Mitigating Distribution Shift in Stock Price Data via Return-Volatility Normalization for Accurate Prediction

Anonymous Author(s)

Abstract

How can we address distribution shifts in stock price data to improve stock price prediction accuracy? Stock price prediction has attracted attention from both academia and industry, driven by its potential to uncover complex market patterns and enhance decision-making. However, existing methods often fail to handle distribution shifts effectively, focusing on scaling or representation adaptation without fully addressing distributional discrepancies and shape misalignments between training and test data.

In this paper, we propose ReVol (Return-Volatility Normalization for Mitigating Distribution Shift in Stock Price Data), a robust method for stock price prediction that explicitly addresses the distribution shift problem. ReVol leverages three key strategies to mitigate these shifts: (1) normalizing price features to remove sample-specific characteristics, including return, volatility, and price scale, (2) employing an attention-based module to estimate these characteristics accurately, thereby reducing the influence of market anomalies, and (3) reintegrating the sample characteristics into the predictive process, restoring the traits lost during normalization. Additionally, ReVol combines geometric Brownian motion for long-term trend modeling with neural networks for short-term pattern recognition, unifying their complementary strengths. Extensive experiments on real-world datasets demonstrate that ReVol enhances the performance of the state-of-the-art backbone models in most cases, achieving an average improvement of more than 0.03 in IC and over 0.7 in SR across various settings. These results underscore the importance of directly confronting distribution shifts and highlight the efficacy of our integrated approach for stock price prediction.

ACM Reference Format:

Anonymous Author(s). 2025. Mitigating Distribution Shift in Stock Price Data via Return-Volatility Normalization for Accurate Prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

How can we address distribution shifts in stock price data to improve the accuracy of stock price prediction? Distribution shifts — distribution discrepancies across samples — pose a fundamental challenge in data mining and machine learning. This issue is especially critical in stock price prediction, where market conditions evolve continually due to economic factors, geopolitical events, and investor sentiment. These shifts occur not only over time as market conditions change but also across different stocks, each exhibiting unique statistical properties such as return and volatility. Successfully overcoming these shifts is crucial for constructing robust and trustworthy predictive models, which in turn support more reliable investment strategies and enhance financial outcomes.

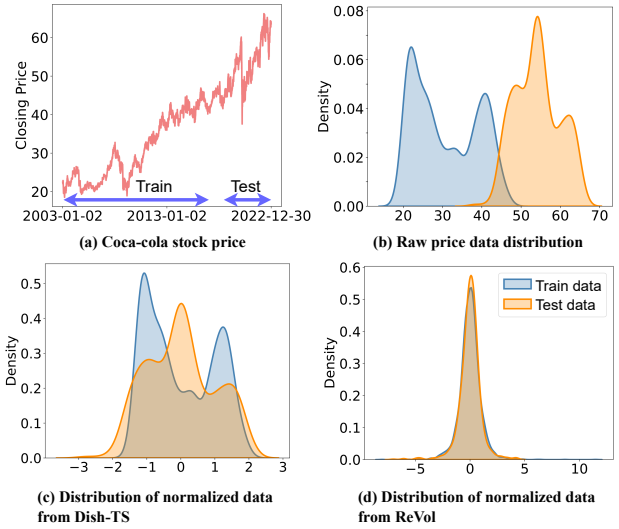

Figure 1: (a) shows a stock price dataset, including both training and test periods. (b) depicts distribution difference between raw price data of the training and test periods. (c) Dish-TS [5] reduces the distribution difference; however, it does not fully align the distributions. (d) ReVol effectively aligns the two distributions with minimal discrepancy.

Despite extensive work on stock price prediction, existing methods [6, 12, 18] often fail to adequately deal with distribution shifts. They do not explicitly mitigate distributional differences across samples, limiting their generalization in non-stationary financial time series. Methods for addressing distribution shifts [5, 8] unify distribution parameters across samples, while failing to align distribution shapes. Figure 1 illustrates how raw price data and normalized data distributions differ between training and test periods. Others [13, 19, 20] rely on adaptation to the representation space, increasing the complexity of the model without directly resolving the distribution shift. These limitations hinder the robustness and accuracy of the prediction under changing market conditions.

We propose ReVol (Return-Volatility Normalization for Mitigating Distribution Shift in Stock Price Data), a method for robust stock price prediction that explicitly addresses distribution shifts by normalizing individual stock characteristics. The main ideas of ReVol

CIKM '25, November 10–14, 2025, Seoul, Republic of Korea

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Table 1: Symbols and descriptions.

Symbol	Definition
S_t	Stock price at time t
B_t	Value of Brownian motion at time t
μ	Return of a stock
σ	Volatility of a stock
$S_t^c, S_t^o, S_t^h, S_t^l$	Closing, opening, the highest, and the lowest prices at time step t
$\epsilon_t^c, \epsilon_t^o, \epsilon_t^h, \epsilon_t^l$	Error terms of closing, opening, the highest, and the lowest prices at time step t
T	The current time step
r	Time interval between opening and previous day's closing prices
u	Time interval between the highest and previous day's closing prices
v	Time interval between the lowest and previous day's closing prices
$\mathbf{x}_t$	Price feature vector at time step t
$\tilde{\mathbf{x}}_t$	Feature vector at time step t from fully connected layer
$\mathbf{h}_t$	Hidden state vector at time step t from LSTM
α_t	Attention weight at time step t
θ	Parameters of the neural network backbone
f_θ	Neural network backbone
w	The window size
β	Hyperparameter to control the weight of guidance loss

are as follows. First, ReVol mitigates distribution shifts by removing sample-specific attributes, such as return, volatility, and price scale, from the input data. Second, it accurately estimates these characteristics through an attention-based weighted averaging technique that alleviates the impact of transient market anomalies. Third, ReVol reintegrates these characteristics into a prediction pipeline to recover the information removed during normalization.

Our contributions are summarized as follows:

- **Addressing distribution shifts in stock price prediction.** We propose ReVol, a novel approach to address distribution shifts by normalizing individual sample characteristics, including return, volatility, and price scale. Figure 1 depicts that the distribution gap between training and test data is mitigated by ReVol.
- **Integration of short-term patterns and long-term trends.** ReVol unifies geometric Brownian motion (GBM) for modeling long-term price behavior with a neural network for capturing nonlinear short-term patterns, thereby leveraging the complementary strengths of both methods.
- **Experiments.** We demonstrate that ReVol improves the performance of the state-of-the-art backbone models in most scenarios, delivering an average gain of more than 0.03 in IC and over 0.7 in SR. Moreover, ReVol effectively narrows the distribution gap between training and test data via the proposed GBM-based normalization.

The definitions of the symbols in this paper is listed in Table 1. The code and datasets are available at https://github.com/AnonymousCoder2357/ReVol.

