the wide variety of available models makes comparative evaluation difficult, especially as evaluation protocols differ significantly from one study to another.

It is within this context that the present research is situated. In a first stage, we investigate the task of three-state classification (Buy, Hold, Sell) on a financial time series of Bitcoin, comparing seven distinct deep learning architectures. In a second stage, we propose a novel approach that treats financial time series across three distinct temporal resolutions (hourly, 4-hour, and daily). Dedicated encoders are trained separately for each timeframe, and their representations are fused to predict multi-horizon returns. Our model can be seen as a simplified and specialized version of the Temporal Fusion Transformer (TFT) (Lim et al. 2020), tailored for cryptocurrency markets, in contrast to the more complex and heavier (computationally). Unlike the classification setup, this second part formulates the problem as a ¡regression task, aiming to provide continuous forecasts over multiple horizons.

Chapter 1

Part I Directional BTC (H4) Forecasting : Benchmarking Multiple Deep Learning Models

Abstract

This chapter studies directional classification of Bitcoin at the 4-hour resolution (H4), with three target states—Buy, Hold, Sell—derived from a volatility-adaptive labelling scheme. We evaluate nine models spanning recurrent and hybrid deep networks (LSTM variants, CNN–LSTM, GRU hybrids, TCN, Transformer+GRU), a feed-forward baseline, and XGBoost. Models are trained on the same feature set (OHLCV and technical indicators) and assessed with accuracy and class-wise / macro F1, using both a conventional chronological split and a rolling walk-forward protocol.

Three models consistently stand out: a Standard LSTM, CNN–LSTM (balanced Buy/Hold/Sell trade-off), and XGBoost (best on the minority Sell class). Architectures with the highest accuracy (e.g., Transformer+GRU, LSTM+Attention) achieve this largely via strong Hold detection, but exhibit weak minority-class recall. Walk-forward evaluation yields more conservative and more stable estimates than a single static split, better capturing regime variation.

Finally, we highlight that cross-paper benchmarking remains fragile: datasets, features, label definitions, horizons and metrics often differ, and many studies omit cost-aware backtests (fees, slippage), which can reverse apparent gains—especially for high-turnover rules (Jaquart, Dann, and Martin 2020; Omole and Enke 2024). Taken together, these insights compel the adoption of standardized, transparent evaluation protocols and set the stage for the multi-horizon regression framework developed in Part II.

1.1 Introduction : The Efficient Market Hypothesis

Financial literature is strongly influenced by the *Efficient Market Hypothesis* (EMH), initially formulated by Fama 1970 in the 1970s. This hypothesis posits that asset prices instantly reflect all available information, rendering any systematic prediction or sustainable arbitrage strategy impossible. Under this framework, any attempt to forecast prices based on historical data would be bound to fail against a naïve benchmark (such as the last observed price or a moving average), since markets would instantly correct any inefficiency.

However, this theoretical view was soon challenged, notably by (Grossman and Stiglitz 1980), who demonstrated that a perfectly efficient market would eliminate any incentive to acquire information. As a result, a compromise is necessary: for agents to invest in data analysis, the market must exhibit certain exploitable inefficiencies.

In the case of cryptocurrencies, several elements challenge the strong form of market efficiency:

- their atypical market structure (low regulation, high fragmentation across platforms),
- their extreme volatility and speculative cycles,
- and an unprecedented transparency of flows (on-chain data, tweets, news, etc.).

All these factors suggest that, unlike traditional markets, the crypto market remains partially inefficient — at least in the short term. Several empirical studies (Omole and Enke 2024; Buzcu et al. 2021; Gurgul, Wójcik, and Sliwinski 2025) have indeed highlighted the ability of deep learning models to extract exploitable signals, enabling better performance than passive benchmarks. For instance, the strategy developed by (Buzcu et al.), combining sentiment analysis with a directional LSTM, outperformed the market over a given period (Buzcu et al. 2021). Similarly, (Kehinde and Smith 2025) demonstrate that their Helformer model, when properly calibrated, yields significant net profitability in simulation.

Thus, although market efficiency serves as a theoretical boundary, the specific conditions of the cryptocurrency market — particularly its immaturity — still allow for statistically significant forecasts and potentially profitable strategies.
Nevertheless, drawing definitive conclusions from the literature remains difficult. Studies often differ in their design choices: data frequency and coverage, input features, evaluation horizons, hyperparameter tuning, and even reporting standards vary widely, making direct comparisons problematic.

Moreover, some works suggest that deep learning does not always outperform simpler statistical or tree-based models. For instance, (Omole and Enke 2024) conducted a large-scale comparison of univariate ensemble methods and deep learning approaches across four major cryptocurrencies, showing that LightGBM strategies often yielded higher profitability than deep neural networks for Bitcoin, Ethereum, and Litecoin. Similarly, Chen et al. (W. Chen, Xu, and Y. Zhang 2021) show that macroeconomic and technology-driven determinants, modeled with classical econometric approaches, remain competitive with deep learning models in terms of predictive accuracy.
Beyond raw predictive accuracy, another limitation of the literature lies in the lack of rigorous trading-based evaluation. Only a minority of studies test their forecasts within realistic trading simulations, and when they do, transaction costs, slippage, and liquidity constraints are often neglected. This oversight can heavily penalize high-frequency strategies, where fees accumulate quickly, potentially turning seemingly profitable predictions into unprofitable or even loss-making strategies. Consequently, claims about the "profitability" of deep learning approaches in cryptocurrency forecasting should be treated with caution, unless validated in cost-aware backtesting frameworks.

In this light, forecasting cryptocurrency markets should not be framed as a solved problem, but rather as an evolving field where model design, evaluation protocol, and data context critically shape reported outcomes. It is within this research landscape that our work is positioned. In the first part, we conduct a comparative study of several deep learning architectures on a three-class classification task (*Buy*, *Sell*, *Hold*) using H4 Timeframe Bitcoin data.

The benchmark comprises a diverse set of models, ranging from classical neural networks to more recent architectures. Specifically, we evaluate a standard LSTM, an enhanced LSTM with Attention, a GRU–LSTM hybrid and its attention-augmented variant, as well as a CNN–LSTM designed to capture both local and long-term dependencies. We also include a Temporal Convolutional Network (TCN), a Feedforward Neural Network (FNN), and the non–deep learning baseline XGBoost. Finally, we test a Transformer–GRU hybrid to assess the benefits of attention-based architectures combined with recurrent dynamics

1.2 Related Work

Research on cryptocurrency forecasting spans heterogeneous choices of models, targets, temporal resolutions, and evaluation protocols. As a result, conclusions are often hard to compare because datasets, label definitions, horizons, and metrics differ substantially across studies (Jaquart, Dann, and Martin 2020). In this Part I, we restrict attention to *directional classification* for Bitcoin at the 4-hour (**H4**) resolution, with three states (**Buy**, **Hold**, **Sell**) at a short prediction horizon.

Architectures for Directional Classification :

Since 2017, recurrent models have provided the de facto baseline for crypto direction forecasting. Early results established feasibility with modest yet above-chance accuracy (McNally, Roche, and Caton 2018), and subsequent work repeatedly found *LSTM* and *GRU* to be competitive across major coins (Aggarwal, Kumar, and Sureka 2019; Ji, Bouri, and Roubaud 2019; Jaquart, Dann, and Martin 2020). Building on this, hybrid CNN–RNN designs use convolutions to capture short-range motifs and recurrent layers for longer dependencies; when well tuned, these hybrids frequently improve directional metrics and trading stability. For instance, Omole and Enke (2024) report that a CNN–LSTM can attain high daily hit-rates for BTC, while also noting that tree-based ensembles (e.g., LightGBM) may rival deep nets from a trading perspective. Attention-augmented CNN–RNNs have also been proposed to stabilize intraday predictions and reduce false signals (Peng, Wu, and Zhou 2024). More recently, Transformers have entered the field; however, their advantage over optimized recurrent hybrids remains mixed on typical crypto datasets. Mazinani, Ghods, and Amiri (2024) document cases where a carefully tuned CNN–GRU outperforms Transformer variants on both short and longer horizons, whereas crypto-specific adaptations that combine time-series decomposition with attention (e.g., Holt–Winters + Transformer) show promise in simulation (Kehinde and Smith 2025). Overall, recurrent baselines and CNN–RNN hybrids remain strong contenders for directional tasks on Bitcoin time series.

Task Formulation and Labeling :

The literature bifurcates between regression (predicting a continuous price/return) and classification (predicting direction or discrete states). Directional classification is commonly framed as binary up/down prediction (Tanwar and Srivastava 2021), while multi-class settings (*Buy*/*Hold*/*Sell*) are less frequent yet attractive for rule-based trading. Label construction is far from standardized: some works use a raw sign of future return; others define neutral bands with fixed thresholds; a smaller subset adopts data-driven thresholds (e.g., rolling quantiles) that adapt to local volatility. Because class imbalance (e.g., prolonged bull regimes) can inflate naive accuracy, studies often complement accuracy with precision/recall, F1, AUC, or correlation-style metrics—though practice remains inconsistent.

In our case, we adopt a probabilistic, volatility-adaptive labelling scheme: future returns are normalized by recent volatility, and trading states (*Buy*, *Hold*, *Sell*) are assigned through rolling quantile thresholds, smoothed via sigmoidal mappings. This strategy ensures dynamic thresholds that adjust to market regimes while avoiding rigid cutoffs, producing labels that are both more realistic for trading simulation and compatible with deep learning training.

Forecast Horizons :

In the vast majority of studies on Bitcoin price forecasting, the main objective is to predict the next day's closing price, or at most over a few days. This focus is explained by two fundamental observations: first, deep learning models are better at capturing short-term dependencies, which are often present at hourly granularities (**H1**, **H4**); second, the further out the forecasting horizon, the more the signal-to-noise ratio deteriorates, and prediction errors accumulate—making the task increasingly uncertain.

Nevertheless, some research has explored multi-week forecasting. For example, (Maji and Manisha 2019) attempted to forecast Bitcoin's price 7 days in advance using daily data, comparing an LSTM with a SARIMA model. They observed that forecasting error increases rapidly with the length of the forecast horizon, and while the LSTM could capture the general trend, it struggled to predict the magnitude of fluctuations a week ahead. This supports the intuition that for very long windows, the advantage of deep learning over simpler models diminishes if the signal becomes nearly random.

1.3 Problem Formulation

We consider a cryptocurrency c for which, at each time step t, we have access to a multivariate observation vector $\mathbf{x}_t \in \mathbb{R}^d$, comprising both market data (open, high, low, close prices, volume) and derived technical indicators (such as RSI, MACD, or Ichimoku components).

From a sliding time window of length w, denoted as:

$$\mathbf{X}_t = [\mathbf{x}_{t-w+1}, \mathbf{x}_{t-w+2}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{w \times d}$$

the goal is to predict, at a fixed horizon h, the market signal class associated with time $t + h$ among the following three:

- **Buy**: an upward trend is expected,
- **Hold**: no clear signal or uncertainty,
- **Sell**: a downward trend is expected.

This task is formalized as a *multi-class classification problem*, where each input sequence $\mathbf{X}_t$ is associated with a label $y_t \in \{0, 1, 2\}$ corresponding respectively to the *Buy*, *Hold*, and *Sell* classes.

The learning objective is to approximate a decision function parameterized by a deep learning model f_θ, such that:

$$\hat{y}_t = \arg \max_{k \in \{0,1,2\}} f_\theta(\mathbf{X}_t)_k$$

where $f_\theta(\mathbf{X}_t) \in \mathbb{R}^3$ is the vector of predicted scores for each of the three classes based on the input sequence.

Training is carried out on a dataset of pairs $(\mathbf{X}_t, y_t)$ extracted from a large historical record.

1.4 Data

In this work, we leverage over seven years of historical market data for the cryptocurrency `BTC/USDT`, covering the period from June 2017 to July 2025. This dataset was collected from Binance's public API.

The cryptocurrency market is unique in that it operates continuously, 24 hours a day, 7 days a week, with no interruptions for weekends or public holidays.

1.4.1 Data Description — Features

We use price–volume observations for `BTCUSDT` at a 4-hour cadence, enriched with *technical indicators* widely employed in chart-based trading. These indicators summarize complementary market facets—level and range (OHLCV), momentum (RSI, MACD), and trend/support–resistance structure (Ichimoku)—so that the model can learn not only from raw prices but also from patterns human traders routinely monitor.

Market data (OHLCV). Each 4-hour bar provides the standard five fields:

- **Open**: price at the bar's open (see Fig. 1.1);
- **High** / **Low**: extremes reached during the interval (intrabar range→volatility proxy);
- **Close**: settlement price at bar end;
- **Volume**: traded quantity over the interval (activity/liquidity proxy).

These variables are scaled independently to $[0, 1]$ (min–max per training split).

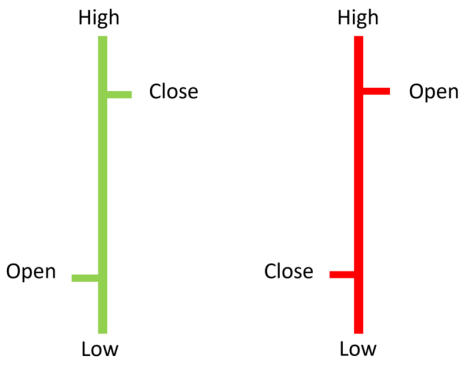


Figure 1.1: 4-hour candlestick (OHLC) — range and close summarize intrabar movement.

RSI (Relative Strength Index). RSI condenses *momentum balance* over a rolling 14-bar window: it compares average gains to average losses, returning a value in $[0, 100]$. High RSI reflects sustained buying pressure; low RSI, sustained selling. By convention[1], RSI > 70 suggests *overbought* (risk of pullback) and RSI < 30 *oversold* (risk of rebound). We use:

- `RSI` (continuous),
- `rsi_oversold` $\equiv \mathbf{1}[\text{RSI} < 30]$,
- `rsi_overbought` $\equiv \mathbf{1}[\text{RSI} > 70]$.

MACD (Moving Average Convergence–Divergence). MACD captures *momentum shifts* via two EMAs: fast (12) and slow (26). The core signal

$$\texttt{MACD} = \text{EMA}_{12} - \text{EMA}_{26}$$

is positive when short-term price is above its longer-term baseline (bullish pressure) and negative otherwise (bearish pressure). A 9-period EMA of MACD forms the `MACD_signal`. Their relationship is interpreted as:

- `MACD` (continuous) and `MACD_signal` (continuous),
- `MACD_hist` = `MACD` − `MACD_signal` (impulse strength),
- `macd_cross` = $\mathbf{1}[$`MACD` $>$ `MACD_signal`$]$ (bullish crossover),
- `strong_macd_bull` = $\mathbf{1}[$`MACD_hist` $>$ median past$]$ (strong impulse).

Intuition: the histogram acts as a *throttle*—larger positive bars ⇒ stronger bullish impulse; larger negative bars ⇒ stronger bearish impulse. We feed normalized continuous versions (`MACD_norm`, `MACD_signal_norm`) plus the two booleans.

Ichimoku. Ichimoku provides a compact view of *trend direction*, *support/resistance zones* (the cloud, or *Kumo*), and *timing* via line crossovers:

- `Tenkan-sen` (`ITS_9`): short baseline (9-bar midprice) — fast trend;
- `Kijun-sen` (`IKS_26`): medium baseline (26-bar midprice) — slow trend;
- `Senkou A/B` (`ISA_9`, `ISB_26`): cloud bounds projected forward — dynamic S/R;
- `Chikou span` (`ICS_26`): close shifted 26 bars back — trend confirmation.

We derive interpretable binary features that the model can combine with prices:

[1] Thresholds 30/70 are practitioner heuristics, not hard rules.

Feature	Meaning / trading intuition
`above_cloud`	Price above Kumo: bullish regime (support below).
`below_cloud`	Price below Kumo: bearish regime (resistance above).
`price_in_kumo`	In-cloud: indecision / congestion.
`kumo_thick`	Cloud thickness (wide = strong S/R / volatility).
`thick_kumo_when_price_inside`	Congestion within a thick cloud (likely whipsaws).
`tenkan_kijun_cross_up`	Fast/slow bullish cross (timing signal).
`tenkan_kijun_cross_down`	Fast/slow bearish cross.
`bullish_cross_above_cloud`	Bull cross in bullish regime (strong confluence).
`price_above_kijun`	Price above medium baseline (trend support).
`price_above_tenkan`	Price above fast baseline (short-term strength).
`chikou_above_price`	Lagging span above price (confirmation).

Scaling and model inputs. Continuous indicators are scaled (min–max to $[0, 1]$) on the *training* split to avoid leakage; binary indicators are in $\{0, 1\}$. The final feature set thus blends raw market state (OHLCV) with trader-facing signals (RSI, MACD, Ichimoku), allowing the classifier to learn how specific momentum/trend configurations map to the three target states (**Buy**, **Hold**, **Sell**).

1.4.2 Data Labelling - Strategy

To annotate the data for supervised learning, we adopted an adaptive strategy that takes into account both market volatility regimes and the dynamic distribution of future returns. The objective is to generate **Buy**, **Hold**, or **Sell** labels at a fixed prediction horizon ($h = 4_H$ in our case). This labelling method is based on an adaptive approach, sensitive to both market volatility regimes and the evolving distribution of future returns.

Future return and local volatility :

For each time t, the future return is defined as:

$$r_{t+h} = \frac{P_{t+h} - P_t}{P_t}$$

where P_t is the closing price at time t.

Dynamic quantiles :

For each time t, we construct a local distribution of future returns using a sliding window (size = 24). The lower and upper quantiles q_{low} and q_{high} are then calculated using an adaptive parameter α_t that varies with volatility. Local volatility is estimated via the rolling standard deviation of past returns over 24 observations, denoted σ_t:

$$\sigma_t = \sqrt{\frac{1}{24} \sum_{i=t-23}^{t} (r_i - \bar{r}_t)^2}, \quad \text{where} \quad r_i = \frac{P_i - P_{i-1}}{P_{i-1}}, \quad \bar{r}_t = \frac{1}{24} \sum_{i=t-23}^{t} r_i$$

The value of σ_t is then normalized and used to adapt the parameter α_t dynamically via bounded linear interpolation:

$$\alpha_t = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot \frac{\sigma_t - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}}, \quad \text{with} \quad \alpha_t \in [0.2, 0.35]$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the 10^{th} and 90^{th} percentiles of the overall observed volatility.

