Trade When Opportunity Comes: Price Movement Forecasting via Locality-Aware Attention and Iterative Refinement Labeling

Liang Zeng[1*] Lei Wang[1*] Hui Niu[1] Jian Li[1 †]
Ruchen Zhang[2] Zhonghao Dai [2] Dewei Zhu [2] Ling Wang [2]
[1]Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, China
[2]Huatai Securities Co., Ltd, China
{zengl18,wanglei20,niuh17}@mails.tsinghua.edu.cn,lijian83@mail.tsinghua.edu.cn
{zhangruchen,daizhonghao,zhudewei,wangling}@htsc.com

arXiv:2107.11972v2 [cs.LG] 13 Jun 2022

ABSTRACT

Price movement forecasting aims at predicting the future trends of financial assets based on the current market conditions and other relevant information. Recently, machine learning (ML) methods have become increasingly popular and achieved promising results for price movement forecasting in both academia and industry. Most existing ML solutions formulate the forecasting problem as a classification (to predict the direction) or a regression (to predict the return) problem over the entire set of training data. However, due to the extremely low signal-to-noise ratio and stochastic nature of financial data, good trading opportunities are extremely scarce. As a result, without careful selection of potentially profitable samples, such ML methods are prone to capture the patterns of noises instead of real signals. To address this issue, we propose a novel price movement forecasting framework, called Locality-Aware Attention and Iterative Refinement Labeling (LARA), which consists of two main components: 1) *Locality-aware attention* automatically extracts the potentially profitable samples by attending to surrounding class-aware label information. Moreover, equipped with metric learning techniques, *locality-aware attention* enjoys task-specific distance metrics and distributes attention on potentially profitable samples in a more effective way. 2) *Iterative refinement labeling* further iteratively refines the labels of noisy samples and then combines the learned predictors to be robust to the unseen and noisy samples. In a number of experiments on three real-world financial markets: ETFs, stocks, and cryptocurrencies, LARA achieves superior performance compared with the traditional time-series analysis methods and a set of machine learning based competitors on the Qlib platform. Extensive ablation studies and experiments also demonstrate that LARA indeed captures more reliable trading opportunities.

KEYWORDS

Price movement forecasting, Locality-aware attention, Iterative refinement labeling

1 INTRODUCTION

Price movement forecasting is one of the most important and challenging problems in quantitative finance. It is notoriously difficult partly because the financial market is highly stochastic, dynamic, and volatile [10]. The classical Efficient Market Hypothesis (EMH) (*e.g.,* [12, 23]) states that in an informationally efficient market, price changes must be unforecastable if all relevant information is reflected immediately. However, perfect efficiency is impossible to achieve in practice [5] due to informational asymmetry and "noisy" behaviors among the traders [4]. Indeed, hundreds of abnormal returns have been discovered in the literature (see the classical review by Grossman and Stiglitz [14] and the recent list [16]). As the more recent endeavor to "beat the market" and achieve excess returns, machine learning based solutions, taking advantage of the nonlinear expressive power, have become increasingly popular and achieved promising results (*e.g.,* [10, 15, 33, 39, 40]).

The most straightforward way to employ machine learning techniques for financial forecasting problems consists of the following two steps [15]. First, mining effective factors (either manually or automatically, *e.g.,* [20]) correlated/predictive with asset returns, including fundamental factors, technical factors, economic factors, and alternative factors. Second, feeding these factors as features into some off-the-shelf machine learning algorithms to generate trading signals. Recently, much effort has been devoted to designing new machine learning algorithms in the second step to make more accurate predictions (*e.g.,* [33, 37–40]). Most work mentioned above trains the model over the entire set of training data (typically spans a consecutive period of time)[1]. However, it is well known that financial time-series data has an extremely low signal-to-noise ratio [10], to the degree that modern machine algorithms are prone to pick up patterns of noises instead of real profitable signals (*a.k.a.,* overfit the training set). Without careful selection of potentially profitable samples from the entire set of training data, tuning the models to generalize well can be very challenging [40]. Even for most trading experts, asset prices behave like (almost) unpredictable random walks most of the time, and good trading opportunities only happen occasionally. In light of this fact, we argue that it can be more effective and robust to concentrate on specific samples that are potentially more predictable and profitable, as the proverb says, *trade when opportunity comes.*

Motivation analogy. We explain the motivation of our proposed framework with an intuitive analogy of evaluating students' academic performance. According to the history of students' performance, we regard the excellent students as positive samples and the ordinary students as negative samples, as shown in Fig. 1. Excellent students typically get higher scores than ordinary ones in a single exam, just like the potentially profitable samples in the context of financial datasets. It inspires us to first extract the potentially profitable samples (excellent students) and then construct a more

*Equal contribution.
†Corresponding author.