2 Related Works

2.1 Geometric Brownian Motion

Geometric Brownian motion (GBM) [3] is one of the most popular models for stock price dynamics. GBM is widely used in various financial tasks, such as option pricing and portfolio optimization, due to its suitability and simplicity. The GBM model assumes that the logarithm of stock prices follows a Brownian motion with drift, allowing for continuous price movements. In GBM, the stock price S_t at time t is modeled as:

$$S_t = S_0 e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma B_t} \tag{1}$$

where μ, σ, and B_t are the return, volatility of the stock, and the value of Brownian motion at time t, respectively. Brownian motion (BM) B is a stochastic process with the following properties:

(1) $B_0 = 0$
(2) if $0 \le s_1 \le t_1 \le s_2 \le t_2$, then $B_{t_1} - B_{s_1}$ and $B_{t_2} - B_{s_2}$ are independent
(3) $B_t - B_s \sim \mathcal{N}(0, t-s)$ (for $0 \le s \le t$)
(4) $t \to B_t$ is almost surely continuous

where B_t is the value of BM at time t.

For discrete time t, S_t is expressed in terms of S_{t-1} by representing the BM part as $\epsilon_t \sim \mathcal{N}(0,1)$ with the following Lemma.

Lemma 2.1. *When time is discretized, the stock price S_t at time $t \in \mathbb{N}$ is expressed as follows:*

$$S_t = S_{t-1} e^{(\mu - \frac{1}{2}\sigma^2) + \sigma\epsilon_t} \tag{2}$$

where the error term $\epsilon_t \sim \mathcal{N}(0,1)$. ϵ_t and ϵ_s are independent for $s < t$ and $s \in \mathbb{N}$.

Proof. See Appendix A.1 □

The assumption that ϵ_t and ϵ_s are independent for $s < t$ and $t, s \in \mathbb{N}$ means that future stock prices cannot be predicted from past prices.

2.2 Deep Learning-Based Stock Price Prediction

With the advancement of deep neural networks, attempts to predict stock prices using the time series processing capabilities of recurrent neural networks have emerged [14, 16]. The attention mechanism has been used to enhance the performance of RNN-based models [6, 10, 15]. GCN or Transformer architectures have been used to leverage the correlations between stocks [11, 12, 18]. However, these methods struggle to address the distribution shift problem since they fail to alleviate distributional discrepancy between training and test data.

2.3 Methods for Addressing Distribution Shift

As distribution shift has emerged as a critical issue in time series analysis, various methods have been proposed to address it in the time series and financial domains. RevIN [8] and Dish-TS [5] equalize the distribution parameters, mean and standard deviation, for each sample through instance normalization. However, financial time series data do not generally follow a normal distribution. As a result, even if the mean and standard deviation are equalized, the distribution shapes still differ across samples due to higher-order characteristics such as skewness and kurtosis. There have been also

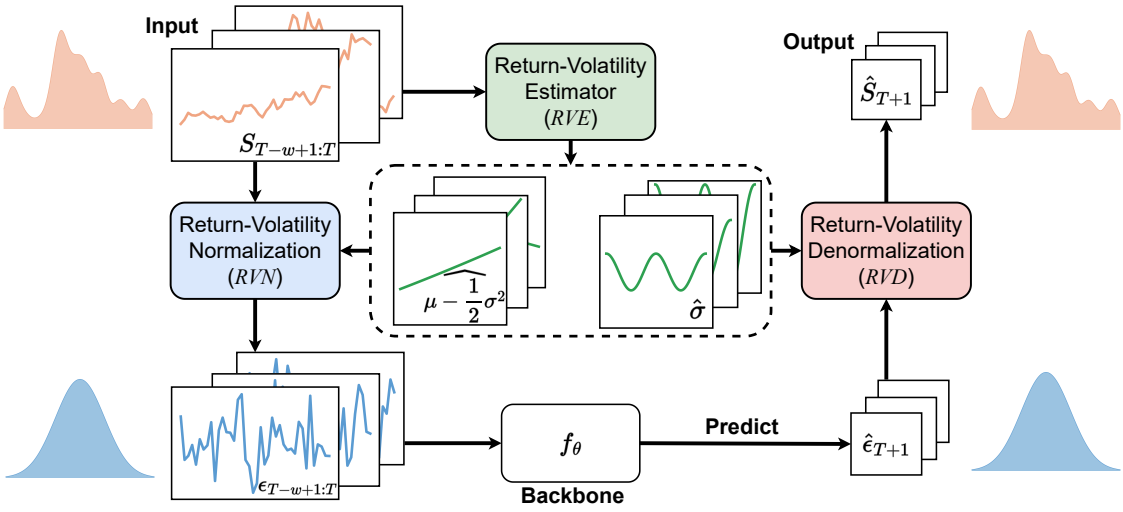

Figure 2: An overview of ReVol.

methods to detect distribution shifts that may occur during the evolution of stock time series, and adapt to various market conditions [13, 19, 20]. However, these methods rely on the representation space to adapt to the distribution on its own, which not only increases the burden on the model but also fails to explicitly address distribution shift. In this work, we aim to resolve the distribution shift problem by making the distribution of each sample identical to improve the accuracy of stock price prediction.

3 Proposed Method

We propose ReVol (Return-Volatility Normalization for Mitigating Distribution Shift in Stock Price Data), an accurate method for stock price prediction by mitigating the distribution shift problem.

3.1 Overview

Our problem definition is as follows:

Problem 1 (Stock Price Prediction.). ***Given*** *the historical prices $S^o_{T-w+1:T}$, $S^h_{T-w+1:T}$, $S^l_{T-w+1:T}$, and $S^c_{T-w+1:T}$ from day $T-w+1$ up to T, where S^o_t, S^h_t, S^l_t, and S^c_t are opening, the highest, the lowest,and closing prices on day t, respectively,* ***find*** *the closing price S^c_{T+1} for the next day.*

To achieve accurate prediction addressing the distribution shift problem, ReVol is designed to address the following challenges.

(1) **Distribution shift in stock price data.** The distribution of stock price data varies across stocks and over time. This distribution shift problem prevents accurate stock price prediction. How can we reduce the discrepancies between distributions?

(2) **Estimating sample characteristics.** We must accurately estimate and remove individual sample characteristics, from stock price data to reduce distribution discrepancies. However, due to external shocks, stock prices sometimes deviate temporarily from their true distribution. Such anomalies hinder the estimation of the true distribution. How can we estimate the true distribution accurately?

(3) **Reintegrating sample characteristics.** Removing sample characteristics from the input data prohibits the prediction model from properly reconstructing the data distribution to predict future stock prices. How can we reintegrate the individual sample characteristics into the prediction?

Our proposed ReVol addresses the challenges with the following ideas. Figure 2 depicts the overall process of ReVol.

(1) **Return-volatility normalization (Section 3.2).** Inspired by GBM, we normalize the input stock price data with sample characteristics (return, volatility, and price scale), to remove them from the data. This effectively reduces distributional discrepancies across samples. As a result, we obtain historical error terms, which serve as inputs to the neural network backbone.

(2) **Return-volatility estimator (Section 3.3).** We calculate return and volatility using a weighted average with attention weights. This approach mitigates the impact of anomalies that disrupt characteristic estimation. Consequently, we acquire precisely estimated sample characteristics.

(3) **Return-volatility denormalization (Section 3.4).** We denormalize the future error term predicted by the neural network backbone with sample characteristics to obtain the final prediction. By doing so, we restore the sample characteristics lost during the normalization process.