Thus, when volatility is high, the thresholds q_{low} and q_{high} get closer together, reducing the *Hold* zone in favor of sharper decisions. This adaptiveness makes the labelling system more responsive to changing market regimes.

Probabilistic assignment :

Future returns are then transformed into continuous probabilities using sigmoid functions:

$$P_{\text{Buy},t} = \sigma\left(\frac{r_{t+h} - q_{\text{high}}}{s}\right), \quad P_{\text{Sell},t} = \sigma\left(\frac{q_{\text{low}} - r_{t+h}}{s}\right)$$

with $s = 0.005$ a slope parameter controlling sensitivity. The remaining probability is assigned to the *Hold* class:

$$P_{\text{Hold},t} = 1 - P_{\text{Buy},t} - P_{\text{Sell},t}$$

This method produces soft and continuous labels, better suited to the inherent uncertainty of financial markets.

Discretization for training. We convert the soft targets into one–hot labels using a fixed probability cut–off τ=0.47. Concretely,

$$\text{label}_t = \begin{cases} \textit{Buy}, & \text{if } P_{\text{Buy},t} > \tau \text{ and } P_{\text{Buy},t} \geq P_{\text{Sell},t}, \\ \textit{Sell}, & \text{if } P_{\text{Sell},t} > \tau \text{ and } P_{\text{Sell},t} \geq P_{\text{Buy},t}, \\ \textit{Hold}, & \text{otherwise.} \end{cases}$$

This rule promotes decisive labels only when a side is sufficiently confident, avoiding brittle fixed bands on raw returns that do not adapt to changing crypto volatility[2].

Empirical sanity check (signal quality) :

To gauge whether the probabilities carry actionable information, we ran a deliberately simple long-only backtest: open a position when $P_{\text{Buy},t} > \tau$ and close it when $P_{\text{Sell},t} > \tau$ (no leverage, fees ignored). Over ∼7 years of BTC history this yields:

- average return per trade: +2.38%,
- annualized Sharpe: 7.54,
- max drawdown: −4.97%,
- # buy signals: 5247; # effective trades: 2774.

These figures should be read as *upper bounds*: because the threshold and signal definitions were calibrated using future returns during label construction, they are not realistic for live deployment. Our goal here is solely to : (i) select a cut–off that yields a well-spread label distribution and (ii) verify that the induced *Buy/Sell/Hold* signals have plausible trading semantics. Robust out-of-sample evaluation (with walk-forward tuning, fees/slippage, and no look-ahead) is reported later.

Choice of an imbalanced dataset :

Finally, the discretization applied to $(P_{\text{Buy}}, P_{\text{Sell}}, P_{\text{Hold}})$ uses a threshold of 0.47 to frame the problem as 3-class classification. Although this threshold does not perfectly balance the classes, it offered a good trade-off between performance and strategy robustness. The final class distribution is:

Buy: 31.10% Hold: 52.47% Sell: 16.43%

This imbalance is a deliberate choice, motivated by several factors:

(1) Signal quality first: A higher threshold ensures that only the most decisive *Buy* and *Sell* signals are kept, avoiding uncertain zones. This aims to favor decision relevance over frequency.

(2) Market behavior realism: Bitcoin evolves in highly asymmetric regimes, with long periods of consolidation or slow uptrends, and very few sustained downtrends. Artificially balancing labels would not reflect the structural nature of the market.

(3) Behavioral analysis of models: Keeping *Hold* as the majority class allows us to test the model's ability to detect rare but significant events. This serves as a good indicator of contextual sensitivity, which is critical in volatile financial environments.

(4) Conservative approach in algorithmic trading: From a risk-management perspective, it's preferable to remain neutral (Hold) than to enter uncertain trades. This imbalance helps reduce unnecessary actions and encourages a more robust and cautious strategy.

[2]In practice, τ was chosen *ex post* to obtain a balanced class mix and economically meaningful signals. This tuning uses information not available in live trading; it serves only to calibrate the labelling scheme.

Final structure of the dataset :

After the probabilistic labelling, discretization, and data cleaning steps, we obtain a final dataset containing 16,875 samples. Each sample represents a time step t, associated with 23 explanatory variables derived from technical indicators (either normalized or binary), along with a short-horizon class label at time $t + h$ (here $h = 1$).

The entire dataset is structured into inputs $X \in \mathbb{R}^{T \times D}$ and outputs $y \in \{0, 1, 2\}^T$, where T is the total number of time steps (samples), and $D = 23$ is the number of input features. A simplified illustration of the data structure is shown below:

Time	`open_norm`	`RSI_norm`	`MACD_norm`	`above_cloud`	...	`chikou_above`	...	**target_class**
$t-12$	0.0231	0.5703	0.4874	1	...	1	...	Hold
$t-8$	0.0245	0.6021	0.4938	1	...	1	...	Buy
$t-4$	0.0250	0.6500	0.5021	0	...	1	...	Hold
t	0.0262	0.7152	0.5123	0	...	0	...	Sell

Temporal splitting is performed without shuffling (time order preserved), using the following proportions:

Training (70%), Validation (15%), Test (15%)

1.5 Experiments

In this section, we introduce our experimental setup and results in detail. We also report the performance of several baseline models to demonstrate the effectiveness of our proposed approach. In addition to the final outcomes, we include ablation study results to highlight the influence of specific components in our model. For completeness, the full architectures of all tested models are provided in the github.

1.5.1 Computational Environment and Tools

The implementation and experimentation of all models were conducted using Python. The deep learning architectures were developed with `TensorFlow` and `Keras`, leveraging `Keras Tuner` for hyperparameter optimization. Training and evaluation were performed on a GPU environment provided by Google Colab, which offers a CUDA-enabled GPU.

The XGBoost classifier, used as a baseline non-sequential model, was trained using CPU only.

All scripts were developed and executed in Jupyter Notebook environments under Google Colab or locally using Python 3.10.

1.5.2 Evaluation Metrics

To assess the performance of our classification models (Buy, Hold, Sell), we employ several standard metrics from the field of supervised learning. Since the data is imbalanced, it is crucial to go beyond simple accuracy and use metrics that reflect the quality of the predictions for each class.

Confusion Matrix Terminology. For a given class (e.g., *Buy*), predictions can be summarized as:

- **True Positives (TP):** The number of times the model correctly predicted the target class.
- **False Positives (FP):** The number of times the model predicted the class, but it was actually a different one.
- **False Negatives (FN):** The number of times the model failed to predict the class when it was the correct one.
- **True Negatives (TN):** The number of times the model correctly predicted another class.

These quantities form the basis for the following evaluation metrics:

Accuracy. Accuracy measures the proportion of correct predictions over the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

It gives a global indication of performance but can be misleading when the dataset is imbalanced, as it may favor the majority class (e.g., *Hold*).

Precision. Precision quantifies how many of the predicted positive instances for a given class were actually correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high precision indicates that when the model predicts a class (e.g., *Buy*), it is often correct — this is important in trading to avoid false Buy signals.

Recall. Recall (or sensitivity) evaluates the model's ability to detect all true positive cases:

$$\text{Recall} = \frac{TP}{TP + FN}$$

A high recall means the model is capable of identifying most opportunities (e.g., real Buy signals), at the risk of more false alarms if precision is low.

F1-score. The F1-score is the harmonic mean of precision and recall, offering a balanced measure when there is a trade-off between the two:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is particularly useful in contexts like ours where we seek both accurate and complete signal detection without being misled by class imbalance.

Each metric is computed per class (Buy, Hold, Sell), and macro- or weighted-averaged to provide a single global score across all classes. For real-world trading scenarios, F1-score is a preferred metric as it captures both the quality and completeness of signal prediction.

1.5.3 Rolling Walk-Forward

In this study, we experimented with an alternative training method: *Rolling Walk-Forward*. This approach, well-known in financial literature, consists of training a model on a sliding window of historical data, then validating and testing it immediately on subsequent periods. At each iteration, the window is shifted forward in time, and the process is repeated on a more recent temporal segment—thus simulating the continuous deployment of a strategy in real-world conditions.

This choice is driven by several key motivations. First, the *Walk-Forward* method strictly respects temporal causality: the model only sees the past to predict the future, which eliminates any form of information leakage. Second, this approach closely mirrors the behavior of an automated trading system operating in a dynamic environment. Indeed, a model deployed in a real-world setting is continuously updated using the most recent data; the *Walk-Forward* method effectively reproduces this iterative and adaptive process. Moreover, this method provides a more robust evaluation framework than a simple static split, as it generates a sequence of performances over various temporal windows. This allows for a better assessment of model stability, resilience to market regime shifts (bullish, bearish, sideways), and the detection of potential performance drift over time.

This experiment was also conducted with a comparative objective: it allows us to contrast the performance obtained with that of a more conventional static temporal split, and thus assess the actual relevance of such an evaluation strategy for financial time series.

Configuration :

In our case, each rolling window spans approximately seven years of 4H data and is structured as follows:

- **3 years** for training,
- **6 months** for validation,
- **3 months** for testing,
- with a **1-month** forward step at each iteration.

Each iteration produces an independent sub-model, evaluated consistently with previous ones. The aggregated performances across all windows are then analyzed (F1-score, precision, recall, confusion matrices), allowing for a more detailed and temporally realistic assessment of the model's predictive capabilities.

This rolling scheme also ensures that the model is evaluated across a broad spectrum of market regimes. By construction, the successive windows naturally cover phases of strong uptrends, prolonged downtrends, and periods of low-volatility sideways movement. It may even include rare "black swan" events, such as crashes or sudden liquidity shocks, which are crucial stress-tests for forecasting models. This diversity of regimes provides a more faithful measure of robustness than evaluation on a single static slice of history.

1.6 Results

Table 1.1: Performance comparison of tested models on the test set (Accuracy and F1-scores by class).

Model	Accuracy	Macro F1	Buy F1	Hold F1	Sell F1
Standard LSTM	0.5051	**0.3838**	0.3685	0.6405	0.1424
LSTM + Attention	0.5566	0.3171	0.2486	**0.7028**	0.0000
GRU + LSTM	0.4600	0.3100	**0.6000**	0.3400	0.0000
GRU + LSTM + Attention	0.5075	0.3380	0.3543	0.6445	0.0151
CNN-LSTM	0.4834	0.3669	0.3787	0.6106	0.1113
Temporal Conv. Network (TCN)	0.5407	0.2977	0.1332	0.7023	0.0577
Transformer + GRU	**0.5621**	0.2985	0.1844	0.7110	0.0000
FNN	0.4798	0.3215	0.3578	0.6066	0.0000
XGBoost	0.4323	**0.3880**	0.4132	0.5392	**0.2116**

1.6.1 Model Comparison

We evaluated the performance of several deep learning and machine learning models trained on the same feature set and labeling strategy. Each model was assessed using the accuracy, precision, recall, and F1-score on the test set. Due to the class imbalance in the dataset (with *Hold* being the majority class), the F1-score and class-wise behavior provide more insightful signals than global accuracy alone.

Headline results :

The **Transformer+GRU** and **LSTM+Attention** reach the highest accuracies (56.21% and 55.66%), but this comes largely from very strong recall on the majority *Hold* class (F1 $\approx$ 0.71 and 0.70) while nearly missing *Sell* (F1 = 0). In contrast, **XGBoost** delivers the top *Macro F1* (0.388) and the best *Sell* F1 (0.212), indicating more balanced detection across classes despite a lower overall accuracy. The **Standard LSTM** is the strongest among recurrent baselines on Macro F1 (0.384), with non-trivial *Sell* detection (0.142). **CNN-LSTM** also offers a comparatively even profile (Macro F1 0.367) with meaningful, though still modest, *Sell* recognition.

Sequential models with attention (Transformer+GRU, LSTM+Attn) tend to "play it safe" by predicting *Hold* more often—good for accuracy but weak for actionable *Buy/Sell*. The **GRU+LSTM** variant peaks on *Buy* (F1 = 0.60) but collapses on *Sell* (0.00), pointing to asymmetric sensitivity to upward vs. downward moves. **TCN** shows the same bias: high *Hold* F1 (0.702) but poor *Buy/Sell*. By contrast, **XGBoost** uncovers non-linear interactions in the tabular feature space that help detect rarer *Sell* events, improving macro-level fairness (Macro F1) even if sequence dynamics are not modeled explicitly.

Note on class imbalance :

Directional labelling on trending markets naturally induces imbalance (dominant *Hold*, fewer *Sell*). Models maximizing accuracy drift toward the majority class, inflating headline scores but under-serving decision-critical tails. In particular, for noisy financial time series, such imbalance may cause classifiers to predominantly predict the majority class, regardless of the underlying input features. For this reason, *Macro F1* and per-class F1 are the right lenses here, and they reveal a clearer ranking (XGBoost Standard LSTM CNN-LSTM).

Conclusion. Our results echo (Bouteska and Zoghlami 2024): tree ensembles (e.g., XGBoost/LightGBM) can match or surpass deep models on certain crypto tasks, especially when the objective rewards balanced detection rather than majority-class accuracy. At the same time, the **Standard LSTM** remains a robust sequential baseline with competitive macro performance and non-zero *Sell* recall, making it a strong candidate in sequence-aware ensembles. Beyond the intrinsic volatility and regime shifts of the H4 series, the feature space mixes binary (rule-like) indicators and continuous technical signals; this heterogeneity is non-trivial for sequence networks that implicitly favor smooth, standardized inputs and may underuse sparse boolean cues. Tree-based models, by contrast, natively accommodate mixed types and non-linear interactions with fewer distributional assumptions, which helps explain their stronger macro-F1 and *Sell* detection here. In the following sections, we therefore examine (i) feature importance via XGBoost gain/cover, (ii) the impact of label discretization thresholds on class balance and tails, and (iii) walk-forward validation across distinct market regimes (uptrend, downtrend, high volatility).

1.6.2 Feature Importance via XGBoost Gain

To better understand which input variables contribute most to classification performance, we trained a multiclass XGBoost model using the complete feature set. This set includes normalized continuous indicators (e.g., RSI, MACD), engineered features over sliding windows (e.g., mean, minimum, slope), and boolean technical patterns (e.g., `rsi_oversold`, `macd_cross`).

We relied on the **gain** metric, which quantifies the average improvement in the loss function whenever a feature is used in a split, to assess feature importance. The chart in Figure 1.2 displays the most decisive features according to this criterion.

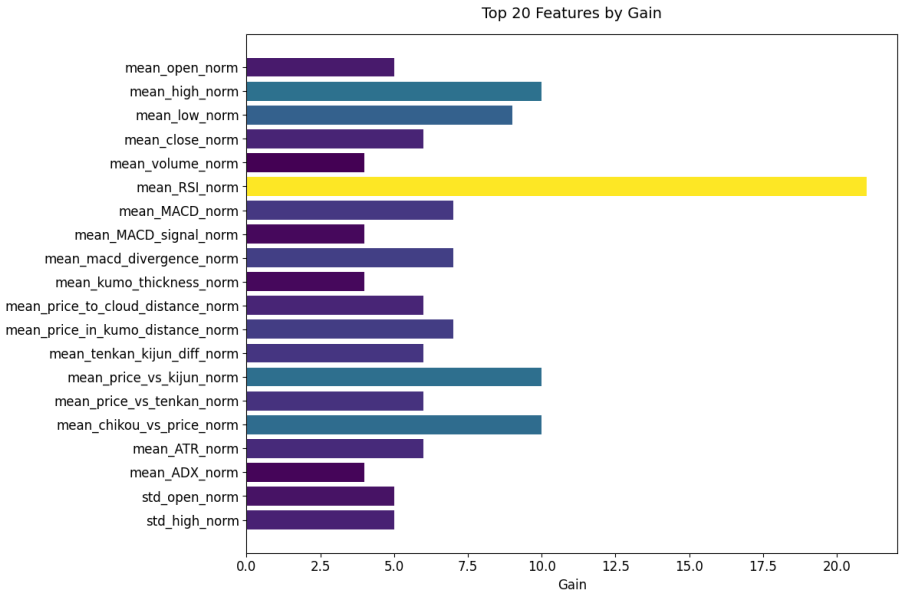


Figure 1.2: Top features ranked by XGBoost Gain.

Key Observations. The most influential features span several families of indicators:

- **RSI-related variables**: features such as `mean_RSI_norm` and `min_RSI_norm` rank among the most informative. These capture both overall trend strength and local oversold conditions, highlighting the importance of RSI-derived dynamics.
- **Ichimoku-based features**: variables measuring distances between price and Ichimoku lines (e.g., `min_tenkan_kijun_diff_norm`) appear prominently, confirming that Ichimoku's support/resistance structure helps the model to delineate meaningful signal zones.
- **Engineered price statistics**: features such as `slope_low_norm` and `mean_high_norm` reflect directional slope and average resistance levels, demonstrating that engineered price trends are informative even in non-sequential models.
- **MACD and momentum**: indicators capturing MACD variation (e.g., `min_MACD_norm`) remain relevant, pointing to the persistent role of momentum in directional signal detection.
- **ADX and ADR**: the *Average Directional Index (ADX)*—a measure of trend strength—and the *Average Daily Range (ADR)*—a proxy for volatility magnitude—also contribute meaningfully. Their presence confirms that both directional conviction (trend persistence) and volatility regimes play a critical role in shaping predictive signals.

Conclusion. This feature importance analysis shows that the model exploits a balanced mix of aggregated statistics (e.g., mean, minimum, slope) and conventional technical indicators (RSI, Ichimoku, MACD), while also benefiting from trend-strength and volatility measures (ADX, ADR). The relative deprioritization of raw volume-based features suggests that, in this dataset, volatility-adjusted and momentum-based variables carry greater explanatory power.