[1]Note that in financial time-series forecasting problems, one should not split the training and testing set randomly, since the data points are not i.i.d., but strongly and temporally correlated.

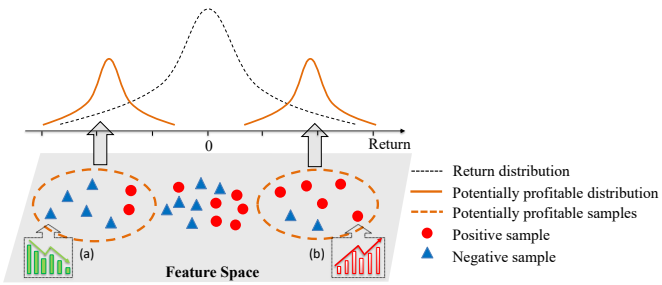


Figure 1: An example to illustrate our intuition. The half top figure represents the probability density function (PDF) of return over corresponding samples (best viewed in color).

accurate classifier on these selected samples (*i.e.*, Fig. 1(a)-(b)). If only one historical exam is observed, it is too hard for us to make a comprehensive evaluation so that we may miss some excellent students and choose the ordinary ones by mistake. Moreover, the ordinary students (red/blue *noisy samples* located in Fig. 1(a)/Fig. 1(b)) mixed into the group of excellent students can be easily distinguished via multiple exams. Thus, it inspires us to further denoise the labels of *noisy samples* with multiple iterative evaluations.

Motivated by this observation, it is desirable to train a powerful and generalizable model for price movement forecasting over extremely noisy market data. However, it also brings two unique technical challenges, which we will tackle in this paper.

Challenge 1: How to extract potentially profitable samples?

Solution 1: We propose the *locality-aware attention* method, which has two main modules: a *metric learning module* for constructing a better metric, and a *localization module* for explicitly extracting potentially profitable samples. Specifically, many algorithms rely critically on being given a good metric over their inputs due to the complex internal relations among different features of financial datasets [36]. It motivates us to introduce the *metric learning module*, which can learn a more compatible distance metric [11, 36] assisting the *localization module*. The *localization module* intends to model an auxiliary relation between the inputs and the labels of historical data. We introduce the *masked attention* scheme to realize this idea by locally integrating label information to extract a better subset of potentially profitable samples.

Challenge 2: How to adaptively distinguish the noisy samples and then denoise their labels?

Solution 2: We observe that in the actual financial market, two samples may have similar features but yield completely opposite labels because of stochastic and chaotic nature of financial datasets [10]. Such *noisy samples*, *i.e.,* similar samples with the opposite labels, preclude the inference of the model [19]. Thus, we propose a novel method, called *iterative refinement labeling*, to adaptively distinguish them according to their supervised losses and reset their labels to the opposite ones to boost the performance of predictors (Fig. 2(d)). This method iteratively refines the labels of noisy samples, and then combines the learned predictors to be robust to the unseen and noisy financial samples.

In short, the main **contributions** can be summarized as follows:

- We propose the hierarchical *locality-aware attention* method to first locally extract potentially profitable samples by attending to label information and then construct a more accurate classifier on these selected samples. Moreover, we utilize a *metric learning module* to learn a well-suited distance metric.
- We propose *iterative refinement labeling* to adaptively reset the labels of noisy samples. Furthermore, we integrate the multiple learned models to boost the performance of predictors.
- We conduct comprehensive experiments on three real-world financial markets. LARA significantly outperforms the traditional time-series analysis methods and a set of machine learning competitors on the Qlib platform [38]. Besides, extensive ablation studies and experiments demonstrate the effectiveness of each component of LARA and suggest that LARA indeed captures more reliable trading opportunities.