Each of the above idea corresponds to each main module of ReVol. First, Return-Volatility Normalization (*RVN*) module normalizes historical stock prices with sample characteristics to generate historical error terms, which act as inputs for the neural network backbone. The backbone model does not suffer from distribution shift since the error terms follow an identical distribution across samples. Second, Return-Volatility Estimator (*RVE*) module accurately estimates sample characteristics through an attention mechanism, which are subsequently used in *RVN*. Lastly, Return-Volatility Denormalization (*RVD*) module denormalizes the future error term predicted by the backbone model, reintegrating information removed from *RVN*. The denormalized output provides the final future price prediction.

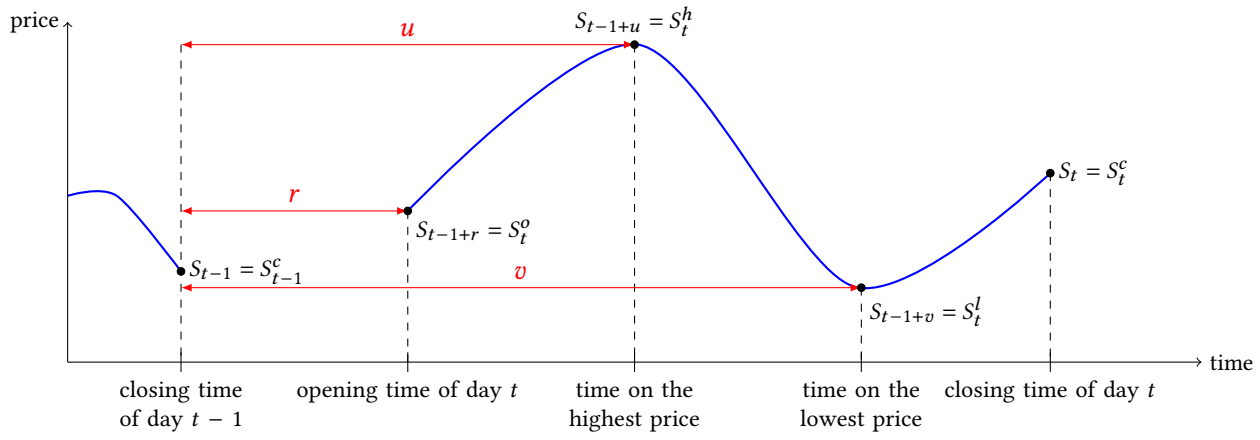

Figure 3: Opening, the highest, the lowest, and closing prices on a continuous time axis.

3.2 Return-Volatility Normalization

To reduce the distributional discrepancy, we propose Return-Volatility Normalization (*RVN*) module to normalize stock price data, inspired by GBM [3]. *RVN* normalizes historical stock prices $S^o_{T-w+1:T}$, $S^h_{T-w+1:T}$, $S^l_{T-w+1:T}$, and $S^c_{T-w+1:T}$ for w days up to day T into normalized inputs $\epsilon^o_{T-w+1:T}$, $\epsilon^h_{T-w+1:T}$, $\epsilon^l_{T-w+1:T}$, and $\epsilon^c_{T-w+1:T}$, which serve as inputs to the backbone model. Note that $\epsilon^o_{T-w+1:T}$, $\epsilon^h_{T-w+1:T}$, $\epsilon^l_{T-w+1:T}$, and $\epsilon^c_{T-w+1:T}$ are not random variables as in Equation (2), but rather samples in this section. The backbone model is unaffected by distribution shift, as the error terms share an identical distribution across samples.

To normalize the stock prices with the sample characteristics, we must first define the sample characteristics. In the widely used stock modeling method GBM, stock characteristics are defined as return, volatility, and price scale. We assume that stock prices follow a GBM, except for the independence assumption (refer to property 2 of GBM in Section 2.1). Observing Equation (2), the equation consists of the variables μ (return), σ (volatility), S_{t-1} (price scale), and ϵ_t (error term). Note that 1) μ, σ, and S_{t-1} represent sample characteristics, and 2) ϵ_t is sampled from a standard normal distribution. Therefore, we eliminate distributional discrepancy by constructing an input with error terms.

We first briefly explain the process of estimating sample characteristics in Section 3.2.1 and then describe normalization methods for closing, opening, and the highest/lowest prices in Sections 3.2.2, 3.2.3, and 3.2.4, respectively.

3.2.1 Return and Volatility Estimation. We first estimate the sample characteristics—return and volatility—based on GBM, using proposed Return-Volatility Estimator (*RVE*), which is an attention-based module designed to accurately estimate return and volatility. The estimation is expressed as follows.

$$\widehat{\mu - \frac{1}{2}\sigma^2}, \hat{\sigma} = RVE\left(S^o_{T-w+1:T}, S^h_{T-w+1:T}, S^l_{T-w+1:T}, S^c_{T-w+1:T}\right). \tag{3}$$

3.2.2 Normalizing Closing Price. We obtain the error term ϵ^c_t by normalizing the logarithmic daily return using the estimated return and volatility, as shown in the following Lemma.

Lemma 3.1. *An error term ϵ^c_t which corresponds to the closing price at day t, is obtained as follows:*

$$\epsilon^c_t = \frac{1}{\hat{\sigma}}\left(\log\frac{S^c_t}{S^c_{t-1}} - \left(\widehat{\mu - \frac{1}{2}\sigma^2}\right)\right). \tag{4}$$

Furthermore, ϵ^c_t is a sample drawn from a standard normal distribution:

$$\epsilon^c_t \overset{sample}{\sim} \mathcal{N}(0,1). \tag{5}$$

Proof. See Appendix A.2 □

The error term ϵ^c_t represents the error of the closing price on day t relative to the closing price on day $t-1$. Thus, we exploit ϵ^c_t as a feature for training in place of S^c_t, the closing price on day t.

3.2.3 Normalizing Opening Price. A simple way to normalize the opening price is to apply Equation (4) in the same way, as follows:

$$\epsilon^o_t = \frac{1}{\hat{\sigma}}\left(\log\frac{S^o_t}{S^o_{t-1}} - \left(\widehat{\mu - \frac{1}{2}\sigma^2}\right)\right). \tag{6}$$

However, in this case, the error term ϵ^o_t represents the change from the opening price on day $t-1$ to the opening price on day t. Then, the relationship between the opening and closing prices cannot be captured. Therefore, we propose to place the opening and closing prices on a continuous time axis $[0,\infty)$ and calculate error terms that capture the relationship between them.

Let S_t represent the stock price at time t on a continuous time axis $[0,\infty)$. In this axis, $t \in \mathbb{N}$ represents the closing time of day t where we regard t as an integer. That is, $S_t = S^c_t$ for $t \in \mathbb{N}$. We place the opening, highest, and lowest prices at day t on this time axis. Let $0 \le r, u, v \le 1$ represent the time intervals between the closing price on day $t-1$ and the opening, highest, and lowest prices on day t, respectively, as shown in Figure 3. Then, on the continuous time axis, $S_{t-1+r} = S^o_t$, $S_{t-1+u} = S^h_t$, and $S_{t-1+v} = S^l_t$.