Overall, these results highlight the value of combining traditional oscillators, price-engineered descriptors, and volatility-sensitive indicators. This naturally motivates the integration of **Bollinger Bands** in the second stage of the study, as they simultaneously capture both volatility expansion/contraction and relative price positioning—two dimensions that appear critical for stable directional forecasting.

More broadly, the interplay between volatility and trend confirmation is crucial: the *Average True Range (ATR)* quantifies raw volatility, *Bollinger Bands* contextualize this volatility relative to a moving average, and the *Average Directional Index (ADX)* evaluates whether such volatility reflects a genuine, exploitable trend or merely noise. Together, these complementary indicators provide a richer representation of market dynamics, reinforcing the case for their inclusion in the subsequent feature engineering stage.

1.6.3 Effect of Classification Classes

In this section, we investigate how class definitions influence the performance and usability of our models—particularly the trade-offs between using a 3-class versus a 2-class classification scheme, and the effect of threshold tuning on label distribution. The model used for the comparison is CNN + LSTM.

Label Balance via Threshold Tuning. Our original labelling approach, based on a relatively high threshold (0.47), led to a dataset dominated by the *Hold* class. To reduce this imbalance and improve the representation of *Buy* and *Sell* signals, we tested a lower threshold of 0.36. This adjustment yielded the following class distribution:

Buy: 42%, Hold: 35%, Sell: 22%

This configuration appears more balanced at first glance and theoretically allows the model to better capture directional signals. However, when evaluated on the test set, the results were disappointing. The model exhibited extremely poor recall on the *Buy* class and completely failed to detect any *Sell* signals:

Table 1.2: Classification report with threshold = 0.36

Class	Precision	Recall	F1-Score	Support
Buy	0.5385	0.0476	0.0875	1029
Hold	0.3923	0.9725	0.5591	983
Sell	0.0000	0.0000	0.0000	516
Accuracy		0.3975		
Macro F1		0.2155		

The *Hold* class dominates again, absorbing most predictions. Despite a better class balance in the dataset, the model heavily overfits to *Hold* (recall = 97.25%), failing to generalize well on more subtle transitions. This confirms that improving label balance through threshold reduction does not automatically translate into better model performance.

Conclusion :

This experiment demonstrates that improving dataset balance via fixed threshold adjustment does not guarantee better generalization or trading performance. In fact, lowering the threshold may introduce excessive labeling noise, making the model's learning task harder and ultimately less effective.

Given that our dataset spans over seven years of market data (from 2017 to 2025), during which macroeconomic conditions, investor behavior, and market structure have evolved significantly, a static threshold is unlikely to capture these shifts adequately.

It could be useful to try *dynamic thresholding schemes* and *probabilistic filtering mechanisms* that adapt to changing volatility regimes, economic cycles, or market phases. Such approaches may offer a better trade-off between signal frequency and reliability, especially in long-term financial time series where structural breaks and behavioral shifts are common.

From 3-Class to 2-Class Classification :

During our initial research phase, we also explored the possibility of reframing the problem as a binary classification task rather than a 3-class problem. This would involve distinguishing between actionable signals (e.g., *Buy*) and all others, instead of attempting to classify each instance into *Buy*, *Hold*, or *Sell*.

As discussed by (W. Chen, Xu, and Y. Zhang 2021), extending a model from two to three output classes often leads to a sharp performance drop. This difficulty arises from the lack of clear separability in economic and technological signals, which rarely provide distinct boundaries between categories in financial forecasting tasks.

The problem lies in defining a robust and interpretable threshold: what return or momentum should be considered "positive" enough to be classified as *Buy*, or "negative" enough to be *Sell*? Many situations fall into a grey zone that could either be a weak signal or mere market noise. As a result, training a model to learn the distinction between a truly "neutral" context and a subtle but significant trend becomes highly unstable and noisy.

Similarly, in our context of short-horizon crypto forecasting (e.g., $h = 4H$), asserting with high certainty that a price should move upward or downward is often inconsistent with the underlying market behavior. Hence, we hypothesized that a binary classification—distinguishing only between *Hold* and *Buy*—might yield more coherent signals for certain use cases.

Binary Classification Experiment. To test this hypothesis, we constructed a simplified binary classification setup using the same CNN + LSTM architecture as in previous experiments. In this version, the *Sell* label was removed and all non-*Buy* cases were labeled as *Hold*. This reformulation allows us to evaluate whether the model becomes more effective when the learning objective is focused solely on detecting strong *Buy* signals (i.e., entry points), without needing to distinguish ambiguous or weak signals such as *Sell* or trendless phases.

The new dataset, generated with a fixed threshold of 0.47, yields a label distribution of:

$$\text{Buy: } 34.06\%, \quad \text{Hold: } 65.94\%$$

While this setup is better balanced than the original 3-class configuration, the classification results remain limited. On the validation set, the model achieved an accuracy of 62.41%, with a macro F1-score of 0.5303:

- **Buy F1-score:** 0.3205 (Precision: 0.3993, Recall: 0.2676)
- **Hold F1-score:** 0.7402 (Precision: 0.6882, Recall: 0.8006)

These results again illustrate the model's difficulty in correctly identifying *Buy* signals, with relatively low recall and F1-score for this class. Despite the simplified binary setup, the model tends to favor the *Hold* prediction due to its majority share, reinforcing the idea that the distinction between *Buy* and *Hold* remains inherently difficult to learn.

Conclusion. This experiment confirms that reducing the output space to two classes does not solve the underlying issue. The difficulty lies in both the quality of the labeling process and the training-validation-test split, which spans market phases that may exhibit radically different behaviors. The underlying challenge is structural: over a dataset spanning more than 7 years, the Bitcoin market has undergone distinct macroeconomic, behavioral, and structural shifts. This regime heterogeneity makes the learning task particularly difficult for static classifiers.

In particular, the model often confuses *Hold* and *Buy*, which may stem from:

- The fuzziness of the labeling strategy in transition zones (where price changes are moderate),
- The lack of granularity in volatility-aware segmentation,
- Or the insufficient temporal alignment between training and deployment contexts.

Moreover, predicting a trading signal over a short horizon of 4 hours ($H + 1$) proves to be a very challenging task, regardless of the model used. At such a fine temporal scale, market noise, microstructure effects, and short-lived anomalies significantly affect predictive reliability. As a result, we argue that future work should include:

- A confidence estimation mechanism associated with each prediction (e.g., via prediction entropy, softmax calibration, or uncertainty-aware modeling),
- A multi-timeframe architecture that aggregates confirmations from different temporal horizons (e.g., H1, H4, Daily) to strengthen the reliability of entry or exit signals.

These observations ultimately justify the adoption of a more realistic training scheme, namely the *Rolling Walk-Forward Validation*, which we explore in the next section.

Conclusion. These two experiments demonstrate that improving dataset balance via fixed threshold adjustment or using a binary classification scheme does not guarantee better generalization or trading performance. In fact, lowering the threshold may introduce excessive labeling noise, while collapsing the problem into two classes can oversimplify complex market behaviors, reducing the model's ability to capture meaningful directional signals.

In practice, the core difficulty lies in the heterogeneity of market regimes across the Bitcoin dataset, which spans more than seven years. During this time, structural shifts—such as volatility surges, macroeconomic shocks, or speculative bubbles—have altered the statistical properties of the series. Designing a single labeling strategy that performs consistently across all these regimes remains a challenging task.

Furthermore, many models struggled to differentiate between *Hold* and *Buy*, often favoring the former due to its dominant presence in the training data. This misclassification undermines the precision of entry signals, which are critical in trading applications.

In future work, we plan to explore *dynamic thresholding schemes* and *probabilistic filtering mechanisms* that adapt to changing volatility regimes, economic cycles, or market phases. Such approaches may offer a better trade-off between signal frequency and reliability, especially in long-term financial time series where structural breaks and behavioral shifts are common.

We also observed a notable discrepancy between validation and test performance metrics in this experiment. This gap is likely due to temporal distance: the validation data precede the test data by several months, potentially placing them in different market regimes. In time series problems, it is essential to recognize that market dynamics can shift abruptly—what works during a bullish or sideways phase might not generalize to a bearish period.

To mitigate this issue, *time-series aware cross-validation strategies* should be considered. For example, structuring the validation sets to be temporally closer to the test windows can help assess model robustness in more realistic deployment conditions.

In this regard, we implemented a more realistic training paradigm based on *Rolling Walk-Forward Validation*, which will be detailed and analyzed in the following section.

1.6.4 Effect of Rolling Walk-Forward

To better evaluate the temporal robustness of our model, we implemented a *Rolling Walk-Forward* validation scheme on a deep LSTM architecture. This method simulates realistic model deployment conditions by retraining and testing the model on successive time windows. A total of 48 overlapping windows were used, each covering approximately three years of training data, six months of validation, and three months of test data, with a step of one month between iterations.

Model Architecture. The tested model is a two-layer LSTM network designed to extract temporal dependencies from sequences of 24 time steps (4-hour granularity), with 23 input features. The architecture includes batch normalization, dropout regularization (30%), and a softmax output layer for 3-class classification. Class imbalance is handled dynamically in each window via class weights computed from the training data.

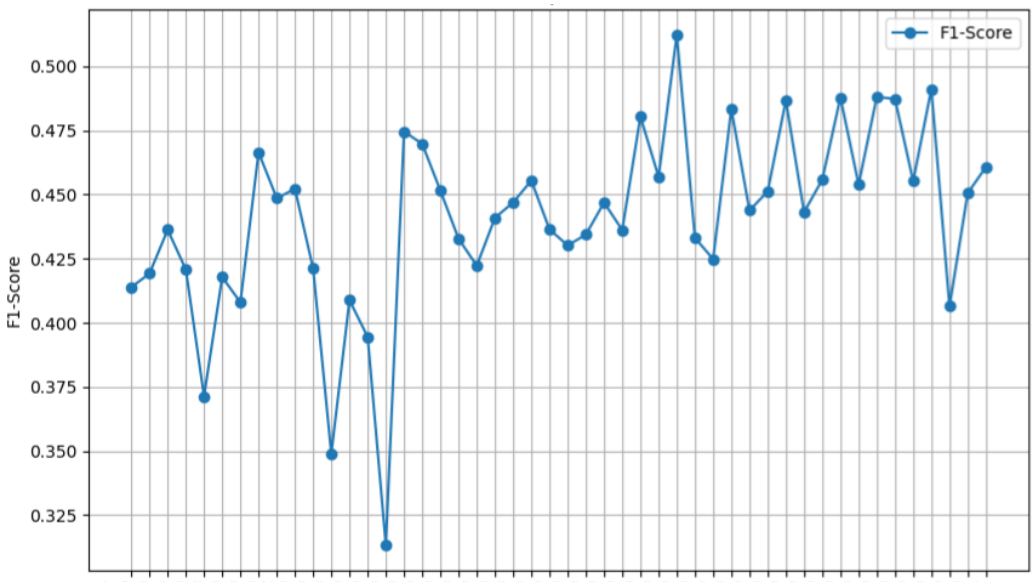


Figure 1.3: Macro F1-score per test window using Rolling Walk-Forward evaluation

Performance Dynamics. The chart in Figure 1.3 shows the evolution of the macro F1-score across the 48 test windows. The overall average macro F1-score is **0.441**, with noticeable variations across market regimes and time periods. The best window achieved an F1-score of **0.512**, while the worst dropped to **0.313**, confirming that temporal distribution shifts significantly impact predictive performance.

Statistical Summary. A detailed statistical summary of the F1-score, precision, and recall across all windows is presented in Table 1.3. While the *Hold* class dominates in both recall and F1-score, the *Buy* and *Sell* classes remain significantly harder to predict—particularly the *Sell* class, with an average F1-score of just **0.084**.

Table 1.3: Rolling Walk-Forward — Metric Summary Across 48 Windows

Metric	Mean	Median	Max	Min
F1-Score (Macro)	0.441	0.445	0.512	0.313
Precision (Macro)	0.444	0.442	0.514	0.375
Recall (Macro)	0.477	0.482	0.583	0.355
Buy F1-Score	0.339	0.362	0.473	0.090
Hold F1-Score	0.598	0.626	0.735	0.270
Sell F1-Score	0.084	0.084	0.228	0.000

Best Performing Window. The highest performance was observed in `window_31`, with a macro F1-score of **0.512**. While *Hold* achieved high recall (0.75) and F1-score (0.694), the *Buy* and *Sell* signals remained modest with F1-scores of 0.337 and 0.133, respectively. This highlights that even under optimal temporal conditions, minority class prediction remains a challenge.

Insights. The results confirm that rolling validation provides a more nuanced view of model stability and adaptation across market phases. It also highlights the need for strategies such as temporal ensembling, adaptive retraining, or regime-aware architectures to address variability in class behavior—especially for critical signals like *Sell* that carry high financial consequences.

However, it is important to note that the *Sell* class remains underrepresented in our dataset, both in terms of raw frequency and labeling sensitivity. Furthermore, the overall trading strategy implemented so far has been designed with a long-only bias, focusing primarily on the identification of buy signals. This structural asymmetry limits the model's exposure to sell opportunities and may partially explain its low performance on this class. Future work should consider rebalancing the training signal distribution and incorporating dedicated strategies for short-side detection.

1.6.5 Discussion

The current results suggest that it remains difficult to build a robust and actionable trading strategy solely based on the trained models. Several key limitations were observed throughout the experimentation process:

- **Data splitting challenges:** The fixed temporal split introduces a major source of bias, as the test data may belong to market regimes (e.g., bull markets, consolidations, or bear phases) that are poorly represented or absent in the training data. This mismatch significantly affects model generalization.
- **Market regime heterogeneity:** The results indicate that a single model may struggle to capture dynamics across all market phases. A promising direction would be to train and deploy models specifically adapted to different market regimes (e.g., regime-aware modeling or context-switching architectures).
- **Motivation for Walk-Forward validation:** These observations validate the use of a *Rolling Walk-Forward* validation strategy, which better reflects real-world deployment and provides more reliable performance metrics across time.

- **Interpretability difficulties:** Understanding the influence of technical indicators on model predictions remains challenging. While some features (e.g., RSI, MACD) appear informative in tree-based models, their roles in deep learning architectures are less transparent and require further analysis (e.g., SHAP values, attention scores).
- **Forecasting complexity at short horizons:** The prediction task is especially difficult at a 4-hour horizon due to high market noise and limited signal strength. Small price movements may not provide enough separation between classes, making directional forecasting less reliable in the short term.

1.7 Conclusion

Across all candidate architectures, three families consistently emerged as the most reliable on our H4 three-class classification task. Standard LSTMs achieved the strongest macro-F1 among deep models, including a non-zero recall on *Sell*, thereby validating their role as a dependable sequential baseline. CNN–LSTMs demonstrated the most balanced distribution across *Buy/Hold/Sell*, a property of particular importance in trading applications where rare but actionable signals must not be overlooked. Finally, XGBoost, despite lower overall accuracy, delivered the highest F1 on *Sell*, corroborating evidence that tree-based ensembles can match or surpass deep neural architectures in crypto settings where minority-class detection is critical.

In contrast, models with the highest reported accuracies (e.g., Transformer+GRU, LSTM+Attention) systematically over-predicted the majority *Hold* class, thereby inflating global accuracy metrics at the expense of *Buy/Sell* sensitivity. This artefact reflects the difficulty of the labeling regime and underlying class imbalance, rather than superior signal extraction.

The broader literature review reveals why benchmarking remains intrinsically difficult. On the input side, heterogeneity arises from variation in data sources and horizons, feature engineering practices (OHLCV-only vs. enriched with technical, sentiment, or macroeconomic features), preprocessing steps, and leakage controls. On the output side, studies diverge in prediction targets (prices vs. returns, linear vs. logarithmic), label construction (fixed vs. adaptive bands), and reported metrics (RMSE/MAE for regression; accuracy/F1/AUC for classification; occasionally financial metrics such as Sharpe ratios). Critically, only a minority of works perform cost-aware backtesting; ignoring trading frictions such as fees, slippage, and execution constraints risks turning seemingly profitable models into strategies with negative net returns. Differences in hyperparameter tuning, early stopping, and seed control further amplify result dispersion, explaining reversals where tree ensembles rival deep architectures. As emphasized in Jaquart, Dann, and Martin (2020), transparent and standardized protocols are essential for comparability.

However, our empirical findings also make it clear that the models developed in Part I are not sufficient to establish a viable trading strategy. Their predictive power remains too limited once realistic market frictions are taken into account. This limitation motivates the transition to Part II of the thesis, where the focus shifts toward a multi-timeframe, multi-horizon regression framework specifically designed to generate calibrated return forecasts and to evaluate their true economic value through rigorous, cost-aware backtesting.

Chapter 2

Part II From Forecasting to Execution : Multi-Horizon Model for Trading Strategies

Abstract

Part I focused on single-horizon, direction-based prediction. Part II widens the scope to a *multi-horizon* setting: we use three separate timeframes (H1, H4, D1) and train a model to predict future returns expressed through a bounded tanh(return) target. The three views are combined to produce short- and medium-horizon forecasts that are consistent across timescales.

To support robust generalization, the data are organized into four chronological splits. We train one model per split and use each model to evaluate the held-out test split under strict out-of-sample conditions. This protocol yields stable results without relying on future information.

Finally, we assess practical usefulness through a realistic trading simulation (fees included) across three market phases—downtrend, uptrend, and high volatility. The system can generate positive P&L in certain regimes and configurations, suggesting genuine promise. However, effectiveness depends on risk-aware choices (e.g., confidence thresholds, position sizing, and horizon emphasis), which should be tuned before any real-world deployment.

2.1 Introduction : James–Stein Paradox

Part I examined three-state directional classification on **H4** data (*Buy/Hold/Sell*). While several models (e.g., Standard LSTM, CNN–LSTM, XGBoost) achieved competitive macro performance, the single-horizon framing proved insufficient for trading: class imbalance, volatility clustering, and regime sensitivity consistently pushed models toward conservative *Hold* decisions, thereby limiting cost-aware deployability.