2 PRELIMINARY

Problem Setup. Consider a time series of asset trading prices $\{\text{Price}_t | t = 1, 2, \ldots, T\}$ over T time-steps. The price movement trend is usually defined as the future change (return) of the asset price [10]. In this paper, we use a fixed-time horizon method to define the asset price trend. Concretely, the label y_t indicates whether the return of the asset price over a time horizon Δ exceeds a certain pre-defined threshold λ. Note that we build individual models for long and short positions, respectively. For long positions:

$$y_t = \begin{cases} 1, & \text{Price}_{t+\Delta}/\text{Price}_t - 1 > \lambda \\ 0, & \text{otherwise} \end{cases} . \tag{1}$$

Similarly, for short positions, we set the label 1 if $\text{Price}_{t+\Delta}/\text{Price}_t - 1 < -\lambda$ and 0 otherwise. At the training step t, we calculate the feature vector $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$ based on price information up to time t and its corresponding label $y_t \in \mathcal{Y} = \{0, 1\}$. The training set is denoted as $\boldsymbol{X} = [\boldsymbol{x}_1; \cdots; \boldsymbol{x}_N] \in \mathbb{R}^{N \times d}$ associated with its label $\boldsymbol{y} = [y_1; \cdots; y_N] \in \mathbb{R}^N$, where N is the number of time steps and d is the feature dimension. Similarly, $\boldsymbol{X}_{\text{test}} = [\boldsymbol{x}_1; \cdots; \boldsymbol{x}_{N_{\text{test}}}] \in \mathbb{R}^{N_{\text{test}} \times d}$ denotes the testing data.

Goal. *Price movement forecasting* aims to build a binary parameterized classifier $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, where $f_\theta(\boldsymbol{x}_t) = Pr(y_t = 1)$ represents the probability of whether the price trend y_t exceeds the pre-defined threshold λ. Note that $f_\theta(\boldsymbol{x}_t)$ makes a prediction on the asset price trend y_t of Δ-step ahead $(t + \Delta)$ based on price information up to t.

Samples with high $p_{\boldsymbol{x}}$. Because we consider the binary classification setting, we assume the predicted label $y_t \sim \text{Bernoulli}(p_{\boldsymbol{x}_t})$ conditional on known information $\boldsymbol{x}_t$ at the t time-step. It means that we observe the positive class with the probability $p_{\boldsymbol{x}_t}$ based on the current market condition and relevant information. A larger $p_{\boldsymbol{x}_t}$ indicates a stronger tendency and lower uncertainty of asset price, and thus the prediction at this time-step can be more reliable. In order to make more accurate prediction, we gather the *samples with high* $p_{\boldsymbol{x}_t}$, where $p_{\boldsymbol{x}_t}$ is greater than a given threshold ($thres$), *i.e.,* $p_{\boldsymbol{x}_t} > thres$. However, it is impossible to obtain the exact underlying probability $p_{\boldsymbol{x}_t}$ from the dataset since we can only observe one sample of the distribution $y_t \sim \text{Bernoulli}(p_{\boldsymbol{x}_t})$ at one certain time-step. Thus, we introduce *locality-aware attention* (Sec. 3.1) to give an approximate estimation $\hat{p}_{\boldsymbol{x}_t}$ of $p_{\boldsymbol{x}_t}$.

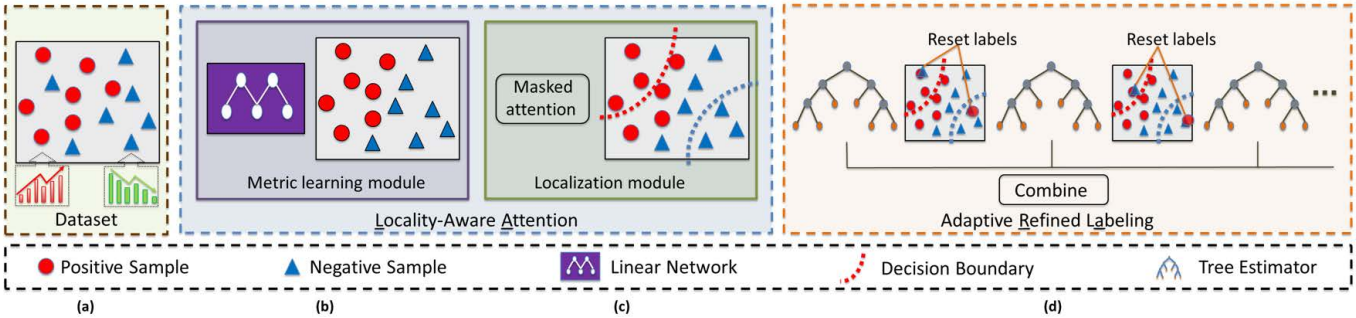


Figure 2: The overview of our proposed LARA framework. (a) Input dataset. (b) Metric learning module. (c) Localization module with masked attention scheme. (d) Iterative refinement labeling method.