By Equation (1), the opening price S_{t-1+r} is expressed in terms of the previous day's closing price S_{t-1}:

$$S_{t-1+r} = S_{t-1}e^{(\mu-\frac{1}{2}\sigma^2)r+\sigma\epsilon^o_t} \tag{7}$$

where $\epsilon^o_t \overset{sample}{\sim} \mathcal{N}(0,r)$ is the error term sampled from $B_{t-1+r} - B_{t-1}$. ϵ^o_t is determined if r is known since other variables (S_{t-1+r}, S_{t-1}, μ, and σ) are already known. However, determining the time interval between the opening and previous day's closing prices

is not straightforward. We might consider setting r to 0 since the opening price is the immediate price following the previous day's closing price. On the other hand, due to price fluctuations during after-hours trading, the opening price may differ from the previous day's closing price, meaning that r cannot be set to 0. Therefore, we need to estimate r before normalizing the opening price. We estimate r based on the assumption that stock prices follow a geometric modeling with respect to time.

Lemma 3.2. *By minimizing a mean squared error, r is estimated as follows:*

$$\hat{r} = \frac{\mathbb{E}\left[\log \frac{S_t^c}{S_{t-1}^c} \log \frac{S_t^o}{S_{t-1}^c}\right]}{\mathbb{E}\left[\left(\log \frac{S_t^c}{S_{t-1}^c}\right)^2\right]}. \tag{8}$$

Proof. See Appendix A.3 □

As a result, we obtain the error term ϵ_t^o with the following Lemma.

Lemma 3.3. *Assume that Equation (8) accurately estimates r, i.e., $\hat{r} = r$. Then, the error term ϵ_t^o corresponding to the opening price on day t, is obtained as follows:*

$$\epsilon_t^o = \frac{1}{\hat{\sigma}}\left(\log \frac{S_t^o}{S_{t-1}^c} - \left(\widehat{\mu - \frac{1}{2}\sigma^2}\right)\hat{r}\right). \tag{9}$$

Furthermore, $\frac{1}{\sqrt{\hat{r}}}\epsilon_t^o$ is a sample drawn from a standard normal distribution:

$$\frac{1}{\sqrt{\hat{r}}}\epsilon_t^o \overset{sample}{\sim} \mathcal{N}(0,1). \tag{10}$$

Proof. See Appendix A.4 □

3.2.4 Normalizing the Highest and Lowest Prices. The highest price also cannot be normalized to the error term using the previous day's highest price, since it fails to capture the relationship with the closing price. Furthermore, it is also impossible to normalize it in the same way as the opening price as shown in Equation (9), since the time interval u between the highest and previous day's closing prices is unknown. Thus, we define ϵ_t^h and normalize the highest price as following Lemma.

Lemma 3.4. *Let*

$$\epsilon_t^h = \frac{1}{\hat{\sigma}} \log \frac{S_t^h}{S_{t-1}^c}. \tag{11}$$

Then, ϵ_t^h is a sample drawn from a half-normal distribution, which is derived by taking the absolute value of a standard normal variable.

Proof. See Appendix A.5 □

We also normalize the lowest price in the same way:

$$\epsilon_t^l = \frac{1}{\hat{\sigma}} \log \frac{S_t^l}{S_{t-1}^c}. \tag{12}$$

Note that ϵ_t^l also is a sample drawn from a half-normal distribution.

3.3 Return-Volatility Estimator

Return-Volatility Estimator (*RVE*) is designed to accurately estimate sample characteristics from historical stock prices. The return and volatility are distribution parameters of the stock prices. Stock prices sometimes deviate temporarily from their true distribution due to external shocks. For example, stock prices may experience a temporary surge due to expectations around a new product launch, or a sharp decline due to institutional investors liquidating their positions for short-term gains. These temporary fluctuations create noise in the data, making it difficult to accurately estimate the underlying distribution of stock prices. We propose to estimate the distribution parameters using a weighted average based on attention weights to handle such anomalies flexibly.

We observe that $\log \frac{S_t^c}{S_{t-1}^c} = \mu - \frac{1}{2}\sigma^2 + \sigma\epsilon_t^c \sim \mathcal{N}(\mu - \frac{1}{2}\sigma^2, \sigma^2)$ by applying Equation (2) to closing prices. According to this observation, a simple approach to estimate $\mu - \frac{1}{2}\sigma^2$ and σ is to compute the arithmetic mean as follows:

$$\widehat{\mu - \frac{1}{2}\sigma^2} = \frac{1}{w}\sum_{t=T-w+1}^{T} \log \frac{S_t^c}{S_{t-1}^c} \tag{13}$$

$$\hat{\sigma} = \sqrt{\frac{1}{w}\sum_{i=T-w+1}^{T}\left(\log \frac{S_t^c}{S_{t-1}^c} - \widehat{\mu - \frac{1}{2}\sigma^2}\right)^2}. \tag{14}$$

However, this approach is significantly affected by anomalies that deviate from the true distribution, making it difficult to accurately estimate the distribution parameters. Our main idea is to estimate $\mu - \frac{1}{2}\sigma^2$ and σ using a weighted average based on attention weights as follows:

$$\widehat{\mu - \frac{1}{2}\sigma^2} = \sum_{t=T-w+1}^{T} \alpha_t \log \frac{S_t^c}{S_{t-1}^c} \tag{15}$$

$$\hat{\sigma} = \sqrt{\sum_{t=T-w+1}^{T} \alpha_t \left(\log \frac{S_t^c}{S_{t-1}^c} - \widehat{\mu - \frac{1}{2}\sigma^2}\right)^2} \tag{16}$$

where α_t is the attention weight at time step t, calculated by Equation (17). Note that $\sum_{t=T-w+1}^{T} \alpha_t = 1$. This approach flexibly handles anomalies by dynamically adjusting the average weights.

Attention module structure. We describe the structure of the attention module to compute the attention weight α_t. We exploit attention LSTM [6] to generate query and key vectors that effectively capture sequential dependencies in time-series data.

We begin by transforming price feature vector $\mathbf{x}_t \in \mathbb{R}^D$ at each time step using a fully connected layer with a hyperbolic tangent activation function:

$$\tilde{\mathbf{x}}_t = tanh(\mathbf{W}\mathbf{x}_t + \mathbf{b}),$$

where the parameters $\mathbf{W} \in \mathbb{R}^{E\times D}$ and $\mathbf{b} \in \mathbb{R}^E$ are shared across all time steps t. This transformation produces a new feature representation before the data are passed into the LSTM, improving the network's ability to capture temporal patterns [6].

The feature vectors $\tilde{\mathbf{x}}_{T-w+1}, \ldots, \tilde{\mathbf{x}}_T$ are then passed into the LSTM and transformed into hidden state vectors $\mathbf{h}_{T-w+1}, \ldots, \mathbf{h}_T$. The hidden state vector $\mathbf{h}_T$ from the last time step represents past information and serves as the query vector. Finally, the attention weight α_t at time step t is calculated as follows:

$$\alpha_t = \frac{e^{\mathbf{h}_t^\top \mathbf{h}_T}}{\sum_{i=T-w+1}^{T} e^{\mathbf{h}_i^\top \mathbf{h}_T}}. \tag{17}$$

Guidance loss. We introduce a guidance loss to ensure robust performance of the *RVE*. It is challenging for the *RVE* to infer appropriate attention weights from the beginning. Additionally, we observe that ReVol with the return and volatility calculated through a arithmetic average, as in Equation (13) and (14), also leads to performance improvement of the backbone model.