In Part II, we shift perspective by adopting a *multi-horizon regression* setup that predicts log-normalized returns jointly at $\{1h, 4h, 12h, 24h, 48h\}$. A central innovation lies in the *multi-timeframe design*: dedicated encoders process **H1**, **H4**, and **D1** sequences (lengths $100, 25, 30$, respectively), whose latent representations are subsequently fused before feeding horizon-specific heads. This design explicitly mirrors the **Top-Down Analysis** method widely practiced in trading: daily charts (**D1**) establish strategic trend context, H4 sequences capture intermediate setups, and H1 sequences refine tactical entry timing. By aligning with this practitioner logic, the model benefits from complementary market information across temporal granularities.

The theoretical underpinning draws on the James–Stein effect: when estimating multiple uncertain quantities, joint modeling reduces expected error compared to independent estimation (James and Stein 1961). In time-series forecasting, this principle motivates multi-horizon architectures such as the Temporal Fusion Transformer (Lim et al. 2020), where shared representations *borrow strength* across horizons while still allowing horizon-specific specialization. Our framework extends this rationale by combining both multi-horizon and multi-timeframe structure, aiming to stabilize forecasts and enhance their trading utility.

Finally, to ensure robustness across regimes, we deliberately train **four models on four distinct temporal splits**, each covering heterogeneous phases of the market (uptrends, downtrends, volatility clusters, and even black-swan events). This design choice allows the evaluation to reflect not only model accuracy, but also resilience to structural shifts and rare shocks that typify crypto markets.

Beyond pure predictive accuracy, we will then subject the resulting models to realistic backtesting on 2024–2025 out-of-sample data, explicitly accounting for transaction fees, in order to evaluate their effective trading performance.

2.2 Related Work

Problem framing. In cryptocurrency forecasting, a major strand treats the task as *regression*—predicting a continuous target such as a future price level or return. Daily price/return regression has been studied with deep and non-deep baselines and evaluated via MSE/RMSE/MAE (Bouteska and Zoghlami 2024). Several works argue for modeling *returns* (percent or log change) rather than levels to stabilize scale and emphasize relative movement (M. Chen et al. 2020). Yet low average error does not guarantee trading usefulness when the sign or timing is systematically off, underscoring the need to align targets and metrics with downstream decisions.

Forecast horizons and joint estimation. Most crypto studies concentrate on short horizons (minutes–hours or one day ahead) where technical inputs carry higher signal-to-noise (Jaquart, Dann, and Martin 2020; Peng, Wu, and Zhou 2024); errors typically increase with horizon length (Albariqi et al. 2020; Mazinani, Ghods, and Amiri 2024). This motivates *multi-horizon* formulations that estimate several future targets jointly, borrowing statistical strength across related predictions—a perspective resonant with James–Stein shrinkage for vector parameters (James and Stein 1961). Architectures designed for this setting, such as the Temporal Fusion Transformer (TFT), blend local sequence encoders with an interpretable attention decoder to deliver path-consistent forecasts across horizons (Lim et al. 2020). These ideas connect directly to our objective of stabilizing $\{1h, 4h, 12h, 24h, 48h\}$ returns while preserving horizon-specific nuance.

Inputs: technical, multimodal, and cross-asset context. Beyond OHLCV, technical indicators (moving averages, RSI, MACD, Bollinger Bands, Ichimoku) remain the dominant augmentations and often improve short-horizon accuracy (Jaquart, Dann, and Martin 2020). On-chain variables (e.g., active addresses, hash rate) can add medium-horizon context, particularly for BTC (Ji, Bouri, and Roubaud

2019), while sentiment/news signals help around jump dynamics (Gurgul, Wójcik, and Sliwinski 2025; Buzcu et al. 2021); some studies also incorporate macro and cross-asset drivers to capture regime effects (Mahfooz et al. 2024). Multi-asset conditioning (e.g., using BTC to inform altcoin dynamics) or shared training across coins exploits inter-market co-movements (Tanwar and Srivastava 2021; Peng, Wu, and Zhou 2024). In our setting, we deliberately focus on market-technical inputs to reduce integration risk and enable clean multi-timeframe modeling; we retain widely adopted indicators (including Bollinger Bands) to capture dispersion and regime shifts in a standardized way.

Architectures for crypto regression. Recurrent baselines (LSTM/GRU) remain competitive for financial sequences; temporal convolutions (CNN/TCN) capture local motifs and can stabilize intraday learning. Hybrids (CNN–RNN) frequently outperform either family alone, while Transformer-style models show mixed results on typical crypto datasets. For instance, a tuned CNN–GRU can surpass Transformer variants on both short and extended windows (Mazinani, Ghods, and Amiri 2024). TFT further integrates variable selection and interpretable attention for multi-horizon, multivariate forecasting, and has been highlighted as effective in non-stationary, multi-scale contexts (Lim et al. 2020; Ouyang 2023). These patterns motivate our multi-encoder design across H1/H4/D1 with horizon-specific heads.

Protocols and evaluation. Heterogeneity in exchanges and time spans, feature sets and normalization windows, temporal splits (static vs. rolling/walk-forward), and target definitions limits cross-paper comparability and can invert model rankings (Jaquart, Dann, and Martin 2020). Beyond statistical error, cost-aware backtests that include fees, slippage, and execution constraints are essential for assessing deployability; only a subset of studies evaluate under realistic frictions, which can materially alter conclusions. We therefore emphasize standardized chronological splits and cost-aware evaluation to bridge the gap between error metrics and actionable trading performance.

2.3 Problem Formulation

We consider a cryptocurrency c observed simultaneously across three distinct timeframes: hourly ($H1$), four-hourly ($H4$), and daily ($D1$). At each reference time t, the model has access to three multivariate sequences of features, denoted as:

$$\mathbf{X}_t^{H1} \in \mathbb{R}^{w_{H1} \times d}, \quad \mathbf{X}_t^{H4} \in \mathbb{R}^{w_{H4} \times d}, \quad \mathbf{X}_t^{D1} \in \mathbb{R}^{w_{D1} \times d},$$

where d is the number of features (raw OHLCV prices and technical indicators), and w_{H1}, w_{H4}, w_{D1} are the sequence lengths specific to each timeframe.

The forecasting objective is to predict the future log-normalized returns of the asset at multiple horizons $h \in \{1h, 4h, 12h, 24h, 48h\}$, defined as:

$$Y_{t+h} = \tanh\left(\frac{P_{t+h} - P_t}{P_t}\right),$$

where P_t is the asset close price at time t.

Formally, this is framed as a *multi-horizon regression problem* in which a deep learning model f_θ learns a mapping:

$$f_\theta : \left(\mathbf{X}_t^{H1}, \mathbf{X}_t^{H4}, \mathbf{X}_t^{D1}\right) \mapsto \left(\hat{Y}_{t+1h}, \hat{Y}_{t+4h}, \hat{Y}_{t+12h}, \hat{Y}_{t+24h}, \hat{Y}_{t+48h}\right).$$

The training dataset is constructed from historical market records, producing pairs of multi-timeframe sequences and their corresponding targets:

$$\mathcal{D} = \left\{\left(\mathbf{X}_t^{H1}, \mathbf{X}_t^{H4}, \mathbf{X}_t^{D1}, Y_{t+1h}, Y_{t+4h}, Y_{t+12h}, Y_{t+24h}, Y_{t+48h}\right)\right\}_{t=1}^{N}.$$

The learning objective is to minimize a weighted loss across horizons, allowing the model to capture both short-term fluctuations and longer-term market trends within a unified multi-timeframe architecture.

2.4 Data

2.4.1 Data Forming

The historical dataset consists of Bitcoin market records spanning from September 2017 to September 2022. Three granularities are considered in parallel: hourly ($H1$), four-hourly ($H4$), and daily ($D1$). Each sequence is constructed by sliding a fixed-size window over the historical data, producing synchronized multi-timeframe inputs. Specifically, windows of length $w_{H1} = 100$, $w_{H4} = 25$, and $w_{D1} = 30$ are used, ensuring that each input sample contains recent information from all timeframes.

The prediction targets are defined as future returns at multiple horizons $h \in \{1h, 4h, 12h, 24h, 48h\}$, computed as:

$$Y_{t+h} = \tanh\left(\frac{P_{t+h} - P_t}{P_t}\right),$$

where P_t is the asset close price at reference time t. This formulation guarantees bounded targets, mitigating the effect of extreme price fluctuations while retaining directional information.

To ensure robust evaluation, the dataset is partitioned into five chronological splits, each with non-overlapping training and validation intervals. The last split is held out for testing on unseen data, while the first four are used for model training and validation, following a walk-forward strategy.

Table 2.1: Extended temporal splits for model training and validation

Split ID	Train Start	Train End	Val Start	Val End
1	2017-09-20	2020-03-20	2020-03-21	2020-09-20
2	2018-03-20	2020-09-20	2020-09-21	2021-03-20
3	2018-09-20	2021-03-20	2021-03-21	2021-09-20
4	2019-03-20	2021-09-20	2021-09-21	2022-03-20
5	2019-09-20	2022-03-20	2022-03-21	2022-09-20

Split 5 is reserved exclusively for out-of-sample testing, providing an unbiased estimate of generalization performance. Data from September 2022 to July 2025 is reserved for out-of-sample evaluation, enabling validation under realistic trading strategies.

Forecastable Component Analysis (ForeCA) analysis on the held-out test split (Split 5) :
Forecastability in ForeCA (Leitner and Hornik 2016) is derived from the distribution of spectral power across frequencies. Given a time series, we first estimate its power spectral density (PSD) using Welch's method. The PSD values are normalized into a probability distribution

$$p_k \;=\; \frac{S_k}{\sum_{j=1}^{N} S_j}, \qquad \sum_{k=1}^{N} p_k = 1,$$

where S_k is the spectral density at frequency bin k. The spectral complexity is then measured by the normalized Shannon entropy

$$H_{\text{norm}} \;=\; \frac{-\sum_{k=1}^{N} p_k \log p_k}{\log N},$$

and the *forecastability index* is defined as

$$\Omega \;=\; 1 - H_{\text{norm}}, \qquad \Omega \in [0, 1].$$

Intuitively, a spectrum concentrated on a few frequencies (low entropy) yields a high Ω, indicating strong cyclical structure and predictability. Conversely, a flat spectrum, characteristic of white noise, produces $\Omega \approx 0$, signaling absence of exploitable patterns. In the multivariate setting, ForeCA employs an EM-based procedure on whitened features to extract the linear combination that maximizes Ω; the reported Ω_1 corresponds to the forecastability of this first component.

We quantify forecastability on Split 5 using ForeCA's entropy-based criterion. The resulting first-component scores (Ω_1) per timeframe are:

Table 2.2: Forecastability (Ω_1) on Split 5

Timeframe	**H1**	**H4**	**D1**
Ω_1	0.078	0.101	0.185

On Split 5, D1 ($\Omega_1 = 0.185$) exhibits the strongest low-frequency structure and is the most predictable, followed by H4 (0.101) and H1 (0.078). Practically, this suggests (i) more stable signals at daily granularity, (ii) modest structure at 4h, and (iii) near-noise behaviour intraday.

Interestingly, our forecastability values on Split 5 (H1: 0.078, H4: 0.101, D1: 0.185) fall within the same interval as those reported in the recent *FinTSB* benchmark for equity markets (0.068–0.187 across movement patterns) (Hu et al. 2025). This convergence suggests that the entropy-based measure captures a similar degree of predictability across asset classes. It reinforces the credibility of our estimation and indicates that cryptocurrency series exhibit spectral entropy levels comparable to those of traditional financial time series.

2.4.2 Normalization

Input features are normalized to ensure comparability across dimensions and to stabilize model training. A hybrid scheme is applied depending on the statistical nature of each feature:

- **MinMax normalization**: applied to features bounded by natural scales, such as prices, volumes, or level-based indicators:
  - `open, high, low, close, volume`
  - `RSI, tenkan_sen, kijun_sen, senkou_span_a, senkou_span_b`
  - `bollinger_mavg, bollinger_hband, bollinger_lband`
- **Z-score normalization**: applied to features representing variation, slope, or volatility, which are unbounded and centered around zero:
  - `MACD, MACD_signal, macd_histogram, macd_divergence, macd_slope`
  - `bollinger_width`

Normalization parameters are computed globally on the training portion of each split to avoid information leakage. The same statistics are then applied consistently to both validation and test sets. This design guarantees that the model only has access to information available up to the training period, ensuring a realistic and leakage-free evaluation.

2.5 Model Architecture

The proposed model is a multi-timeframe, multi-horizon forecaster designed to fuse heterogeneous temporal contexts (H1, H4, D1) within a unified architecture. It consists of (i) lightweight sequence encoders per timeframe, (ii) a temporal aggregation block that preserves both extremes and global structure, (iii) a learned fusion gate that adaptively weights timeframes, (iv) a shared representation trunk, and (v) multiple regression heads, one per horizon.

2.5.1 Input Representation

At a reference time t, three synchronized sequences are constructed:

$$\mathbf{X}_t^{H1} \in \mathbb{R}^{w_{H1} \times d}, \quad \mathbf{X}_t^{H4} \in \mathbb{R}^{w_{H4} \times d}, \quad \mathbf{X}_t^{D1} \in \mathbb{R}^{w_{D1} \times d},$$

with $(w_{H1}, w_{H4}, w_{D1}) = (100, 25, 30)$ and feature dimension d (OHLCV and indicators). Targets are the bounded returns $Y_{t+h} = \tanh\left(\frac{P_{t+h} - P_t}{P_t}\right)$ for $h \in \{1h, 4h, 12h, 24h, 48h\}$.

2.5.2 Timeframe Encoders

Each timeframe is encoded by a lightweight block:

$$\mathbf{H}^{(\tau)} = \text{Dropout}\Big(\text{GLU}\Big(\text{LSTM}\big(\mathbf{X}^{(\tau)} W_{\text{in}}^{(\tau)}\big)\Big)\Big), \quad \tau \in \{H1, H4, D1\}.$$

The block projects inputs to a hidden size (256 for H1; 128 for H4/D1), applies a stacked LSTM with orthogonal initialization, then a gated linear unit (GLU) to enhance non-linear feature selection while remaining parameter-efficient.

2.5.3 Temporal Aggregation

Given the sequence of hidden states $\mathbf{H}^{(\tau)} = [\mathbf{h}_1, \ldots, \mathbf{h}_T]$ produced by the LSTM encoder, each $\mathbf{h}_t$ already summarizes the past inputs up to time t due to the causal recurrence. In particular, $\mathbf{h}_T$ can be seen as a global embedding of the entire sequence, although it may compress information and potentially discard useful details.

To enrich this representation, four complementary summary statistics are extracted:

$$\mathbf{s}_{\text{last}} = \mathbf{h}_T, \quad \mathbf{s}_{\text{mean}} = \tfrac{1}{T}\sum_t \mathbf{h}_t, \quad \mathbf{s}_{\text{max}} = \max_t \mathbf{h}_t, \quad \mathbf{s}_{\text{attn}} = \sum_t \alpha_t \, \mathbf{h}_t,$$

with additive attention weights $\alpha_t = \text{softmax}\big(\mathbf{v}^\top \tanh(W\mathbf{h}_t)\big)$.

The concatenation $\big[\mathbf{s}_{\text{last}} \| \mathbf{s}_{\text{mean}} \| \mathbf{s}_{\text{max}} \| \mathbf{s}_{\text{attn}}\big]$ of dimension $4H$ is finally projected to a 256-dimensional embedding $\mathbf{e}_\tau \in \mathbb{R}^{256}$ for each timeframe τ. This design ensures that both local recency, global trends, extreme values, and salient temporal patterns are captured jointly.

2.5.4 Learned Fusion Gate

To fuse the three embeddings $(\mathbf{e}_{H1}, \mathbf{e}_{H4}, \mathbf{e}_{D1})$, we learn simplex weights via a small MLP:

$$\mathbf{w} = \text{softmax}\Big(\phi\big([\mathbf{e}_{H1} \| \mathbf{e}_{H4} \| \mathbf{e}_{D1}]\big)\Big) \in \Delta^2, \qquad \mathbf{z}_{\text{fused}} = \sum_\tau w_\tau \, \mathbf{e}_\tau \in \mathbb{R}^{256}.$$

This gate is more stable than multi-head attention over only three tokens and empirically reduces overfitting.

2.5.5 Shared Trunk and Multi-Horizon Heads

A shared MLP processes $\mathbf{z}_{\text{fused}}$ into a compact representation $\mathbf{z} \in \mathbb{R}^{128}$, followed by $K = 5$ horizon-specific heads:

$$\hat{Y}_{t+h_k} = g_k(\mathbf{z}), \quad k = 1, \ldots, 5, \quad h_k \in \{1h, 4h, 12h, 24h, 48h\}.$$

Each head is a shallow MLP (`Linear`–ReLU–Dropout–`Linear`) enabling specialization per horizon while sharing most capacity.

2.5.6 Learning Objective

The model produces predictions for multiple horizons $h \in \{1h, 4h, 12h, 24h, 48h\}$ simultaneously. For each horizon h, the prediction $\hat{Y}_{t+h}$ is compared to the true return Y_{t+h} using both mean squared error (MSE) and mean absolute error (MAE). To combine horizon-specific losses into a single objective, we consider two strategies:

- **Fixed weighting**: each horizon contributes with a predefined scalar weight α_h, yielding:

$$\mathcal{L}_{\text{fixed}} = \sum_h \alpha_h \, \text{MSE}(\hat{Y}_{t+h}, Y_{t+h}).$$

- **Uncertainty-based weighting (Kendall, Gal, and Cipolla 2018)** : horizon-specific variances σ_h^2 are learned jointly with the model, such that:

$$\mathcal{L}_{\text{Kendall}} = \sum_h \frac{1}{2\sigma_h^2} \|\hat{Y}_{t+h} - Y_{t+h}\|^2 + \frac{1}{2} \log \sigma_h^2.$$

In practice, the network maintains *learnable log-variances* $\log \sigma_h^2$ per horizon. These parameters are not prediction targets, but are used internally within the loss to adaptively balance tasks with different noise levels.