3 LARA: THE PROPOSED FRAMEWORK

In this section, we propose our LARA framework, which is composed of the following two main components—**locality-aware attention** (Sec. 3.1) and **iterative refinement labeling** (Sec. 3.2), as shown in Fig. 2. The key idea is to extract profitable samples from the noisy market. *Locality-aware attention* extracts the *samples with high* $p_{\boldsymbol{x}}$ and *iterative refinement labeling* denoises the labels of *noisy samples* to boost the performance of predictors.

3.1 Locality-Aware Attention (LA-Attention)

3.1.1 Localization Module. Recalling that $y_t \sim \text{Bernoulli}(p_{\boldsymbol{x}_t})$, it is hard to recover the Bernoulli distribution from the noisy asset price dataset, as we can only observe one sample of the distribution at the t-th time-step. Suppose the parameter $p_{\boldsymbol{x}_t}$ of the probability distribution is continuous with respect to known information $\boldsymbol{x}_t$ (*e.g.,* the asset price). We design a *localization module* attending to other observed samples in the dataset to obtain an approximate estimation $\hat{p}_{\boldsymbol{x}_t}$ of $p_{\boldsymbol{x}}$. In light of the concepts of attention mechanism [31], $(\boldsymbol{x}_i, y_i)$ denotes the key-value pair, where y_i is the label of $\boldsymbol{x}_i$. Let $\boldsymbol{x}_t$ denote the query sample. The *localization module* can be formulated as follows:

$$\hat{p}_{\boldsymbol{x}_t} = \sum_{1 \le i \le N} y_i \cdot \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_t)}{\sum_{1 \le j \le N} k(\boldsymbol{x}_j, \boldsymbol{x}_t)} \quad , \tag{2}$$

where N is the size of the training set, and $k(\boldsymbol{x}_i, \boldsymbol{x}_t)$ is the attention coefficient between $\boldsymbol{x}_i$ and $\boldsymbol{x}_t$. However, in the most general attention mechanisms, the model allows each sample to attend to every other sample. Intuitively, closer neighbors of a query sample have greater *similarity* than neighbors which are further away. We inject the locality structure into the scheme by performing the *masked attention* scheme and merely pay attention to the neighbors $\mathcal{N}(\boldsymbol{x}_t)$ of $\boldsymbol{x}_t$. Then we reformulate the *localization module* as follows:

$$\hat{p}_{\boldsymbol{x}_t} = \sum_{z_1 \in \mathcal{N}(\boldsymbol{x}_t)} y_{z_1} \cdot \frac{k(z_1, \boldsymbol{x}_t)}{\sum_{z_2 \in \mathcal{N}(\boldsymbol{x}_t)} k(z_2, \boldsymbol{x}_t)} \quad . \tag{3}$$

We propose two easy-to-follow schemes (detailed in Sec. 5.2) to define the neighbors $\mathcal{N}(\boldsymbol{x}_t)$ in the *localization module*: first, the k-nearest neighbors of the query point (K-Neighbor); second, the k-nearest neighbors within a moderate radius (R-Neighbor). Furthermore, there exist several reasonable similarity metrics for the implementation of the attention weight in Eq. 3. We recommend two practical approaches: the first is the identical weight and the second is the reciprocal of distance $d_{\boldsymbol{M}}(z, \boldsymbol{x}_t)$ (as introduced in Sec. 3.1.2) considering the influence of distance, detailed as below:

$$k(z, \boldsymbol{x}_t) := \begin{cases} k_i(z, \boldsymbol{x}_t) = 1 & \text{, the identical weight} \\ k_r(z, \boldsymbol{x}_t) = 1/d_{\boldsymbol{M}}(z, \boldsymbol{x}_t), & \text{the reciprocal of distance} \end{cases} . \tag{4}$$

Specifically, in the training phase, we first extract the *samples with high* $p_{\boldsymbol{x}_t}$ satisfying $\hat{p}_{\boldsymbol{x}_t} > thres$, which implicitly denote the potentially profitable samples. Then we train a more robust model on these selected samples according to the corresponding supervised losses. In the testing phase, we first gather the *samples with high* $p_{\boldsymbol{x}_t}$ by the $\hat{p}_{\boldsymbol{x}_t} > thres$ criterion, and then we only make predictions on these selected samples. Through empirical observations, LA-Attention can largely improve the precision of the prediction model, which constitutes the two-stage training and testing phases, as shown in Algorithm 2.