Therefore, we propose to guide $\widehat{\mu - \frac{1}{2}\sigma^2}$ to resemble the arithmetic mean of the logarithmic daily return through minimizing guidance loss $\left(\frac{1}{w}\sum_{t=T-w+1}^{T} \log \frac{S_t^c}{S_{t-1}^c} - \widehat{\mu - \frac{1}{2}\sigma^2}\right)^2$. The guidance loss is combined with the mean squared error loss to form the final optimization loss, leading to robust performance.

3.4 Return-Volatility Denormalization

Return-Volatility Denormalization (*RVD*) module denormalizes the backbone model's prediction to produce the final output. It reintegrates the information of sample characteristics lost in the normalization process into the prediction of the backbone model.

A naive approach to predict future prices is to train a neural network that maps the normalized price features to the future price. However, this approach is inadequate since the future price inherently contains individual stock characteristics. Specifically, the future stock price $S_{T+1} = S_T e^{(\mu - \frac{1}{2}\sigma^2) + \sigma\epsilon_{T+1}}$ contains information about the sample characteristics: return, volatility, and price scale. It is difficult to predict future stock prices from the error terms since they do not contain such information. Thus, instead of directly forecasting the future stock price, we first predict the future error term ϵ_{T+1}, which does not include sample characteristics. After that, we denormalize the predicted error term using stock characteristics to ultimately predict the future stock price.

To achieve this, we replace the BM part of the GBM with a neural network backbone:

$$\hat{S}_{T+1}^c = S_T^c e^{\left(\widehat{\mu - \frac{1}{2}\sigma^2}\right) + \hat{\sigma} f_\theta(\epsilon^o_{T-w+1:T}, \epsilon^h_{T-w+1:T}, \epsilon^l_{T-w+1:T}, \epsilon^c_{T-w+1:T})} \tag{18}$$

where θ is the backbone model parameters, f_θ is the neural network backbone, and w is the window size. Breaking down Equation (18) into two steps, the backbone model f_θ first predicts the future error term $\hat{\epsilon}^c_{T+1} = f_\theta(\epsilon^o_{T-w+1:T}, \epsilon^h_{T-w+1:T}, \epsilon^l_{T-w+1:T}, \epsilon^c_{T-w+1:T})$, and then the predicted future error term $\hat{\epsilon}_{T+1}$ is denormalized with the sample characteristics to $S_T^c e^{\left(\widehat{\mu - \frac{1}{2}\sigma^2}\right) + \hat{\sigma}\hat{\epsilon}^c_{T+1}}$. As mentioned in Section 2.1, the error term ϵ_{T+1} originally represents the BM part.

Due to the independence assumption that future prices are independent of past prices, GBM cannot predict future prices. However, by eliminating the independence assumption and replacing the BM part with a neural network, the model gains predictive power. The backbone model can be any existing time series prediction model or a stock price prediction model. From Equation (18), ReVol considers both learned patterns of the historical error terms and inherent price dynamics reflected in the sample characteristics.

Model training. The backbone model and *RVE* are trained to minimize the following final optimization loss:

Algorithm 1 Training Algorithm for ReVol

Input: historical stock price dataset
Output: optimal neural network parameter θ

for each mini-batch **do**
  for each sample **do**
    $\widehat{\mu - \frac{1}{2}\sigma^2}, \hat{\sigma} \leftarrow RVE(S^o_{T-w+1:T}, S^h_{T-w+1:T}, S^l_{T-w+1:T}, S^c_{T-w+1:T})$
    $\epsilon^o_{T-w+1:T}, \epsilon^h_{T-w+1:T}, \epsilon^l_{T-w+1:T}, \epsilon^c_{T-w+1:T} \leftarrow$
    $RVN\left(S^o_{T-w+1:T}, S^h_{T-w+1:T}, S^l_{T-w+1:T}, S^c_{T-w+1:T}; \widehat{\mu - \frac{1}{2}\sigma^2}, \hat{\sigma}\right)$
    $\hat{S}^c_{T+1} \leftarrow S^c_T e^{\left(\widehat{\mu - \frac{1}{2}\sigma^2}\right) + \hat{\sigma} f_\theta(\epsilon^o_{T-w+1:T}, \epsilon^h_{T-w+1:T}, \epsilon^l_{T-w+1:T}, \epsilon^c_{T-w+1:T})}$
  end for
  Minimize $\mathbb{E}\Big[\left(\frac{\hat{S}^c_{T+1}}{S^c_T} - \frac{S^c_{T+1}}{S^c_T}\right)^2$
  $+\beta\left(\frac{1}{w}\sum_{t=T-w+1}^{T} \log \frac{S^c_t}{S^c_{t-1}} - \widehat{\mu - \frac{1}{2}\sigma^2}\right)^2\Big]$ using Adam [9]
end for

Table 2: Summary of datasets.

Data	Stocks	Days	Dates
U.S.[1]	178	5035	2003-01-02 to 2022-12-30
China[1]	171	2430	2011-01-06 to 2020-12-31
U.K.[1]	21	2383	2014-01-06 to 2024-01-10
Korea[1]	220	2170	2014-01-02 to 2022-11-01

[1] https://github.com/AnonymousCoder2357/ReVol

Table 3: An input feature vector $\mathbf{x}_t$ at day t.

Feature	Calculation
x_t^o	$x_t^o = S_t^o / S_t^c - 1$
x_t^h	$x_t^h = S_t^h / S_t^c - 1$
x_t^l	$x_t^l = S_t^l / S_t^c - 1$
x_t^c	$x_t^c = S_t^c / S_{t-1}^c - 1$

$$\mathbb{E}\left[\left(\frac{\hat{S}^c_{T+1}}{S^c_T} - \frac{S^c_{T+1}}{S^c_T}\right)^2 + \beta\left(\frac{1}{w}\sum_{t=T-w+1}^{T} \log \frac{S^c_t}{S^c_{t-1}} - \widehat{\mu - \frac{1}{2}\sigma^2}\right)^2\right]. \tag{19}$$

The left term is the MSE loss for daily return prediction, the right term is the guidance loss, and β is a hyperparameter to control its weight.

Algorithm 1 illustrates the whole training process.

4 Experiments

We present experimental results to answer the following research questions about ReVol:

- Q1. **Performance (Section 4.2).** Does ReVol outperform previous methods in stock price prediction?
- Q2. **Comparison with other normalization methods (Section 4.3).** How well does ReVol address distribution shifts in stock price data compared to other normalization methods?
- Q3. **Hyperparameter sensitivity (Section 4.4).** How sensitive is ReVol to hyperparameters?

Table 4: Stock prediction performance of ReVol and baselines measured by Information Coefficient (IC), Rank Information Coefficient (RIC), and Sharpe Ratio (SR). ReVol consistently improves the performance of backbone models in most cases, showing an average improvement of over 0.03 in IC and over 0.7 in SR.