This multi-horizon learning objective encourages the network to capture both short-term fluctuations and long-term dependencies. The Kendall weighting introduces an adaptive mechanism that reflects the noise associated with each horizon, reducing the need for manual tuning and often improving stability across heterogeneous forecast horizons.

Model Architecture:

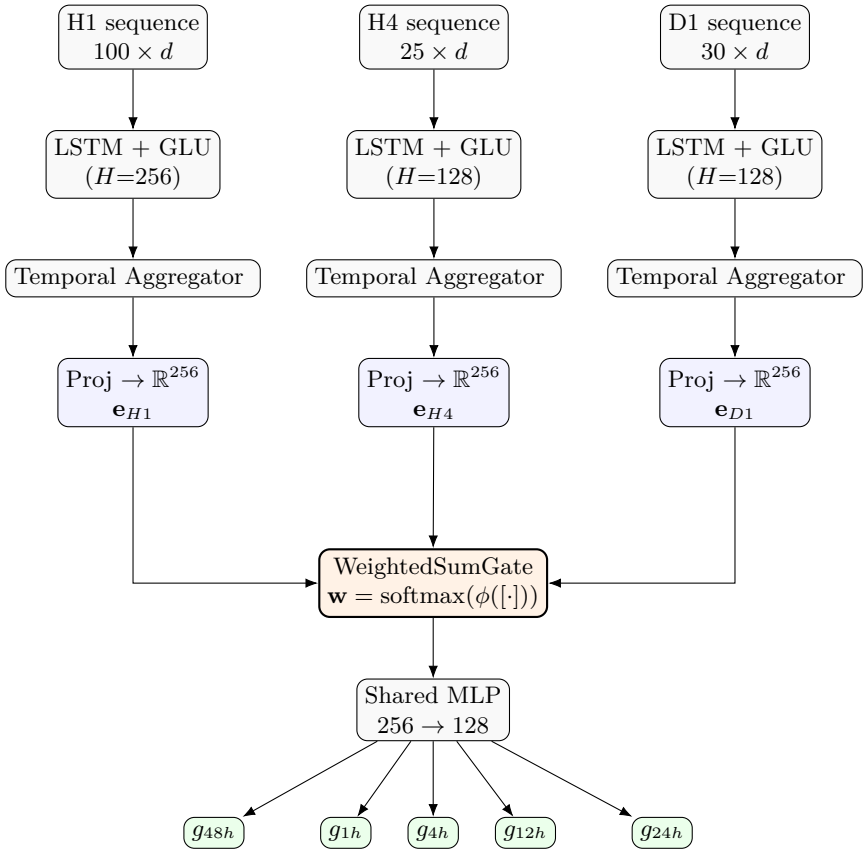


Figure 2.1: Overview of the architecture

2.5.7 Design Rationale

- **Separate inputs per timeframe**: each horizon (**H1**, **H4**, **D1**) is processed by a dedicated encoder, faithfully reflecting the *top-down* trading perspective. Each timeframe carries its own informational content (intraday micro-structure, mid-range dynamics, long-term cycles), which must be preserved before fusion.
- **Asymmetric encoders**: the network capacity is adjusted to the signal-to-noise ratio of each horizon (e.g., H1 being richer in short-term variation, thus assigned a larger hidden dimension H).
- **Four-way temporal aggregation**: multiple summaries are extracted from each sequence—recency (`last`), average level (`mean`), extremes (`max`), and global saliency (`attn`). This controlled redundancy ensures that both local and global temporal patterns are retained.
- **Gated fusion**: instead of over-parameterizing attention over only three tokens, a gated fusion mechanism provides context-dependent weighting while maintaining parsimony.
- **Shared trunk with horizon-specific heads**: after fusion, a common backbone extracts generic representations, while horizon-specific heads allow specialization of forecasts. This structure maximizes feature reuse while minimizing interference across horizons.

2.6 Training

The training procedure is organized around a temporal cross-validation scheme, where the historical dataset is split into multiple contiguous folds. For each fold, the model is trained on a training interval and validated on a subsequent unseen validation interval. This walk-forward strategy ensures that the evaluation is performed in a chronologically consistent way, avoiding any form of look-ahead bias.

Mini-batches are constructed from multi-timeframe sequences $(\mathbf{X}_t^{H1}, \mathbf{X}_t^{H4}, \mathbf{X}_t^{D1})$, with fixed lengths of $w_{H1} = 100$, $w_{H4} = 25$, and $w_{D1} = 30$. Each batch is processed on GPU with a size of 32, and optimization is performed for up to 200 epochs with early stopping based on validation performance. The AdamW optimizer is employed, together with a learning rate scheduler combining linear warm-up and cosine annealing, which stabilizes convergence and reduces the risk of overfitting. The best-performing checkpoint is retained for each fold based on the validation mean squared error (MSE).

2.6.1 Learning Objective

The model produces predictions for multiple horizons $h \in \{1h, 4h, 12h, 24h, 48h\}$ simultaneously. For each horizon h, the prediction $\hat{Y}_{t+h}$ is compared to the true return Y_{t+h} using both mean squared error (MSE) and mean absolute error (MAE). To combine horizon-specific losses into a single objective, two strategies are considered:

- **Fixed weighting**: each horizon contributes with a predefined scalar weight α_h, yielding:

$$\mathcal{L}_{\text{fixed}} = \sum_h \alpha_h \, \text{MSE}(\hat{Y}_{t+h}, Y_{t+h}).$$

- **Uncertainty-based weighting (Kendall et al.)**: horizon-specific variances σ_h^2 are learned jointly with the model, such that:

$$\mathcal{L}_{\text{Kendall}} = \sum_h \frac{1}{2\sigma_h^2} \|\hat{Y}_{t+h} - Y_{t+h}\|^2 + \frac{1}{2} \log \sigma_h^2.$$

  This formulation allows the model to dynamically balance horizons, placing more emphasis on targets with lower inherent uncertainty.

This multi-horizon learning objective encourages the network to simultaneously capture both short-term fluctuations and long-term dependencies, while the Kendall weighting introduces an adaptive mechanism that reflects the noise level associated with each prediction horizon.

2.6.2 Regularization and Generalization

Several mechanisms are employed to improve generalization and mitigate overfitting:

- **Dropout**: applied within encoders, fusion layers, and heads to prevent co-adaptation of neurons.
- **Weight decay**: via the AdamW optimizer, encouraging smaller parameter magnitudes.
- **Early stopping**: halts training if validation loss does not improve after a patience window.
- **Learning rate scheduling**: linear warm-up stabilizes the initial phase, while cosine annealing improves convergence.

2.6.3 Evaluation Metrics

Model performance is assessed with both regression and directional metrics:

- **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - y_i\right)^2$$

- **Root Mean Squared Error (RMSE)**:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$$

- **Coefficient of determination (R^2)**:

$$R^2 = 1 - \frac{\sum_i(\hat{y}_i - y_i)^2}{\sum_i(y_i - \bar{y})^2}$$

- **Directional Accuracy (DA)**:

$$\text{DA} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}_{\{\text{sign}(\hat{y}_i)=\text{sign}(y_i)\}}$$

- **Hit Ratio**: identical to DA but computed per horizon and averaged across horizons.

Regression metrics evaluate numerical precision, while directional metrics reflect practical trading relevance by quantifying whether the model correctly predicts the sign of future returns.

The Mean Absolute Percentage Error (MAPE) is deliberately excluded, as financial returns y_i can be arbitrarily close to zero. In such cases, the denominator $|y_i|$ leads to inflated or undefined values, making MAPE unreliable and misleading for return-based forecasting tasks.

2.7 Results

All training and evaluation procedures were conducted in the same computational environment as in Part I, namely on Google Colab equipped with GPU acceleration.

Evaluation Setup and Protocol We train four models on distinct temporal folds (Model 1 with split 1, Model 2 with split 2,...) and evaluate exclusively on the unseen out-of-sample fold (Split 5). Beyond the four stand-alone models, we also form three ensembles from the per-horizon predictions $\hat{\mathbf{Y}}_t^{(m)}$ of the four models $m \in \{1, \ldots, 4\}$:

$$\textbf{Ensemble mean:} \quad \hat{\mathbf{Y}}_t^{\text{mean}} = \frac{1}{4}\sum_{m=1}^{4}\hat{\mathbf{Y}}_t^{(m)},$$

$$\textbf{Ensemble median:} \quad \hat{\mathbf{Y}}_t^{\text{med}} = \text{median}\big(\hat{\mathbf{Y}}_t^{(1)}, \ldots, \hat{\mathbf{Y}}_t^{(4)}\big) \quad \text{(component-wise)},$$

$$\textbf{Validation-weighted mean:} \quad \hat{\mathbf{Y}}_t^{\text{vw}} = \sum_{m=1}^{4} w_m\,\hat{\mathbf{Y}}_t^{(m)}, \qquad w_m \propto \frac{1}{\text{MSE}_m^{\text{val}}}, \ \sum_m w_m = 1.$$

Horizon weighting during training :

Let MSE_h denote the per-horizon loss. We consider two aggregation schemes: (i) *fixed* weights α_h with $\mathcal{L} = \sum_h \alpha_h \text{MSE}_h$; (ii) *uncertainty (Kendall) weights* with $\mathcal{L} = \sum_h \frac{1}{2\sigma_h^2}\text{MSE}_h + \frac{1}{2}\log\sigma_h^2$, where $\log\sigma_h^2$ are learned jointly with the model. In our main experiments, the Kendall scheme is enabled (dynamic horizon weighting). At inference, no additional reweighting across horizons is applied; unless stated otherwise, global metrics average uniformly across horizons (simple pooling over all elements).

Global Performance Table 2.3 reports aggregate results on Split 5. Best values per column are in **bold**.

Table 2.3: Global metrics on the held-out test fold (Split 5). Best per column in **bold**.

Model	MSE	MAE	RMSE	R^2	DA
split_1	0.0005	0.0135	0.0216	0.4984	0.6684
split_2	0.0004	0.0132	0.0212	0.5194	0.6434
split_3	0.0005	0.0139	0.0220	0.4818	0.6539
split_4	0.0004	0.0129	**0.0210**	0.5278	0.6868
ensemble_mean	0.0004	**0.0127**	0.0207	0.5404	0.6868
ensemble_median	0.0004	0.0128	0.0208	0.5369	0.6763
ensemble_val_weighted	**0.0004**	**0.0127**	**0.0207**	**0.5411**	**0.6882**

Horizon-wise Behaviour and Calibration Figure 2.3 overlays the distributions of true vs. predicted returns (ensemble mean) for 12h and 24h. Short horizons are more under-dispersed, while 24h aligns better with the empirical distribution.

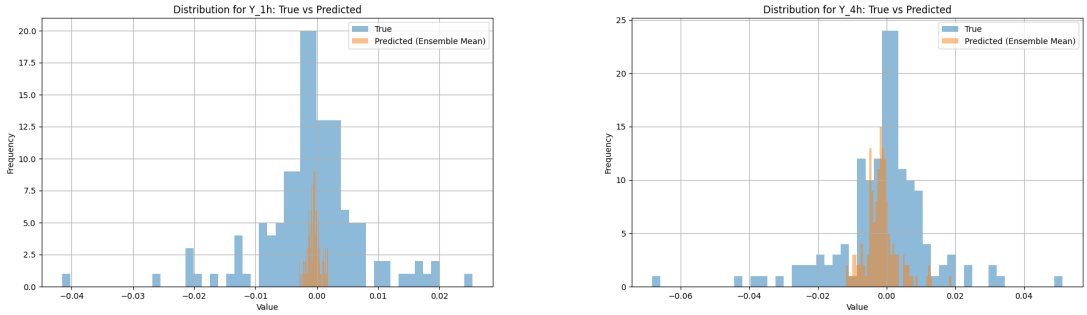


Figure 2.2: Distributions of true vs. predicted returns (ensemble mean) for 1h (left) and 4h (right).

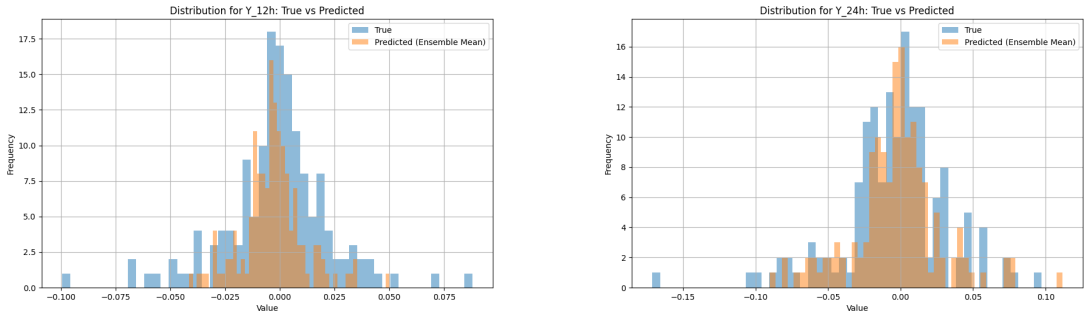


Figure 2.3: Distributions of true vs. predicted returns (ensemble mean) for 12h (left) and 24h (right). Short horizons exhibit stronger shrinkage towards zero; 24h is better calibrated.

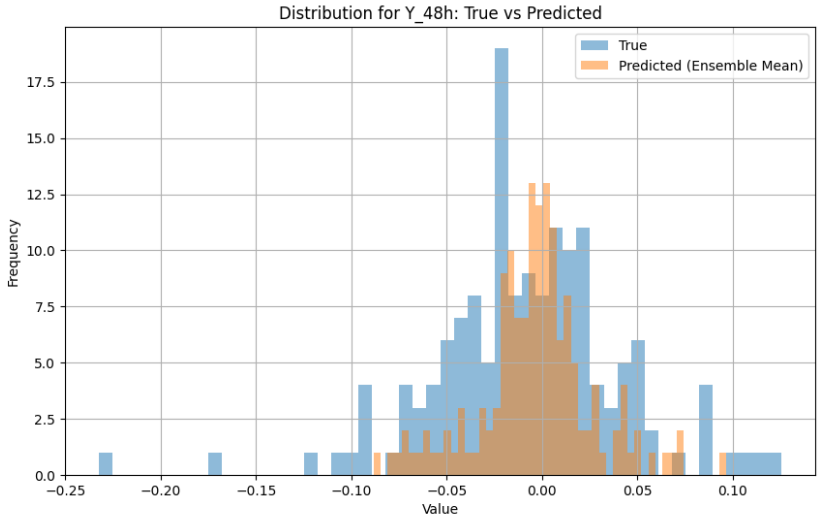


Figure 2.4: Distributions of true vs. predicted returns 48H

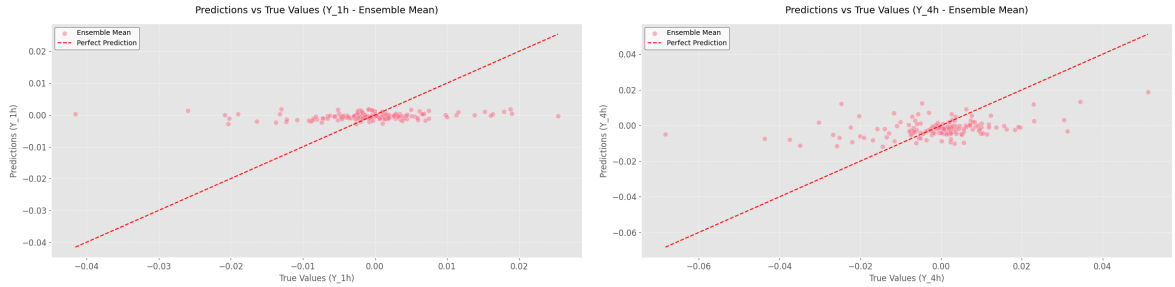


Figure 2.5: Predicted vs. true returns (ensemble mean). Left: 1h ; Right: 4h

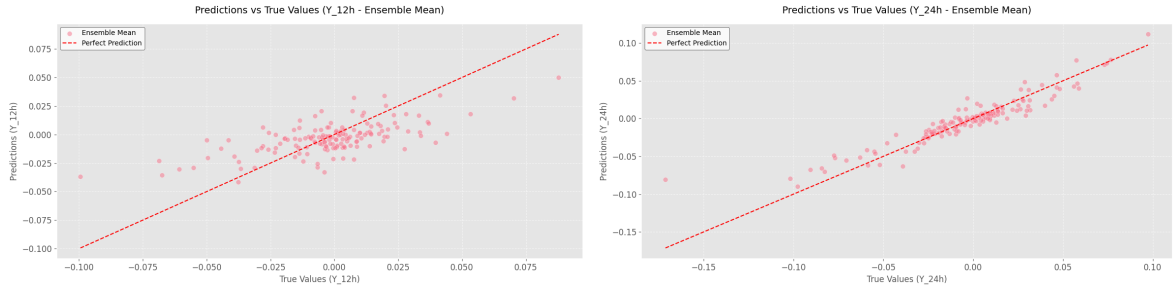


Figure 2.6: Predicted vs. true returns (ensemble mean). Left: 12h ; Right: 24h aligns closely with the diagonal.

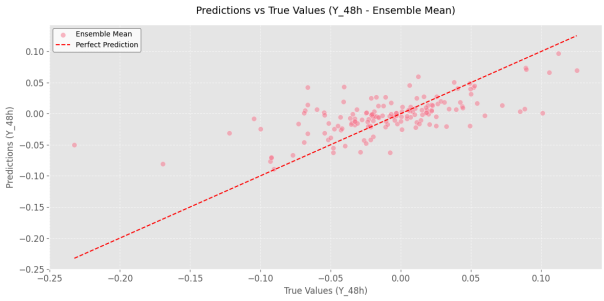


Figure 2.7: Predicted vs. True returns 48H

Table 2.4: Comparative R^2 and Directional Accuracy (DA) across horizons. Ensemble results consistently outperform individual splits.