3.1.2 Metric Learning Module. As introduced in Sec 3.1.1, we need a reasonable measurement of the distance among data samples. Typically, it is a common practice to choose a proper distance metric using prior domain knowledge, *e.g.,* the Euclidean metric. However, in the context of quantitative investment, different investors have different understanding of the financial market and prefer completely different types of factors, *e.g.,* value investors prefer fundamental factors, while technical analysts may choose technical factors. In order to enhance the flexibility of LARA, we turn to the metric learning technique to mitigate the pressure of manually searching for the suitable distance measurement of different factors. The goal of metric learning is to learn a Mahalanobis distance metric $\boldsymbol{M}$ to pull the similar samples close and push the dissimilar samples away. A straightforward solution is to minimize the dissimilarity of samples within the same class. We first define the indicator function as $K_{i,j} = 1$ if $y_i = y_j$ and 0 otherwise. Given two input feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$, we optimize the (squared) Mahalanobis matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ [26] as follows:

$$\begin{aligned} &\min_{\boldsymbol{M}} \quad \frac{1}{2} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} d_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) K_{i,j} \\ &= \min_{\boldsymbol{M}} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} (\boldsymbol{x}_i^T \boldsymbol{M} \boldsymbol{x}_i - \boldsymbol{x}_i^T \boldsymbol{M} \boldsymbol{x}_j) K_{i,j} \quad , \end{aligned} \tag{5}$$

where d is the feature dimension, and $d_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{M}(\boldsymbol{x}_i - \boldsymbol{x}_j)$ denotes the Mahalanobis distance. Metric learning can thus be seen as learning a new embedding space of the feature, which can better leverage the similarity and dissimilarity on samples compared with the Euclidean or Cosine space. Thus, LARA adopts the *metric learning module* to automatically learn a well-suited distance metric in a given feature space assisting the *localization module* to extract potentially profitable samples.

3.2 Iterative Refinement Labeling (RA-Labeling)

Price movement is notoriously difficult to forecast because financial markets are complex dynamic systems [10]. Indeed, we observe that the complicated financial markets sometimes mislead ML systems to produce opposite prediction results with extremely high confidence. For example, some training samples get the high (low) probability of the positive prediction $Pr(y = 1)$, but their labels are negative (positive). Thus, LARA should be robust to these samples that interfere with the prediction of the model [19], called *noisy samples*. We propose to distinguish the *noisy samples* with their training error and reset their labels to the opposite ones, *i.e.,* positive training samples with the extremely low probability of positive predictions are modified to the negative labels, and vice versa. Formally, given the training data $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, our method runs in an iterative manner with adaptively refined labels $\{\boldsymbol{y}^{(i)}\}_{0 \le i \le k}$ and additive predictors $\{f_{\theta_i}\}_{0 \le i \le k}$, where k is the number of rounds in iterative refinement labeling. In the i-th iteration of the training phase, we can leverage the iteratively refined label $\boldsymbol{y}^{(i)}$ to refine the prediction model. We optimize the following objective:

$$\arg\min_{\theta_i} \sum_{j=1}^{N} \mathcal{L}\left(f_{\theta_i}(\boldsymbol{x}_j), y_j^{(i)}\right), \tag{6}$$

where $(\boldsymbol{x}_j, y_j^{(i)})$ denotes the j-th sample associated with its label of the training data $(\boldsymbol{X}, \boldsymbol{y}^{(i)})$, and $\mathcal{L}(\cdot, \cdot)$ is a corresponding loss function, *e.g.*, the cross-entropy loss.