Method	U.S.			China			U.K.			Korea		
	IC	RIC	SR	IC	RIC	SR	IC	RIC	SR	IC	RIC	SR
LSTM [14]	0.019	**0.026**	0.642	0.015	0.004	1.283	0.040	0.014	0.820	0.043	0.029	2.092
+ ReVol	**0.022**	0.021	**1.001**	**0.042**	**0.030**	**2.628**	**0.108**	**0.056**	**1.035**	**0.044**	**0.045**	**2.338**
GRU [4]	0.020	**0.027**	0.656	0.015	0.008	1.181	0.037	0.020	0.382	0.043	0.023	2.008
+ ReVol	**0.022**	0.022	**0.901**	**0.032**	**0.024**	**2.090**	**0.092**	**0.058**	**1.289**	**0.044**	**0.046**	**2.348**
ALSTM [6]	0.017	**0.022**	0.796	0.022	0.014	1.281	0.062	0.028	1.006	0.041	0.028	1.961
+ ReVol	**0.022**	0.021	**0.943**	**0.039**	**0.026**	**2.436**	**0.112**	**0.059**	**1.358**	**0.047**	**0.049**	**2.728**
Vanilla Transformer [17]	0.015	**0.022**	0.768	0.011	0.005	1.110	0.046	0.026	0.804	0.033	0.014	1.306
+ ReVol	**0.020**	0.019	**0.865**	**0.032**	**0.017**	**2.608**	**0.127**	**0.062**	**1.594**	**0.039**	**0.046**	**1.315**
DTML [18]	0.016	**0.017**	0.834	0.013	0.010	1.109	-0.004	-0.004	0.115	0.006	-0.042	0.106
+ ReVol	**0.018**	0.016	**0.886**	**0.048**	**0.043**	**2.454**	**0.095**	**0.046**	**0.879**	**0.035**	**0.044**	**1.674**
MASTER [11]	0.013	0.014	0.753	0.002	0.014	0.542	0.019	0.001	0.345	0.005	0.015	0.359
+ ReVol	**0.015**	**0.015**	**0.794**	**0.035**	**0.019**	**2.248**	**0.102**	**0.050**	**1.213**	**0.038**	**0.042**	**1.688**

Table 5: Performance comparison on Information Coefficient (IC), Rank Information Coefficient (RIC), and Sharpe Ratio (SR) with the state-of-the-art normalization methods. ReVol outperforms other methods, demonstrating better capability to reduce distributional discrepancy in stock price data.

Method	U.S.			China			U.K.			Korea		
	IC	RIC	SR	IC	RIC	SR	IC	RIC	SR	IC	RIC	SR
RevIN [8]	0.010	0.011	0.749	0.030	0.024	2.096	0.013	0.015	0.283	0.023	0.021	1.511
Dish-TS [5]	0.012	**0.020**	0.518	0.020	0.032	1.477	0.015	0.019	0.250	0.015	0.002	1.296
ReVol (proposed)	**0.018**	0.016	**0.886**	**0.048**	**0.043**	**2.454**	**0.095**	**0.046**	**0.879**	**0.035**	**0.044**	**1.674**

Q4. **Noise-aware behavior of attention weights (Section 4.5).** Does the attention mechanism in the RVE module effectively suppress noisy inputs during return-volatility estimation?

Q5. **Ablation study (Section 4.6).** Does each module of ReVol contribute to improving the performance for stock price prediction?

4.1 Experimental Setup

We describe datasets, competitors, and evaluation metrics for experiments.

Datasets. We use stock market datasets collected from the stock markets of the United States, China, United Kingdom, and Korea. The summary of the datasets is presented in Table 2. The goal is to predict the daily return $\frac{S^c_{T+1}}{S^c_T}$ given the price features up to day T. We split the datasets in chronological order into 70% for training, 10% for validation, and 20% for testing.

Data normalization. We generate an input vector $\mathbf{x_t}$ at day t for baselines using widely used price ratio-based normalization method [6, 18]. In contrast, ReVol exploits the normalization technique described in Section 3.2. The normalization method for competitors is shown in Table 3. x^o_t, x^h_t, and x^l_t indicate the relative values of the opening, highest, and lowest prices in comparison to closing price at day t, respectively. x^c_t indicates the relative value of the closing price at day t in comparison to closing price at day $t-1$.

Baselines. ReVol is a model-agnostic method applied to neural network backbones for stock price prediction. We apply ReVol to the following baselines to validate its performance.

- **LSTM** [14] is a recurrent neural network designed to capture long-term dependencies in time series data.
- **GRU** [4] is a simplified variant of LSTM that uses gating mechanisms to effectively capture sequential dependencies in time series data with fewer parameters.
- **ALSTM** [6] enhances the LSTM model by incorporating an attention mechanism, allowing it to focus on the most relevant time steps for improved stock price prediction.
- **Vanilla Transformer** [17] is a self-attention-based architecture originally proposed for machine translation, which can model long-range dependencies in sequences without relying on recurrence.
- **DTML** [18] leverages a transformer structure to capture temporal and inter-stock correlations with global market context for accurate prediction.
- **MASTER** [11] is a transformer-based approach for stock price forecasting that effectively captures momentary and cross-time stock correlations through two-step self-attentions in temporal and stock dimensions, respectively.

Evaluation metrics. We conduct 10 experiments with different random seeds ranging from 0 to 9, and report the average across all cases. We evaluate the results of stock price prediction using Information Coefficient (IC) and Rank Information Coefficient (RIC). IC

and RIC are the dominant metrics in finance, which measure the linear relationships between the predicted values and the actual values with Pearson correlation coefficient and Spearman correlation coefficient, respectively.

We evaluate the portfolio's performance using annualized Sharpe Ratio (SR). The portfolio is rebalanced at the close of each day to allocate a weight of 0.2 to the top 5 stocks with the highest predicted returns.

Hyperparameters. We optimize the network parameters using the Adam [9] optimizer and apply early stopping based on the IC of the validation set. We conduct a hyperparameter search on learning rate, weight decay, the number of attention heads (for transformer-based models), window size, hidden layer size, and guidance loss weight in $\{0.00001, 0.0001, 0.001\}$, $\{0, 0.0001, 0.001, 0.01\}$, $\{1, 2, 4, 8\}$, $\{8, 16, 24, 32, 40\}$, $\{128, 256, 512\}$, and $\{0, 0.25, 0.5, 1\}$, respectively.

4.2 Performance (Q1)

Table 4 presents the performance comparison of ReVol and the baseline models for stock price prediction across four markets (U.S., China, U.K., and Korea). ReVol consistently improves performance across all backbone models and metrics, demonstrating its effectiveness in addressing distribution shifts. On average, ReVol improves IC by over 0.03 and SR by more than 0.7 compared to the baselines, confirming its ability to enhance predictive accuracy and profitability.

Notably, ReVol improves model stability and generalization across diverse market conditions, achieving consistent gains across multiple backbones. This robustness to distributional variations makes ReVol a practical and scalable solution for real-world financial forecasting tasks.

4.3 Comparison with Other Normalization Methods (Q2)

Table 5 compares the performance of ReVol with two state-of-the-art normalization methods, RevIN [8] and Dish-TS [5]. RevIN, Dish-TS and ReVol utilize DTML as the backbone model.