	1h	4h	12h	24h	48h
R^2 (splits avg.)	0.03	0.13	0.40	0.81	0.40
R^2 (ensemble)	0.04	0.15	0.44	0.87	0.42
DA (splits avg.)	0.55	0.50	0.67	0.85	0.74
DA (ensemble)	0.57	0.51	0.71	0.89	0.75

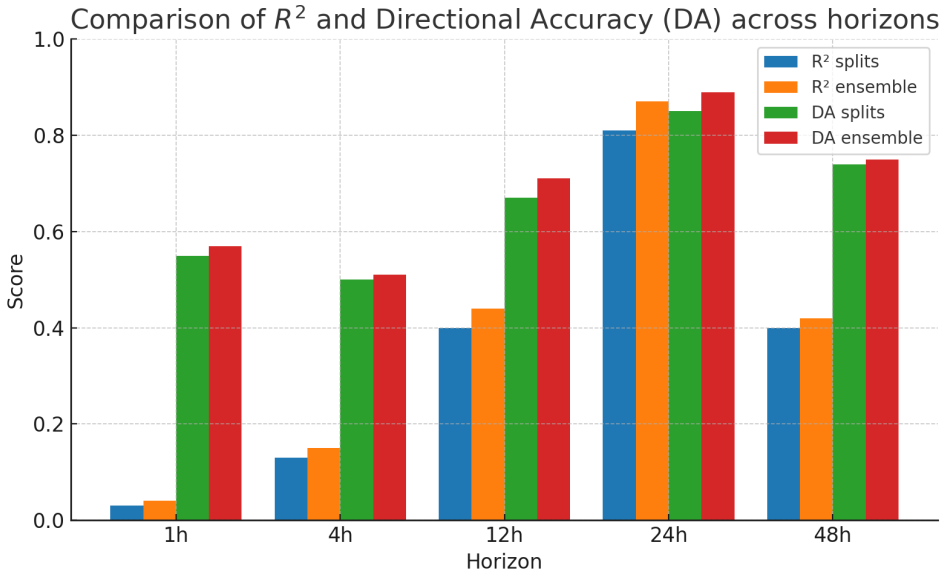


Figure 2.8: Comparison of R^2 and Directional Accuracy (DA) by horizon

Performance improves markedly from 1h/4h to 12h/24h; the 24h horizon yields the strongest overall results.

2.8 Discussion

Overall findings. The validation-weighted ensemble achieves the best trade-off (lowest RMSE 0.0207, highest $R^2 = 0.5411$, best DA 0.6882), slightly surpassing the best single model (split_4, RMSE 0.0210). Averaging models trained on different temporal regimes stabilizes errors and improves explained variance. DA $\approx$ 0.67–0.69 indicates a consistent directional edge over chance on the unseen split.

Short vs. long horizons. Short horizons (1h, 4h) are intrinsically noisy (lower signal-to-noise ratio). Under a multi-horizon MSE objective, this fosters conservative predictions near zero, observed as under-dispersion in histograms and a shallow slope in 1h scatter plots. Longer horizons (12h–24h) aggregate more information, leading to better calibration and higher sensitivity to larger moves. The comparative performance of R^2 and Directional Accuracy (DA) across horizons is summarized in Table 2.4 and Figure 2.8.

Interpreting error magnitudes under the tanh **transform.** Targets are $Y = \tanh\left(\frac{P_{t+h}-P_t}{P_t}\right)$, a monotonic, compressive mapping. In tanh-space, extreme raw returns are squashed toward ± 1, so reported MSE/RMSE constitute a conservative view of error; conversely, mapping back via atanh inflates errors approximately by $(1 - Y^2)^{-1}$. In our data, Y mostly lies in $[-0.1, 0.1]$, so the inflation is mild on average; tail events would amplify it. Directional metrics are unaffected (the sign is preserved).

Each split-specific model captures patterns tied to its training regime; ensembling reduces variance and idiosyncratic biases. The validation-weighted mean further emphasizes the most reliable models (lowest validation MSE), explaining the systematic gains in RMSE, R^2, and DA.

Practical implications :

For deployment purposes, the *validation-weighted ensemble* emerges as the most reliable option. In practice, model sensitivity at very short horizons (1h, 4h) remains limited, whereas mid-range horizons (12h–24h) provide more stable and actionable signals. Furthermore, since target variables are expressed in tanh-space, reported error metrics should be interpreted as conservative relative to the raw-return scale. To avoid masking difficulties specific to short horizons, both global and horizon-specific metrics must be systematically reported.

2.8.1 Effect of the tanh Target Transformation

In this study, forecast targets are defined as

$$Y \;=\; \tanh\Big(\frac{P_{t+h}-P_t}{P_t}\Big),$$

where P_t and P_{t+h} denote the asset price at the prediction origin and forecast horizon, respectively. This monotonic and compressive mapping has two direct implications:

1. **Error deflation in the transformed domain.** Because the tanh function compresses extreme raw returns toward ± 1, squared errors in the transformed space are naturally reduced compared to the raw-return domain. Consequently, MSE and RMSE values reported in tanh-space represent conservative estimates.

2. **Error inflation when unwarping.** To recover raw returns, one applies the inverse mapping $r = \operatorname{atanh}(Y)$, with local derivative

$$\frac{dr}{dY} \;=\; \frac{1}{1-Y^2}.$$

Hence, a prediction error $\varepsilon_{\tanh}$ in the transformed domain translates into a raw-return error of approximately

$$\varepsilon_{\text{raw}} \;\approx\; \frac{\varepsilon_{\tanh}}{1-Y^2},$$

and the expected squared error becomes

$$\text{MSE}_{\text{raw}} \;\approx\; \mathbb{E}\left[\frac{\varepsilon_{\tanh}^2}{(1-Y^2)^2}\right].$$

Since $\frac{1}{1-Y^2} > 1$ for all $Y \neq 0$, errors are systematically inflated when projected back to the raw-return scale, with stronger effects when predictions approach the saturation regions ($|Y| \to 1$). In our dataset, most values lie within $[-0.1, 0.1]$, implying only mild inflation (a few percent), although rare tail events could substantially magnify this effect.

2.9 Trading Evaluation

2.9.1 Objectives & Out-of-Sample Setup

We evaluate a trading strategy derived from the model's multi-horizon signals on periods strictly *unseen* by training 2.9 : learning stops on **2022-09-20**, and all trading windows are posterior. We consider BTCUSDT (Binance) with hourly execution, and three market regimes diagnosed with ForeCA :

- **Downtrend**: 2025-01-20 $\to$ 2025-04-09, $\Omega_1 = 0.077$,
- **Uptrend**: 2025-04-09 $\to$ 2025-08-08, $\Omega_1 = 0.1285$,
- **High-volatility**: 2024-03-12 $\to$ 2024-11-06, $\Omega_1 = 0.1009$.

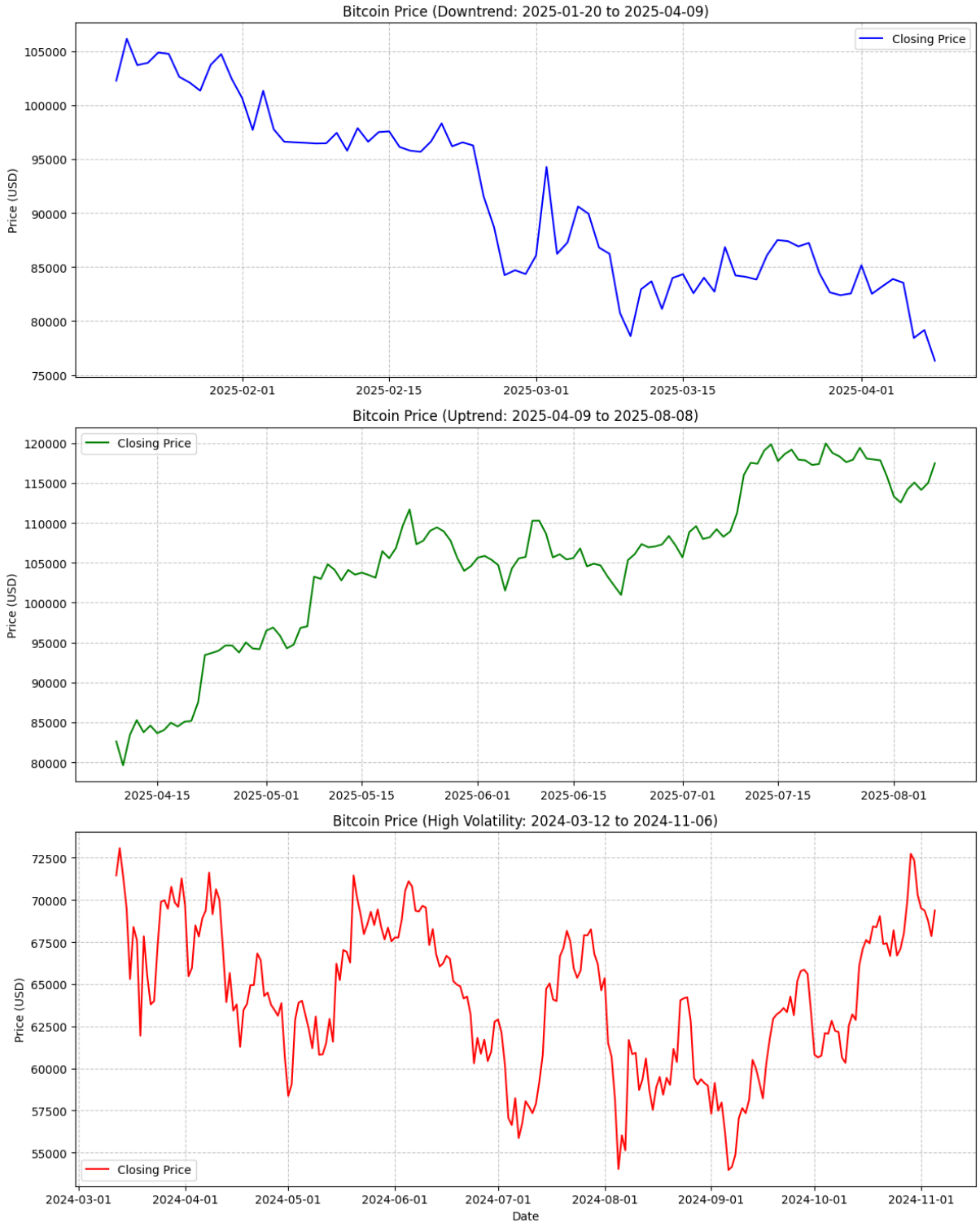


Figure 2.9: Bitcoin Price Trends: Downtrend (Jan-Apr 2025), Uptrend (Apr-Aug 2025), High Volatility (Mar-Nov 2024)

2.9.2 From Model Outputs to Trade Signals

Daily anchors and normalization. Once per day (every 24h), we form an *anchor* timestamp t and extract synchronized sequences $(\mathbf{X}^{H1}, \mathbf{X}^{H4}, \mathbf{X}^{D1})$ with lengths $(100, 25, 30)$. Features are standardized using a *rolling robust* scheme (median/IQR) over a past-only window of 180 days per timeframe, ensuring no look-ahead.

Ensemble over split-specific models. We load the four models trained on Splits 1–4 and aggregate their per-horizon predictions $\hat{\mathbf{Y}}_t^{(m)} \in \mathbb{R}^5$ (for horizons $1h, 4h, 12h, 24h, 48h$) using either:

$$\hat{\mathbf{Y}}_t^{\text{mean}} = \tfrac{1}{4}\sum_{m=1}^{4} \hat{\mathbf{Y}}_t^{(m)}, \qquad \hat{\mathbf{Y}}_t^{\text{median}} = \text{median}_m\left(\hat{\mathbf{Y}}_t^{(m)}\right), \qquad \hat{\mathbf{Y}}_t^{\text{vw}} = \sum_{m=1}^{4} w_m \hat{\mathbf{Y}}_t^{(m)}, \;\; w_m \propto 1/\text{MSE}_m^{\text{val}}.$$

Unless stated otherwise, the default aggregator is the simple mean.

Consensus gating across horizons. Let $\hat{\mathbf{Y}}_t = (\hat{y}_{1h}, \hat{y}_{4h}, \hat{y}_{12h}, \hat{y}_{24h}, \hat{y}_{48h})$ denote the (ensembled) predictions at anchor t (each in tanh-space). We define horizon weights via a softmax over absolute scores with temperature τ:

$$w_h \;=\; \frac{\exp(|\hat{y}_h|/\tau)}{\sum_k \exp(|\hat{y}_k|/\tau)}, \qquad s \;=\; \sum_h w_h\, \hat{y}_h\,.$$

The scalar s is the *consensus score* used to decide trade direction; the vector $\mathbf{w}$ is also used later for PnL attribution by horizon. We use $\tau = \texttt{0.10}$ unless specified.

Decision thresholds and confirmation. We convert s to a position using symmetric thresholds calibrated from the empirical distribution of $|s|$ within the test window:

$$\text{long if } s > \text{thr}, \quad \text{short if } s < -\text{thr}, \quad \text{flat otherwise}, \quad \text{thr} \in \{\text{P70}, \text{P80}, \text{P90}\}.$$

A *confirmation* rule ensures cross-horizon agreement: the sign of the *TP-horizon* prediction (default: 24h) must match at least a fraction $\rho = 0.55$ of the remaining horizons; otherwise the signal is set to flat.

2.9.3 Execution and Risk Management

Entry, fees, and holding. Signals are executed at the *next hour* following the anchor. Fees are set to `6` bps per side (entry and exit). Positions are unit-sized (no leverage in base results), with a maximum holding time of 48 hours.

TP/SL and trailing. We set a take-profit level using the TP-horizon prediction scaled to price:

$$\text{TP} \;=\; P_{\text{entry}} \cdot \big(1 + \text{dir} \cdot \hat{y}_{\text{TP}} \cdot \texttt{tp\_scaling}\big), \;\; \texttt{tp\_scaling} = \texttt{0.7},$$

where $\text{dir} \in \{+1, -1\}$. A fixed stop-loss of `2%` is applied, with optional trailing at `1%`. Positions can also be closed early on a *strong opposing signal* when $|s| > \text{thr} \times$ `opposing_mult` with `opposing_mult=1.5`.

Practical notes. Execution is discrete on H1 closes; slippage is not modeled (stress tests discussed in sensitivity).

2.9.4 Backtest Protocol

- **No leakage:** rolling robust normalization uses past-only windows; anchors and execution times are strictly posterior to training end-date.
- **Daily anchors:** one anchor per day; horizon-specific predictions are mapped to a single s via consensus gating.
- **Grid of settings:** thresholds at P70/P80/P90; aggregator {mean, median, validation-weighted}; τ scans (optional); TP-horizon = 24h unless stated; `tp_scaling`, SL, trailing as above.

2.9.5 Metrics

We evaluate models through a set of complementary metrics capturing both statistical and trading performance. Let r_t denote the strategy return at time t, $\bar{r}$ the mean return, σ_r the standard deviation of returns, r_f the risk-free rate (set to 0 here), and N the number of observations.

- **Sharpe ratio (annualized).** The Sharpe ratio measures risk-adjusted performance by comparing average return to total volatility:

$$\text{Sharpe} \;=\; \frac{\bar{r} - r_f}{\sigma_r} \times \sqrt{H},$$

where $H = 8760$ is the annualization factor for 1-hour bars. Higher values indicate better risk-adjusted performance.

- **Sortino ratio (annualized).** The Sortino ratio penalizes only downside volatility by replacing σ_r with the downside deviation σ_-:

$$\text{Sortino} \;=\; \frac{\bar{r} - r_f}{\sigma_-} \times \sqrt{H}, \quad \sigma_- = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \min(r_t - r_f, 0)^2}.$$

- **Compound Annual Growth Rate (CAGR).** The CAGR measures geometric average growth:

$$\text{CAGR} \;=\; \left(\frac{V_T}{V_0}\right)^{\frac{1}{Y}} - 1,$$

where V_T and V_0 are the final and initial portfolio values, and Y is the horizon in years.

- **Maximum Drawdown (MaxDD).** Maximum drawdown quantifies the worst peak-to-trough decline in portfolio value.

- **Calmar ratio.** The Calmar ratio links long-term growth to drawdown risk:

$$\text{Calmar} \;=\; \frac{\text{CAGR}}{|\text{MaxDD}|}.$$

- **Profit Factor (PF).** The ratio of gross profits to gross losses: Thus, the G_i winning trades on L_j the losing trades.

- **Hit Ratio.** The fraction of winning trades:

$$\text{Hit Ratio} \;=\; \frac{N^+}{N^+ + N^-},$$

where N^+ and N^- are the number of profitable and unprofitable trades.

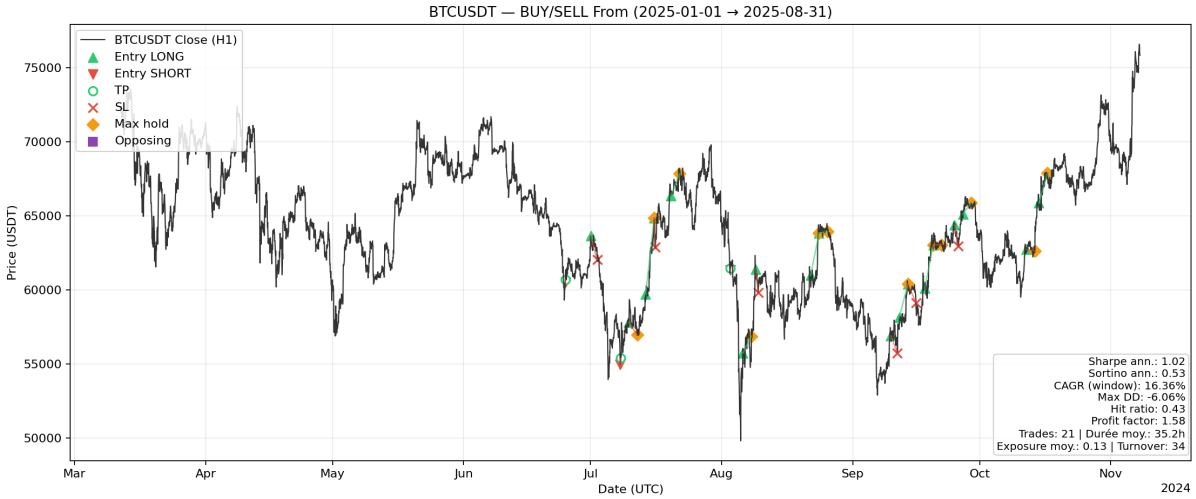
- **Turnover.** The number of executed trades relative to the time horizon, often expressed as annualized trading frequency.