In the i-th iteration of iterative refinement labeling, we distinguish the *noisy samples* with relatively large training loss and modify their labels to the opposite ones:

$$y_j^{(i+1)} := \begin{cases} y_j^{(i)}, & \mathcal{L}\left(f_{\theta_i}(\boldsymbol{x}_j), y_j^{(i)}\right) \le \epsilon_i \\ 1 - y_j^{(i)}, & \mathcal{L}\left(f_{\theta_i}(\boldsymbol{x}_j), y_j^{(i)}\right) > \epsilon_i \end{cases}, \tag{7}$$

where ϵ_i is a hyper-parameter to control the exchange ratio r. The exchange ratio parameter $r \in [0, 1]$ encodes a roughly estimation of the proportion of *noisy samples* appeared in $\boldsymbol{X}$. Furthermore, we set $r = 10\%$ in our experiments because we empirically find that relatively large r would reset the labels of samples to the opposite ones so frequently that deteriorate the performance of the predictor.

Finally, after running k iterations of iterative refinement labeling, we obtain $k + 1$ iterated predictors $\{f_{\theta_0}, \cdots, f_{\theta_k}\}$. We integrate all predictors using the following two different combination schemes: (1) $C = C_{\text{Last}}$: making a prediction from the last iterated predictor. (2) $C = C_{\text{Vote}}$: making a prediction that averages all iterated predictors. Then the output predictor F is defined as follows:

Algorithm 1: RA-Labeling: Iterative Refinement Labeling

Input: Training data $\left(\boldsymbol{X}, \boldsymbol{y}^{(0)}\right) \in \mathbb{R}^{N \times d} \times \mathbb{R}^N$, the initial model $f_{\theta_0} : \mathbb{R}^d \to \mathbb{R}$, iteration number k, exchange ratio r, and combination scheme C.

Output: $F : \mathbb{R}^d \to \mathbb{R}$

Init: $\theta_0 = \arg\min_{\theta_0} \sum_{j=1}^{N} \mathcal{L}\left(f_{\theta_0}(\boldsymbol{x}_j), y_j^{(0)}\right)$

for $i \in \{0, \ldots, k-1\}$ **do**

  for $j \in \{1, \ldots, N\}$ **do**

    $y_j^{(i+1)} = y_j^{(i)} + (1 - 2 \cdot y_j^{(i)}) \cdot \mathbb{1}\left(\mathcal{L}\left(f_{\theta_i}(\boldsymbol{x}_j), y_j^{(i)}\right) > \epsilon_i\right)$

  $\theta_{i+1} = \arg\min_{\theta_{i+1}} \sum_{j=1}^{N} \mathcal{L}\left(f_{\theta_{i+1}}(\boldsymbol{x}_j), y_j^{(i+1)}\right)$

Return: $F = C\left(\{f_{\theta_i} | 0 \le i \le k\}\right)$.

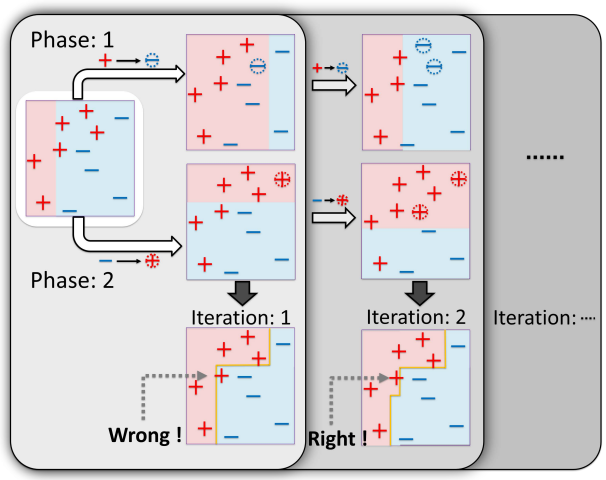


Figure 3: The illustration for the *Iterative Refinement Labeling* method. The plus signs mean the positive samples and the minus signs mean the negative samples. The dotted circle means resetting the label of samples to the opposite one. The decision areas are represented in different colors

$$F(\boldsymbol{x}) := \begin{cases} C_{\text{Last}}\left(\{f_{\theta_i}\}_{0 \le i \le k}\right)(\boldsymbol{x}) = f_{\theta_k}(\boldsymbol{x}), & C = C_{\text{Last}} \\ C_{\text{Vote}}\left(\{f_{\theta_i}\}_{0 \le i \le k}\right)(\boldsymbol{x}) = \frac{1}{k} \sum_{i=0}^{k} f_{\theta_i}(\boldsymbol{x}), & C = C_{\text{Vote}} \end{cases}. \tag{8}$$

An illustration using the vote combining method is shown in Fig. 3. It is shown that, in iteration 1, the weak learners can discriminate the vast majority of samples correctly, except for only one misclassified sample. In iteration 2, the algorithm further classifies all samples correctly after adaptively resetting labels of two *noisy samples* and combining two more predictors. Finally, we combine all the trained models so far to boost the performance of predictors. Note that RA-Labeling can be seamlessly incorporated into existing machine learning methods and the pseudocode is shown in Algorithm 1.