ReVol shows consistent improvement in IC and SR across all markets, outperforming the other normalization methods. As illustrated in Figure 1, ReVol accurately aligns the distribution shapes between the training and test sets, unlike other methods, which only match distribution parameters. This precise alignment reduces distributional discrepancies that degrade predictive accuracy. Additionally, unlike other methods, which normalize features independently and lose inter-feature correlations, ReVol preserves these relationships, enabling more accurate and stable predictions.

4.4 Hyperparameter Sensitivity (Q3)

Figure 4 compares the hyperparameter sensitivity of ReVol with DTML as the backbone to standalone DTML across four key hyperparameters: (a) hidden layer size, (b) window size, (c) number of attention heads, and (d) weight decay. DTML shows significant sensitivity to these hyperparameters, resulting in unstable performance and fluctuations in IC, especially for window size and weight decay settings.

In contrast, ReVol consistently maintains high IC values across all hyperparameter settings. This stability is attributed to ReVol

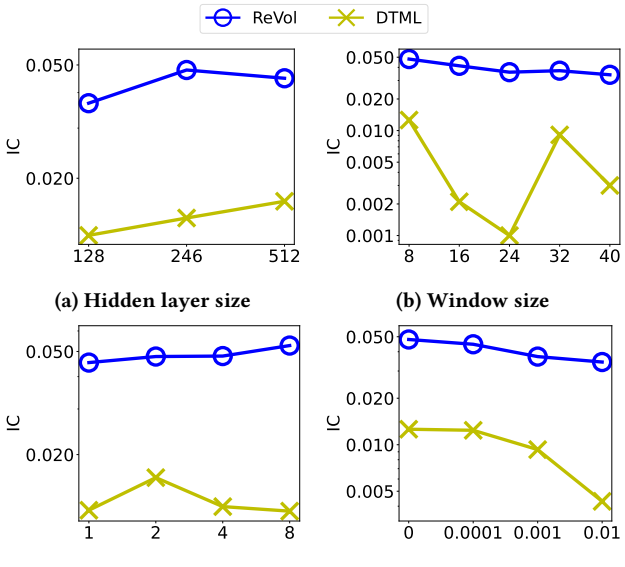

Figure 4: Comparison of hyperparameter sensitivity between ReVol and DTML.

Table 6: Empirical evidence that the attention mechanism down-weights noisy observations: (left) Pearson correlation coefficient between $\left|\frac{S_t^c}{S_{t-1}^c}\right|$ and attention weight α_t, (right) ratio $\frac{\mathbb{E}\left[\alpha_t \,\middle|\, \left|\frac{S_t^c}{S_{t-1}^c}\right| \geq 0.1\right]}{\mathbb{E}\left[\alpha_t \,\middle|\, \left|\frac{S_t^c}{S_{t-1}^c}\right| < 0.1\right]}$ of attention weights for noisy inputs to normal inputs.

Market	Correlation	Attention Ratio
U.S.	−0.179	0.788
China	−0.498	0.940
U.K.	−0.056	0.966
Korea	−0.424	0.483

's ability to mitigate distribution shifts, reducing overfitting and improving generalization. By aligning distributions across samples, ReVol decreases the burden on DTML, making the model more robust and less dependent on precise hyperparameter tuning. Consequently, ReVol achieves more reliable and consistent performance under various training conditions.

4.5 Noise-Aware Behavior of Attention Weights (Q4)

Table 6 shows that our model consistently assigns lower attention weights to time steps with high return magnitudes, indicating its overall tendency to suppress noisy inputs. These results suggest that our attention mechanism effectively identifies and down-weights noisy observations that degrade the estimation of return and volatility.

We quantify this behavior using two complementary analyses. First, the Pearson correlation coefficient between the absolute return and attention weight is negative across all markets, showing that higher-volatility inputs receive lower attention weights. Second, the attention ratio between noisy and less noisy time steps

Table 7: An ablation study of ReVol on China dataset.

Method	IC	RIC	SR
ReVol-N	0.002	-0.011	0.158
ReVol-E	0.045	0.040	0.595
ReVol-D	0.035	0.020	0.168
ReVol	**0.048**	**0.043**	**2.454**

is consistently below 1, confirming that noisy observations are down-weighted.

This behavior is attributed to the optimization dynamics of the model. During training, the attention weights are adjusted to minimize the final loss. If noisy time steps receive high attention weights, the estimated return and volatility become less accurate, leading to larger prediction errors. Through backpropagation, the model receives gradients that penalize high attention weights assigned to noisy inputs, since these inputs increase prediction error. Hence, the attention mechanism implicitly filters out noise by learning to trust stable time steps more.

4.6 Ablation Study (Q5)

We analyze the performance of ReVol with DTML as the backbone, and its variants by removing each of the three primary modules, as shown in Table 7:

- ReVol-N: ReVol without the return-volatility normalization module
- ReVol-E: ReVol without the return-volatility estimator module
- ReVol-D: ReVol without the return-volatility denormalization module

Each module contributes to improved prediction performance, with ReVol incorporating the three modules achieving the highest results. The return-volatility normalization module contributes the most to the overall performance as it directly addresses the fundamental issue of distribution shift in stock price data. By normalizing the input data using return, volatility, and price scale, it effectively reduces distribution discrepancies across samples, enabling the model to focus on learning standardized patterns from historical error terms. This process stabilizes the input distribution, facilitating more robust and efficient feature learning.

5 Conclusion

We propose ReVol, an accurate method for alleviating distribution shift in stock price data. ReVol addresses distribution shift in stock price data to accurately predict future prices. The main ideas of ReVol are: 1) normalizing price features to address distribution shifts by eliminating individual sample characteristics such as return, volatility, and price scale, 2) precisely estimating these characteristics through an attention-based module to minimize the influence of market anomalies, and 3) reintegrating the sample characteristics into the prediction process to restore lost information. Additionally, ReVol leverages geometric Brownian motion to capture long-term trends and neural networks for short-term patterns, effectively integrating their strengths for stock price modeling. ReVol boosts the performance of state-of-the-art backbone models in most scenarios, demonstrating notable improvements in predictive accuracy and stability through the proposed normalization approach.

References

[1] Torben G Andersen and Tim Bollerslev. 1998. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International economic review* (1998), 885–905.

[2] Torben G Andersen, Tim Bollerslev, Francis X Diebold, and Heiko Ebens. 2001. The distribution of realized stock return volatility. *Journal of financial economics* 61, 1 (2001), 43–76.

[3] Tomas Bjork. 2019. Arbitrage theory in continuous time. (2019).

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. 2023. Dish-ts: a general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 7522–7529.

[6] Fuli Feng, Huimin Chen, Xiangnan He, Jie Ding, Maosong Sun, and Tat-Seng Chua. 2019. Enhancing Stock Movement Prediction with Adversarial Training.. In *IJCAI*, Vol. 19. 5843–5849.

[7] Kurt Jacobs. 2010. *Stochastic processes for physicists: understanding noisy systems*. Cambridge University Press.