- **Average Exposure.** The proportion of time the portfolio is active in the market.

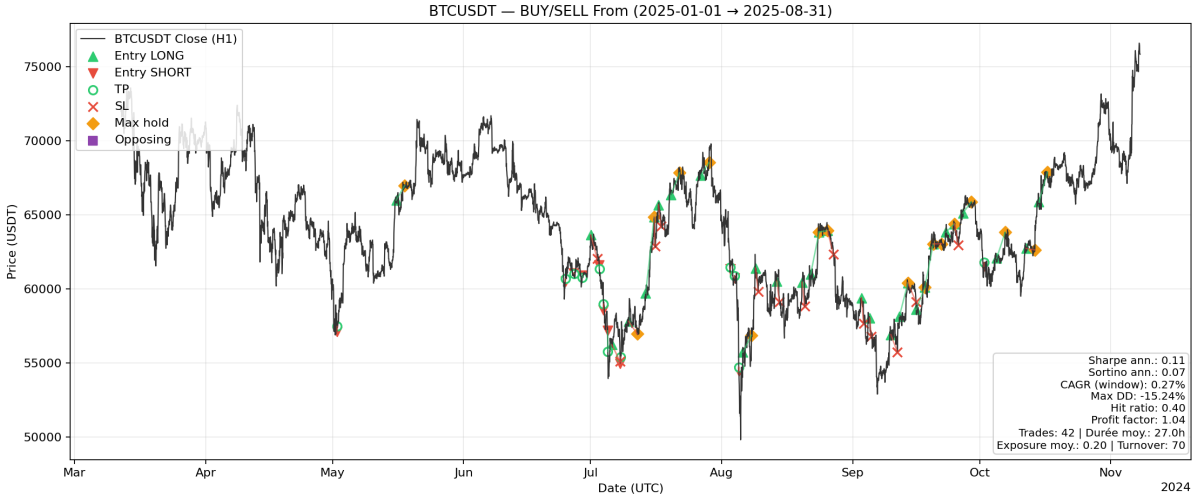
- **Average Trade Duration.** The mean holding period of trades, in hours, capturing persistence of signals.

2.9.6 Results

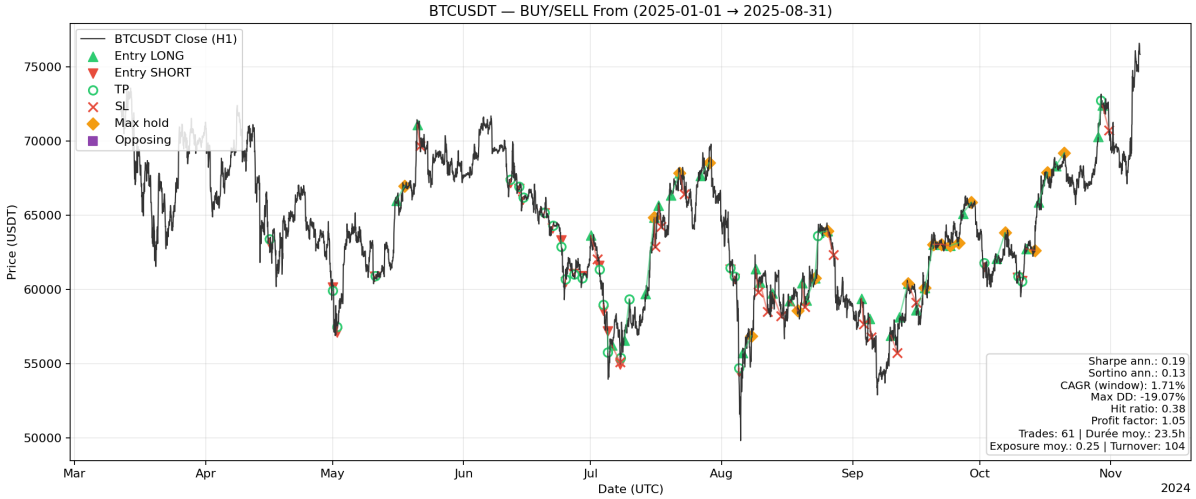
We summarize performance by regime and decision threshold. Below are the graphical charts illustrating the long and short positions taken across the three distinct market regimes: volatility, downtrend, and uptrend. For each regime, the results are reported under different decision thresholds (P90, P80, and P70). The model demonstrates the ability to identify promising entry points; however, the trading strategy applied here is deliberately very strict. A more flexible approach—adjusting position sizing and risk management—could further enhance trade frequency and overall profitability.



(a) Volatility at $P90$

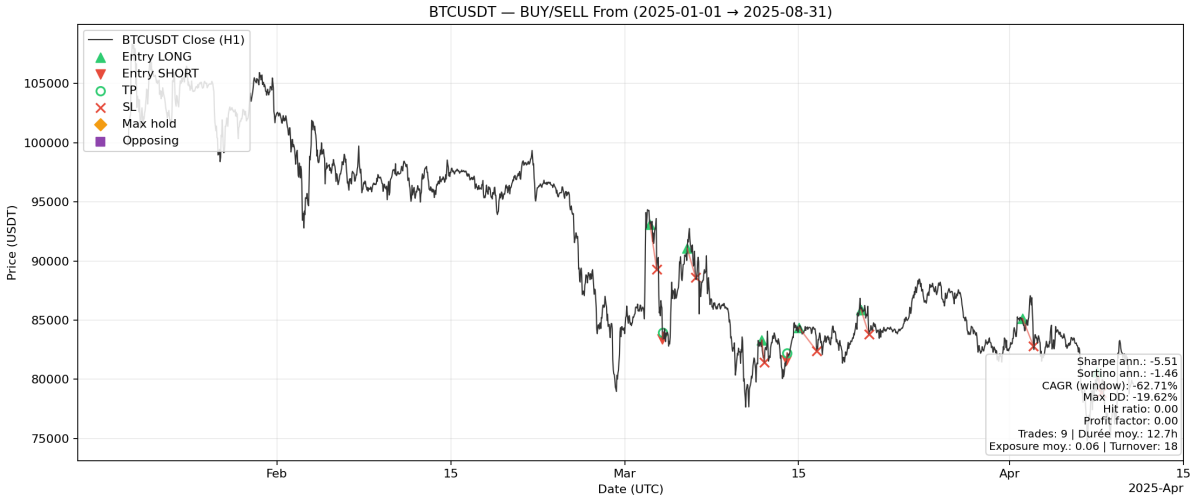


(b) Volatility at $P80$

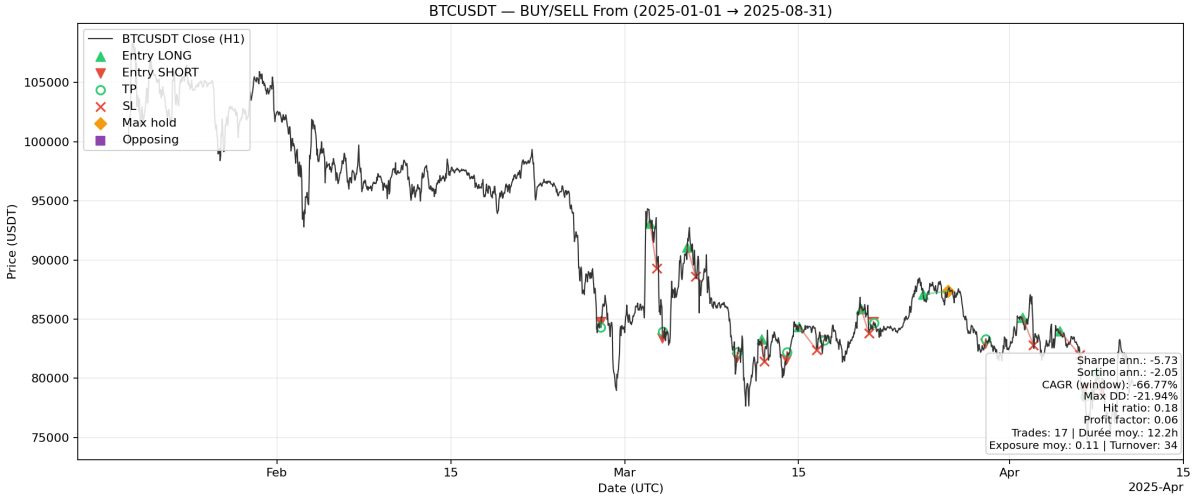


(c) Volatility at $P70$

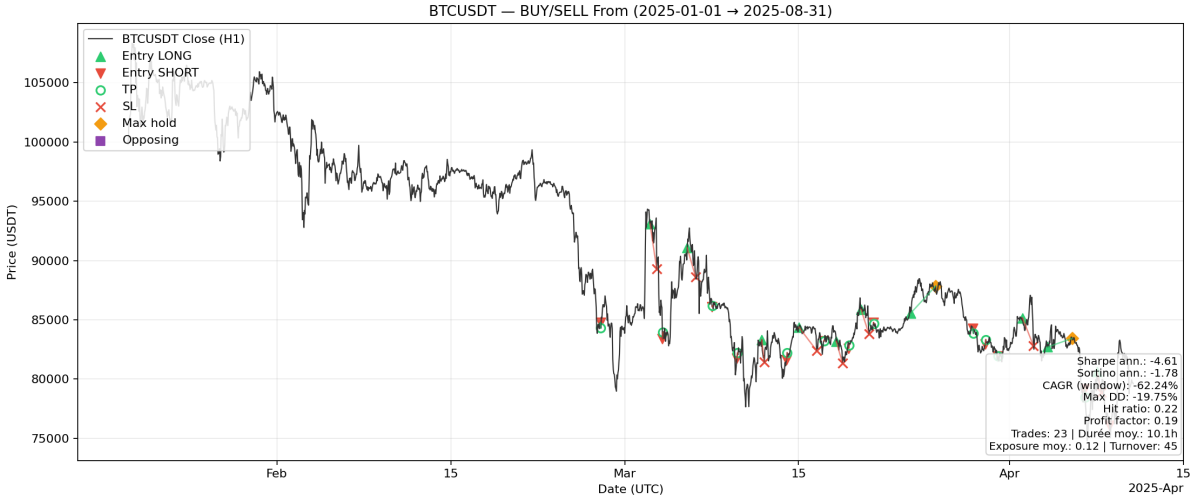
Figure 2.10: Trades executed during high-volatility regimes at different thresholds ($P90$, $P80$, $P70$).



(a) Downtrend at $P90$

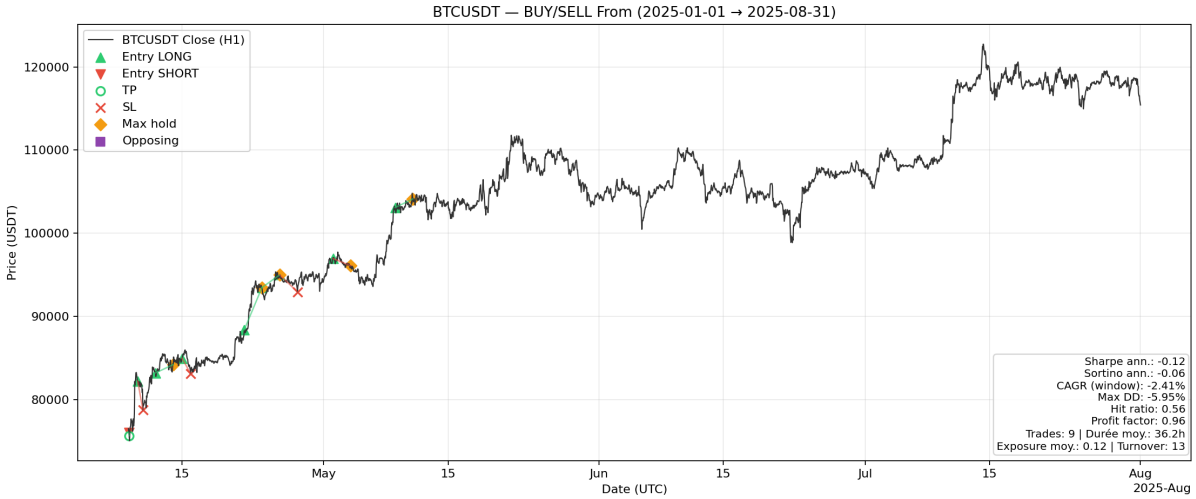


(b) Downtrend at $P80$

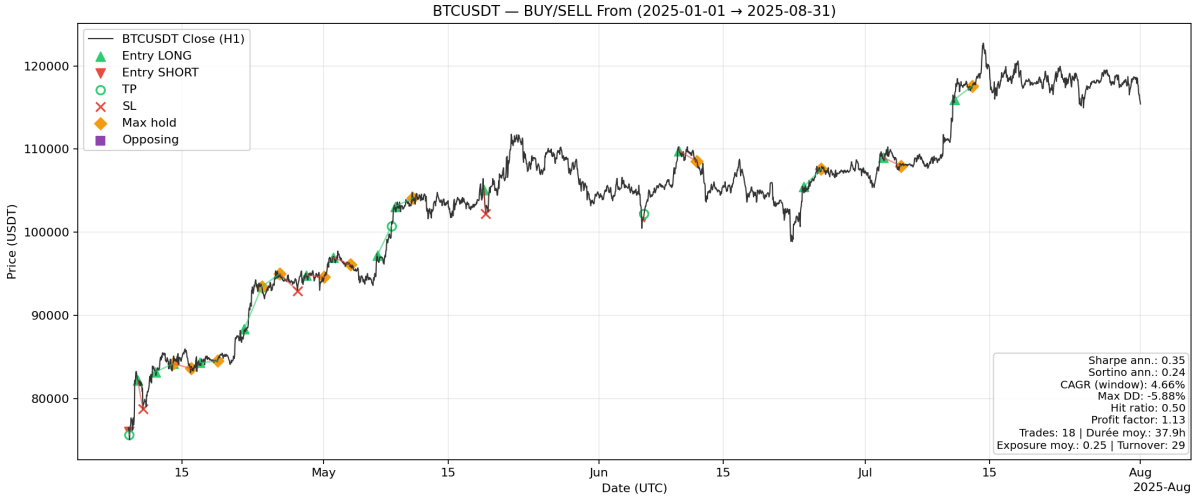


(c) Downtrend at $P70$

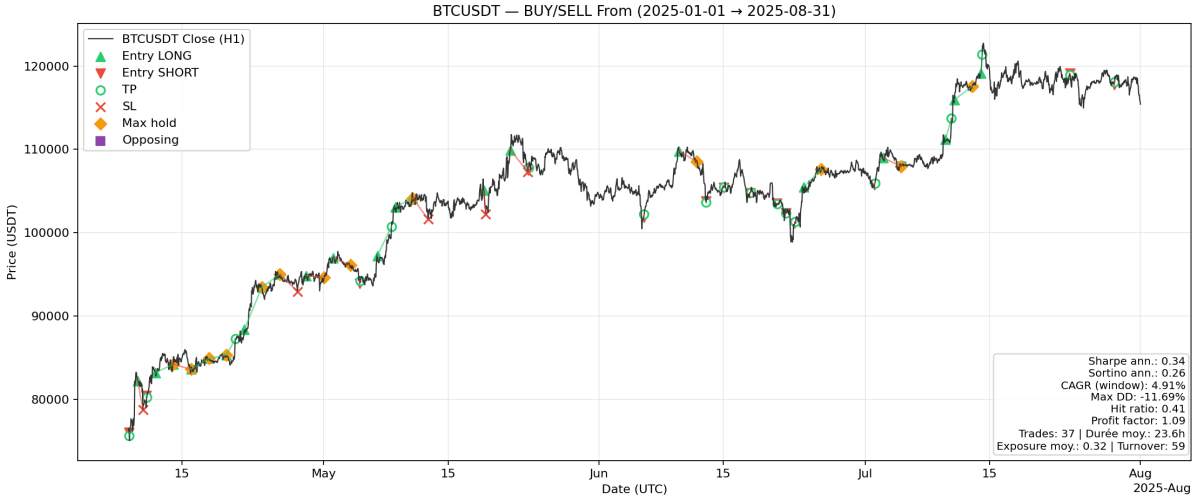
Figure 2.11: Trades executed during strong downtrend regimes at different thresholds ($P90$, $P80$, $P70$).



(a) Uptrend at $P90$



(b) Uptrend at $P80$



(c) Uptrend at $P70$

Figure 2.12: Trades executed during strong uptrend regimes at different thresholds ($P90$, $P80$, $P70$).

Performance by regime and threshold :

Table 2.5: Performance by market regime and decision threshold (default ensemble = mean).

Regime	Thr	Sharpe	Sortino	CAGR	Calmar	MaxDD	PF	Hit	#Trades	AvgExp
Downtrend	P70	-4.614	-1.778	-0.622	-3.152	-0.197	0.191	0.217	23	0.123
Downtrend	P80	-5.726	-2.048	-0.668	-3.044	-0.219	0.056	0.176	17	0.109
Downtrend	P90	-5.508	-1.461	-0.627	-3.196	-0.196	0.000	0.000	9	0.060
Uptrend	P70	0.338	0.260	0.049	0.420	-0.117	1.095	0.405	37	0.319
Uptrend	P80	0.351	0.240	0.047	0.792	-0.059	1.130	0.500	18	0.250
Uptrend	P90	-0.124	-0.058	-0.024	-0.406	-0.059	0.957	0.556	9	0.119
Volatility	P70	0.187	0.134	0.017	0.090	-0.191	1.054	0.377	61	0.250
Volatility	P80	0.113	0.071	0.003	0.018	-0.152	1.037	0.405	42	0.198
Volatility	P90	1.022	0.534	0.164	2.702	-0.061	1.578	0.429	21	0.129

2.9.7 Sensitivity and Robustness

We study the dependence of performance on key knobs of the decision layer:

- **Thresholds (P70/P80/P90):** higher thresholds improve selectivity and often increase Profit Factor and reduce drawdowns, at the cost of fewer trades and lower exposure.
- **Temperature τ:** lower τ concentrates $\mathbf{w}$ on the strongest horizons, typically boosting attribution to 12h–24h; higher τ spreads risk across horizons but may dilute edge.
- **Ensemble choice:** median is robust to outliers; validation-weighted leans toward the most reliable split-specific model; mean provides a stable baseline.
- **Costs and slippage:** results are stress-tested by raising fees (e.g., 10–15 bps per side); conclusions should remain qualitatively stable.[1]
- **TP-horizon and scaling:** tuning the TP horizon (e.g., 12h vs 24h) and `tp_scaling` adjusts the take-profit target and the realized win/loss distribution.