Algorithm 2: LARA framework

Input: Training data $(\boldsymbol{X}, \boldsymbol{y}) \in \mathbb{R}^{N \times d} \times \mathbb{R}^{N}$, testing data $\boldsymbol{X}_{\text{test}} \in \mathbb{R}^{N_{\text{test}} \times d}$, the initial model f_{θ_0}, threshold *thres*, combination scheme C, hyper-parameters k, r.

Output: $Y_{out} \in \mathbb{R}^{N}$.

▷ **Training phase:**

Init *emtpy set* $\boldsymbol{X}', \boldsymbol{y}'$.

for $(\boldsymbol{x}, y)$ ***in*** $(\boldsymbol{X}, \boldsymbol{y})$ **do**

  Estimate $\hat{p}_{\boldsymbol{x}}$ via Eq. 3 with the query sample $\boldsymbol{x}$ and the key-value pair $(\boldsymbol{x}_i, y_i) \in (\boldsymbol{X}, \boldsymbol{y})$.

  if $\hat{p}_{\boldsymbol{x}} > thres$ **then** $\boldsymbol{X}' \cup \{\boldsymbol{x}\}$; $\boldsymbol{y}' \cup \{y\}$.

F = RA-Labeling $\left(\boldsymbol{X}', \boldsymbol{y}', f_{\theta_0}, k, r, C\right)$.

▷ **Testing phase:**

Init *emtpy set* $\boldsymbol{X}'_{\text{test}}$.

for $\boldsymbol{x}_{test}$ ***in*** $\boldsymbol{X}_{test}$ **do**

  Estimate $\hat{p}_{\boldsymbol{x}_{\text{test}}}$ via Eq. 3 with the query sample $\boldsymbol{x}_{\text{test}}$ and the key-value pair $(\boldsymbol{x}_i, y_i) \in (\boldsymbol{X}, \boldsymbol{y})$.

  if $\hat{p}_{\boldsymbol{x}} > thres$ **then** $\boldsymbol{X}'_{\text{test}} \cup \{\boldsymbol{x}_{\text{test}}\}$.

Return: $Y_{out} = F(\boldsymbol{X}'_{\text{test}})$.

4 MORE DISCUSSIONS ON LARA

In this section, we use a running example to illustrate why LA-Attention help extract potentially profitable samples, followed by the discussion of the computational complexity of LARA.

4.1 Locality-Aware Attention

We use a visual and intuitive example to further understand the effectiveness of the *metric learning module* incorporated into the *locality-aware attention* method, as shown in Fig. 4. We plot the feature embedding of randomly sampled 5000 points in the ETF market dataset using t-SNE [29]. We aim to search for more potentially profitable *samples with high* $p_{\boldsymbol{x}_t}$ via the $\hat{p}_{\boldsymbol{x}_t} > thres$ criterion (*thres* equals 0.5 in this demo). In Fig. 4(a), the positive and negative points are mixed together and there are 174 positive samples in total, which have the return distribution with almost the zero mean and are far away from the desired high $p_{\boldsymbol{x}_t}$. After introducing the *localization module* (Fig. 4(b)), more positive samples (279) are found and the mean return of potentially profitable samples is obviously larger than zero, which indicates the effectiveness of LA-Attention. However, the negative ones still account for a large part. In Fig. 4(c), further combined with the *metric learning module*, the number of the positive samples (424) accounts for almost half of all samples and the learned embedding has a more compact structure and clearer distinct boundaries. It achieves the $8.42E$-4 mean return, which is larger than zero with a greater margin compared with (a) and (b). Hence, this learned distance metric can assist the *locality-aware attention* method to distinguish *samples with high* $p_{\boldsymbol{x}_t}$.