[8] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*. https://openreview.net/forum?id=cGDAkQo1C0p

[9] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Hao Li, Yanyan Shen, and Yanmin Zhu. 2018. Stock price prediction using attention-based multi-input LSTM. In *Asian conference on machine learning*. PMLR, 454–469.

[11] Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. 2024. MASTER: Market-Guided Stock Transformer for Stock Price Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 162–170.

[12] Wei Li, Ruihan Bao, Keiko Harimoto, Deli Chen, Jingjing Xu, and Qi Su. 2021. Modeling the stock relation with graph network for overnight stock movement prediction. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*. 4541–4547.

[13] Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. 2021. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1017–1026.

[14] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. 2017. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*. Ieee, 1419–1426.

[15] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 2627–2633.

[16] Md Arif Istiake Sunny, Mirza Mohd Shahriar Maswood, and Abdullah G Alharbi. 2020. Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In *2020 2nd novel intelligent and leading emerging sciences conference (NILES)*. IEEE, 87–92.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[18] Jaemin Yoo, Yejun Soun, Yong-chan Park, and U Kang. 2021. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2037–2045.

[19] Donglin Zhan, Yusheng Dai, Yiwei Dong, Jinghai He, Zhenyi Wang, and James Anderson. 2024. Meta-adaptive stock movement prediction with two-stage representation learning. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 508–516.

[20] Lifan Zhao, Shuming Kong, and Yanyan Shen. 2023. Doubleadapt: A meta-learning approach to incremental learning for stock trend forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3492–3503.

Appendix

A Proofs

A.1 Proof of Lemma 2.1

Proof. By Equation (1), the stock price S_{t-1} at time $t-1$ is as follows:

$$S_{t-1} = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)(t-1) + \sigma B_{t-1}}. \tag{20}$$

By dividing Equation (1) by Equation (20), we obtain the following:

$$\frac{S_t}{S_{t-1}} = e^{\left(\mu - \frac{1}{2}\sigma^2\right) + \sigma(B_t - B_{t-1})}. \tag{21}$$

Let $\epsilon_t = B_t - B_{t-1}$. Then, ϵ_t and ϵ_s are independent for $s < t$ and $s \in \mathbb{N}$, and $\epsilon_t \sim \mathcal{N}(0,1)$ due to properties 2 and 3 of BM, respectively. By replacing $B_t - B_{t-1}$ with ϵ_t in Equation (21), we obtain the result. □

A.2 Proof of Lemma 3.1

Proof. By Equation (2), the closing price S_t on day t is as follows:

$$S_t^c = S_{t-1}^c e^{\left(\mu - \frac{1}{2}\sigma^2\right) + \sigma\epsilon_t^c}. \tag{22}$$

Note that $\epsilon_t^c \overset{\text{sample}}{\sim} \mathcal{N}(0,1)$ by Lemma 2.1. We rearrange Equation (22) as follows:

$$\epsilon_t^c = \frac{1}{\sigma}\left(\log\frac{S_t^c}{S_{t-1}^c} - \left(\mu - \frac{1}{2}\sigma^2\right)\right). \tag{23}$$

We get the result by replacing $\mu - \frac{1}{2}\sigma^2$ and σ with their estimations. □

A.3 Proof of Lemma 3.2

Proof. According to the geometric modeling of stock prices, we express the opening price on day t as follows:

$$S_{t-1+r} = S_{t-1}\left(\frac{S_t}{S_{t-1}}\right)^r. \tag{24}$$

By taking the log of Equation (24), we rewrite it as follows:

$$\log\frac{S_{t-1+r}}{S_{t-1}} = r\log\frac{S_t}{S_{t-1}}. \tag{25}$$

To find the best r that satisfies the equation above, we minimize the following mean squared error:

$$\mathbb{E}\left[\left(r\log\frac{S_t}{S_{t-1}} - \log\frac{S_{t-1+r}}{S_{t-1}}\right)^2\right]. \tag{26}$$

Equation (26) attains its minimum at the point where the derivative with respect to r is equal to zero since it is convex with respect to r:

$$0 = \frac{d}{dr}\mathbb{E}\left[\left(r\log\frac{S_t}{S_{t-1}} - \log\frac{S_{t-1+r}}{S_{t-1}}\right)^2\right] \tag{27}$$

$$= \mathbb{E}\left[2\log\frac{S_t}{S_{t-1}}\left(r\log\frac{S_t}{S_{t-1}} - \log\frac{S_{t-1+r}}{S_{t-1}}\right)\right] \tag{28}$$

$$= \mathbb{E}\left[2\log\frac{S_t^c}{S_{t-1}^c}\left(r\log\frac{S_t^c}{S_{t-1}^c} - \log\frac{S_t^o}{S_{t-1}^c}\right)\right] \tag{29}$$

Solving Equation (29), we obtain the result. □

A.4 Proof of Lemma 3.3

Proof. Rewriting Equation (7), we obtain the following:

$$\epsilon_t^o = \frac{1}{\sigma}\left(\log\frac{S_{t-1+r}}{S_{t-1}} - \left(\mu - \frac{1}{2}\sigma^2\right)r\right). \tag{30}$$

Since $\epsilon_t^o \overset{\text{sample}}{\sim} \mathcal{N}(0,r)$, $\frac{1}{\sqrt{r}}\epsilon_t^o \overset{\text{sample}}{\sim} \mathcal{N}(0,1)$. We obtain the result by replacing $\mu - \frac{1}{2}\sigma^2$, σ, and r with their estimations. □

A.5 Proof of Lemma 3.4

Proof. Since S_t^h is the highest price on day t,

$$S_t^h = \max_{0<u\le 1} S_{t-1}e^{\left(\mu - \frac{1}{2}\sigma^2\right)u + \sigma B_u}. \tag{31}$$

It is common to simplify the model by ignoring the drift term $\left(\mu - \frac{1}{2}\sigma^2\right)u$, as its contribution is negligible compared to the volatility term σB_u on a daily scale [1, 2]. To empirically justify this simplification, we compute the average daily ratio $\frac{|\mu - \frac{1}{2}\sigma^2|}{\sigma}$ of the drift term to the volatility term across different markets: 0.066 (U.S.), 0.054 (China), 0.063 (U.K.), and 0.069 (Korea). These small values indicate that the drift's effect is minor in intraday settings.

Thus, for deriving a closed form solution, we ignore the drift term without significantly affecting the results:

$$S_t^h = \max_{0<u\le 1} S_{t-1}e^{\sigma B_u}. \tag{32}$$

Rearranging Equation (32),

$$\frac{1}{\sigma}\log\frac{S_t^h}{S_{t-1}} = \max_{0<u\le 1} B_u. \tag{33}$$

By reflection principle [7],

$$\mathbb{P}\left(\frac{1}{\sigma}\log\frac{S_t^h}{S_{t-1}} \ge a\right) = \mathbb{P}\left(\max_{0<u\le 1} B_u \ge a\right) = 2\mathbb{P}\left(B_1 \ge a\right) \tag{34}$$

for $a > 0$. Since $B_1 \sim \mathcal{N}(0,1)$, $\frac{1}{\sigma}\log\frac{S_t^h}{S_{t-1}}$ is a sample drawn from a half-normal distribution. We obtain the result by replacing σ with its estimation. □