Note on adaptivity. Thresholds based on percentiles of $|s|$ were calibrated within each test window for analysis; for deployment, a rolling calibration (e.g., last 30 days) avoids adaptive bias.

2.9.8 Discussion

The trading backtest conducted on regime-specific subsets (Uptrend, Downtrend, Volatility) provides valuable insights into the interplay between predictive signals and execution rules. The first observation is that the number of trades varies substantially across regimes, with the volatility regime naturally generating the highest trade count. This is explained by its longer duration in the dataset, leading to more anchors and therefore more opportunities. Comparisons across regimes must therefore be interpreted with caution, as sample size strongly conditions stability and statistical significance of the reported metrics.

From a design perspective, the strategy can be characterized as deliberately *rigid*. Trade entry is conditioned on high decision thresholds (P70, P80, P90) and exits are governed by a fixed stop-loss at -2% and symmetric take-profit rules, in addition to a forced closure after 24–48 hours. While this configuration provides a clean and reproducible evaluation framework, it also exposes the system to premature exits: in cryptocurrency markets, a 2% adverse movement is often noise rather than a directional reversal. As a result, many trades are closed at a loss before longer-term signals have time to materialize. This explains the high rate of stop-loss hits observed across regimes, especially in downtrend phases, where the strategy performed poorly (Sharpe ratios below zero across all thresholds).

Nevertheless, the results also reveal that the framework is *adaptive* in the sense that performance varies systematically with parameter adjustments. For instance, under the volatility regime, raising the decision threshold to P90 improves discipline, reduces overtrading, and yields a Sharpe ratio above 1.0 with a Profit Factor around 1.58. This suggests that the predictive model is indeed extracting useful

[1] If slippage is material on your venue, re-run with higher effective costs.

signal, but that profitability is highly sensitive to the calibration of execution rules. In other words, the bottleneck is not necessarily in signal generation, but in the rigidity of the trade management layer.

Another important aspect is the balance between precision and recall in directional calls. The backtests highlight that while the hit ratio remains modest (typically 40–50%), Profit Factors above 1 indicate that the average winning trade outpaces losses when conditions are correctly aligned. However, the symmetric risk–reward profile (fixed 2% stop, forced closure after 24–48h) does not exploit this advantage fully. Allowing greater flexibility—e.g., wider or volatility-scaled stops, dynamic profit-taking, or trailing exits—would likely unlock higher Sharpe ratios while retaining robustness.

Moreover, results should not be generalized across all market conditions without caution. Forecasting performance and trading profitability are highly regime-dependent: a model that appears effective in uptrend phases may fail in prolonged downtrends, high-volatility intervals, or during black swan events. Rigorous evaluation therefore requires stratified testing across diverse regimes rather than reporting aggregate metrics alone, as phase-specific weaknesses can otherwise be masked and overall results artificially inflated. Furthermore, many studies ignore transaction costs and slippage in their trading simulations. A strategy may appear extraordinarily profitable in theory (e.g., the +6654% annual return reported by (Omole and Enke 2024) using daily algorithmic trading), but such figures neglect liquidity constraints (slippage) and execution fees, which would drastically reduce actual returns. This oversight risks producing overly optimistic claims about the "profitability" of deep learning approaches. For robust assessment, future work must explicitly integrate costs, slippage, and liquidity considerations into backtesting frameworks.

In summary, this backtest should be interpreted less as a finalized trading system than as an *illustrative stress test under strict constraints*. By design, the criteria are hard and conservative, ensuring that any profitability signal is genuine rather than an artifact of lax rules. The results show that even under these rigid conditions, profitable pockets exist (notably in high-volatility phases), and that parameter adjustments can turn marginal performance into clearly positive outcomes. This underlines both the promise of the predictive framework and the need for a more nuanced execution layer to translate raw model outputs into sustainable trading performance.

2.10 Conclusion

Part II introduced a multi-horizon, multi-timeframe forecasting pipeline designed to jointly predict bounded returns at horizons $\{1h, 4h, 12h, 24h, 48h\}$. The framework fuses representations from H1, H4, and D1 through gated attention, aggregates split-specific models into an ensemble, and translates forecasts into trades via a horizon-consensus rule. Training relies on either fixed or uncertainty-based horizon weighting, while evaluation is conducted under strict, cost-aware, out-of-sample protocols using daily anchors and rolling normalization.

The ensemble weighted by validation performance achieved the strongest generalization on the unseen Split 5, with metrics of *MSE* 0.0004, *MAE* 0.0127, *RMSE* 0.0207, R^2 0.5411, and *DA* 0.6882, consistently outperforming any single split model. Empirical evidence shows that predictive calibration and directional sensitivity are concentrated at 12h–24h horizons, whereas the 1h and 4h horizons remain dominated by noise. Cost-aware backtests confirm the role of market context: a high-confidence policy (P90) achieves Sharpe ratios above one and Profit Factors around 1.58 in high-volatility regimes, yields only marginally positive outcomes in uptrends at moderate thresholds (P70/P80), and performs poorly in downtrends across all thresholds. These findings demonstrate that profitability is not constrained by the predictive model itself, but by the calibration of execution rules, and that averaging across temporal splits stabilizes performance.

Limitations remain. Reported errors are measured in the tanh domain, making them conservative relative to raw-return scale. Short-horizon signals are fragile, and execution rules—fixed stop-losses, symmetric take-profits, and discrete hourly fills—are intentionally rigid. Slippage is not modeled beyond fee stress tests, and performance is highly sensitive to both thresholding and the consensus temperature, with significant degradation observed in persistent downtrends.

Looking ahead, future work should focus on enhancing regime awareness and adaptivity. This includes integrating mixture-of-experts or explicit regime-switching mechanisms, calibrating thresholds and horizon weights dynamically (potentially via Bayesian or uncertainty-aware methods), and refining position sizing strategies. Further gains are expected from cost-aware learning through differentiable backtesting or reinforcement learning approaches that optimize trade management under realistic assumptions of fees and slippage. Enriching the feature space with order-book and funding data, macroeconomic and news embeddings, and cross-asset signals may also improve robustness.

Conclusion

This thesis provides a exploration of deep learning for cryptocurrency market forecasting. First, it assembles a comprehensive H4 BTC/USDT dataset spanning over seven years and constructs an adaptive labelling scheme that uses sliding-window return distributions to define Buy, Hold and Sell classes. Benchmarks of LSTM, GRU, convolutional and transformer-based networks against a tree-based baseline reveal that, despite modest gains over XGBoost, deep learners remain challenged by class imbalance and regime shifts; macro F1-scores rarely exceed 0.39 and Sell signals are seldom identified. Moreover, binary classification and fixed-threshold adjustments do not alleviate the underlying difficulty; model performance drifts across temporal splits, underscoring the need for walk-forward evaluation.

Second, the work shifts to a multi-horizon regression perspective and introduces a multi-timeframe architecture inspired by the Temporal Fusion Transformer. By separately encoding 1-hour, 4-hour and daily sequences and fusing their embeddings through a learnable gate, the model capitalises on the James–Stein effect: joint estimation of correlated targets reduces variance relative to independent models. Validation-weighted ensembling across temporal splits further stabilises predictions. On unseen data, the ensemble achieves $RMSE \approx 0.0207$, $R^2 \approx 0.54$ and directional accuracy near 0.69 in tanh-transformed space; mid-range horizons (12–24 h) exhibit higher predictability and calibration than short horizons (1–4 h), reflecting greater signal-to-noise ratios. Trading simulations that map the forecasts into position signals show that profit is possible but dependent on precise calibration of confidence thresholds, stop-loss levels and horizon weights. The bottleneck lies less in signal generation than in trade execution and risk management; rigid profit-taking rules can negate a valid edge.

Limitations of the study include reliance on a single asset and on technical indicators alone; exogenous signals such as order-book depth, derivatives markets, sentiment and macroeconomic variables were excluded to focus on proof of concept. The tanh transform compresses extreme returns, making reported RMSE values conservative. Data splits extend only to September 2022 for model training and to April–August 2025 for backtests; recent events beyond this period are not captured. Finally, the trading layer uses simple long–flat positions with fixed leverage and stop-loss; more sophisticated strategies may yield different outcomes.

Overall, this thesis demonstrates that deep learning can extract modestly predictive structure from cryptocurrency price series and that multi-timeframe, multi-horizon modelling improves stability. However, real-world deployment demands robust validation schemes, adaptive labelling, uncertainty quantification and rigorous risk management. The research paves the way for future exploration of richer feature sets and decision-making frameworks.

Future Work

Future work could build on these findings through:

- **Dynamic labelling and thresholding:** Develop volatility- or regime-adaptive labelling schemes that adjust class boundaries and decision thresholds according to market conditions, possibly using Bayesian uncertainty estimates.
- **Incorporation of exogenous signals:** Enrich the feature space with order-book data, sentiment scores, macroeconomic indicators and cross-asset relationships to capture drivers beyond technical patterns.
- **Cross-asset and transfer learning:** Extend the multi-timeframe architecture to multiple cryptocurrencies, employing transfer learning or meta-learning to exploit inter-asset similarities and diversify risk.
- **Reinforcement learning for execution:** Replace rule-based trade management with reinforcement learning agents that jointly optimise signal generation, position sizing and exit strategies under realistic transaction costs.
- **Interpretability and risk control:** Integrate attention analyses, SHAP values or counterfactual explanations to understand model drivers, and couple forecasts with probabilistic risk metrics (e.g., value-at-risk) to inform position sizing.

Bibliography

Aggarwal, Ayush, Nikhil Kumar, and Ashish Sureka (2019). "Using LSTM Networks and Ensemble Methods for Stock Market Prediction". In: *Proceedings of the 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 522–525. DOI: 10.1109/CONFLUENCE.2019.8776949.

Albariqi, Mukhamad et al. (2020). "Bitcoin Price Prediction Using Deep Learning Approach". In: *Proceedings of the 2020 3rd International Conference on Computer and Communication Engineering Technology (CCET)*. IEEE.

Biais, Bruno et al. (2018). "Equilibrium Bitcoin Pricing". Working Paper. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3261063.

Bolt, Wilko and Maarten R. C. van Oordt (2019). "On the Value of Virtual Currencies". In: *Journal of Money, Credit and Banking* 51.4, pp. 835–862. DOI: 10.1111/jmcb.12593.

Bouteska, Ahmed and Nader Zoghlami (2024). "Forecasting cryptocurrency prices using ensemble and deep learning methods". In: *Financial Innovation*.

Burniske, Chris and Adam White (2017). *Bitcoin: Ringing the Bell for a New Asset Class*. ARK Invest White Paper. URL: https://research.ark-invest.com/hubfs/1_Download_Files_ARK-Invest/White_Papers/Bitcoin-Ringing-The-Bell-For-A-New-Asset-Class.pdf.

Buzcu, Mustafa et al. (2021). "Sentiment-Aware Hybrid Deep Learning Model for Bitcoin Price Prediction Using FinBERT and LSTM". In: *Proceedings of the 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE.

Chen, Min et al. (2020). "Blockchain-Based Data Sharing and Trading Framework for Industrial Artificial Intelligence". In: *IEEE Transactions on Industrial Informatics* 16.6, pp. 3974–3982.

Chen, Wei, Zilong Xu, and Yongjie Zhang (2021). "Machine learning model for Bitcoin exchange rate prediction using economic and technology determinants". In: *International Journal of Forecasting*.

CoinMarketCap (2025). *Cryptocurrency Prices, Charts and Market Capitalizations*. URL: https://coinmarketcap.com/.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314.

Dyhrberg, Anne Haubo (2016). "Bitcoin, gold and the dollar—A GARCH volatility analysis". In: *Finance Research Letters* 16, pp. 85–92. DOI: 10.1016/j.frl.2015.10.008.

Fama, Eugene F. (1970). "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *The Journal of Finance* 25.2, pp. 383–417. DOI: 10.2307/2325486.

Glaser, Florian et al. (2014). "Bitcoin—Asset or Currency? Revealing Users' Hidden Intentions". In: *Proceedings of the European Conference on Information Systems (ECIS)*, pp. 1–15.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. URL: https://www.deeplearningbook.org/.

Green, Jeremiah, John R. M. Hand, and X. Frank Zhang (2013). "The supraview of return predictive signals". In: *Review of Accounting Studies* 18.3.

Grossman, Sanford J. and Joseph E. Stiglitz (1980). "On the Impossibility of Informationally Efficient Markets". In: *The American Economic Review* 70.3, pp. 393–408.

Gurgul, Henryk, Krzysztof Wójcik, and Lukasz Sliwinski (2025). "Multimodal Deep Learning for Cryptocurrency Trend Prediction using Blockchain and Sentiment Data". In: *Finance Research Letters*.

Hu, Yifan et al. (2025). *FinTSB: A Comprehensive and Practical Benchmark for Financial Time Series Forecasting*. URL: https://arxiv.org/abs/2502.18834.

James, W. and C. Stein (1961). "Estimation with Quadratic Loss". In: *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1*. University of California Press, pp. 361–379.

Jaquart, Patrick, David Dann, and Carl Martin (2020). "Machine Learning for Bitcoin Pricing — A Structured Literature Review". In: *Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS)*. Karlsruhe Institute of Technology.

Ji, Qiang, Elie Bouri, and David Roubaud (2019). "Information spillovers across Bitcoin markets: A rolling sample VAR approach". In: *Finance Research Letters*.

Kara, Yasemin, M. A. Boyacioglu, and Omer K. Baykan (2011). "Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines: The Sample of the Istanbul Stock Exchange". In: *Expert Systems with Applications*.

Kehinde, Ayodele and Warren Smith (2025). "Helformer: Holt-Winters Enhanced Transformer for Cryptocurrency Forecasting". In: *Journal of Computational Finance*.

Kendall, Alex, Yarin Gal, and Roberto Cipolla (2018). "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kong, Wei et al. (2019). "Short-term residential load forecasting based on LSTM recurrent neural network". In: *IEEE Transactions on Smart Grid*. DOI: 10.1109/TSG.2017.2753802.

Leitner, Christoph and Kurt Hornik (2016). "ForeCA: An R Package for Forecastable Component Analysis". In: *Journal of Statistical Software* 74.11, pp. 1–29. DOI: 10.18637/jss.v074.i11. URL: https://doi.org/10.18637/jss.v074.i11.

Lim, Bryan et al. (2020). "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting". In: *arXiv preprint arXiv:1912.09363v3*. URL: https://arxiv.org/abs/1912.09363.

Mahfooz, Sohail et al. (2024). "Bitcoin Price Prediction Using Attention-based BiLSTM Model with Sentiment Analysis". In: *Expert Systems with Applications* 235.

Maji, Gourav and Manisha (2019). "Bitcoin Price Forecasting using Deep Learning Models". In: *Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE.

Mazinani, Navid, Leila Ghods, and Mehrdad Amiri (2024). "Time Series Forecasting in Cryptocurrency: Transformer or CNN-RNN Hybrid?" In: *Applied Soft Computing*.

McNally, Sean, Jason Roche, and Simon Caton (2018). "Predicting the Price of Bitcoin Using Machine Learning". In: *Proc. of Euromicro Conf. on Parallel, Distributed and Network-Based Processing*, pp. 339–343. DOI: 10.1109/PDP2018.2018.00060.

Mishkin, Frederic S. (2018). *The Economics of Money, Banking, and Financial Markets*. 11th. Pearson.

Murray, Alexander and Joseph Thomas (2020). "The Complexity of Crypto: Forecasting Challenges in Volatile Markets". In: *Journal of Financial Data Science*.

Nakamoto, Satoshi (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. White paper. URL: https://bitcoin.org/bitcoin.pdf.

Narayanan, Arvind et al. (2016). *Bitcoin and Cryptocurrency Technologies*. Princeton University Press.

Omole, Oluwadamilare and David Enke (2024). "Deep learning for Bitcoin price direction prediction: models and trading strategies empirically compared". In: *Financial Innovation* 10.1, pp. 1–26. DOI: 10.1186/s40854-024-00643-1.

Ouyang, Zuokun (2023). "Time Series Forecasting: From Econometrics to Deep Learning". PhD thesis. Université d'Orléans.

Pagnotta, Emiliano and Andrea Buraschi (2018). "An Equilibrium Valuation of Bitcoin and Decentralized Network Assets". Working Paper.

Peng, Kai, Jianhua Wu, and Rui Zhou (2024). "Attention-based Deep Learning Framework for High-Frequency Cryptocurrency Prediction". In: *Expert Systems with Applications*.

Rasp, Stephan and Nils Thuerey (2021). "Data-driven medium-range weather prediction with a ResNet pretrained on climate simulations: A new model for WeatherBench". In: *Journal of Advances in Modeling Earth Systems* 13.2. DOI: 10.1029/2020MS002405.

Schilling, Linda and Harald Uhlig (2019). "Some Simple Bitcoin Economics". In: *Journal of Monetary Economics*.

Statista Research Department (2023). "Cryptocurrency market size forecast worldwide 2019-2030". In: URL: https://www.statista.com/statistics/1202468/global-cryptocurrency-market-size/.

Tanwar, Sakshi and Ruchi Srivastava (2021). "Deep Learning-Based Cryptocurrency Price Prediction Scheme With Inter-Dependent Relations". In: DOI: DOI:10.1109/ACCESS.2021.3117848.