4.2 Discussion of Computational Complexity

There are N samples in the training phase and N_{test} samples in the testing phase. Each sample has d-dimensional features. Metric learning (SDML) intends to build a $d \times d$ sparse Mahalanobis distance matrix. Thanks to the low dimension in our task (only 40), the optimization speed is fast. For LA-Attention, the main time consumption

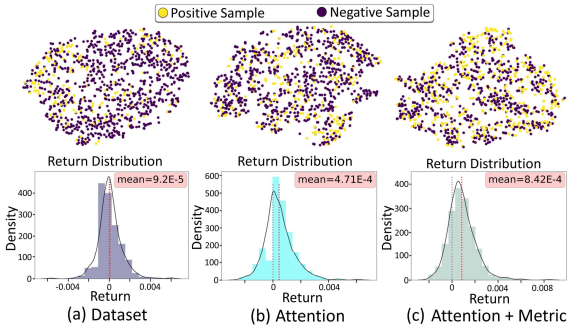


Figure 4: The 2-dimensional visualization with t-SNE and its corresponding return distribution for (a) the input dataset (b) *samples with high* $p_{\boldsymbol{x}_t}$ selected by the *localization module* only (c) *samples with high* $p_{\boldsymbol{x}_t}$ selected by LA-Attention (*localization module+metric learning module*).

is to search for the neighbors of the corresponding sample and we utilize a fast approximate nearest neighbors search method called HNSW [24]. Note that HNSW takes $O(\log N)$ to search for the neighbors of each sample. Hence, it takes $O((2N + N_{\text{test}}) \log N)$ in LA-Attention in total when considering $(N + N_{\text{test}})$ samples to search for neighbors and $O(N \log N)$ for construction of the training data structure. In addition, we define the time for constructing one estimator as T_c in *LightGBM* [18]. Let N_l be the number of estimators in total and T_p be the time for predicting the testing dataset. RA-Labeling repeats the training and testing process for k times, thus, the time cost of the training and the testing phase is $O\left(k\left(N_l T_c + T_p\right)\right)$. Note that searching for neighbors in LA-Attention, constructing the tree estimators, and predicting the results can all be well paralleled, so we can react quickly in the actual financial market.

5 EXPERIMENTS

To demonstrate the effectiveness of our proposed LARA framework, we further conduct extensive experiments on three real-world financial markets: ETFs (an exchange-traded fund market), China's A-share market (a securities exchange), and the cryptocurrency market (OKEx). ETFs are traded in huge volumes in Chinese exchanges and the markets are extremely competitive. We conduct our experiments cooperating with one of China's largest securities brokerages, whose name has been omitted to maintain their privacy and anonymity. We compare our framework LARA on four liquid Chinese ETFs with a set of baseline models: time-series analysis methods, standard machine learning algorithms, and several ablated variants of LARA (Sec. 5.3). To further evaluate LARA in different trading markets with much longer periods and various granularities, we conduct additional extensive experiments with China's A-share market and the cryptocurrency market (Sec. 5.4) on the Qlib platform [38], a popular open-sourced quantitative investment platform equipped with the latest AI technologies. Moreover, LARA achieves superior performance against a set of competitors and captures more reliable trading opportunities. Our codes[2] and project webpage[3] are publicly available.

[2] https://tinyurl.com/LARA-KDD2022

[3] https://sites.google.com/view/lara-fintech

Table 1: Details of the datasets.

Dataset		#Instances	Granularity	Δ	λ	Positive (Ratio)	Negative (Ratio)
ETF	159915.SZ	448,360	3 secs	1 min	1E-3	60,141 (13.41%)	58,344 (13.01%)
	512480.SH	433,231	3 secs	1 min	1E-3	99,907 (23.06%)	102,967 (23.77%)
	512880.SH	448,613	3 secs	1 min	1E-3	97,579 (21.75%)	99,775 (22.24%)
	515050.SH	459,648	3 secs	1 min	1E-3	53,457 (11.63%)	52,399 (11.40%)
Stock	CSI300	859,230	1 day	1 day	1E-3	409,487 (47.66%)	-[6]
Cryptocurrency	BTC/USDT	10,865,879	0.1 sec	10 secs	1E-4	4,265,413 (39.26%)	-[6]

5.1 Experimental Setups

Performance Metrics. We use four commonly used financial metrics [22] to comprehensively evaluate the performance of LARA.

1) *Precision.* In quantitative trading, we care more about samples with the positive prediction, because these samples can lead us to identify real trading signals. Hence, *precision* is an important criterion to evaluate different models, which can be calculated as $TP/(TP + FP)$, where TP = True Positive and FP = False Positive.

2) *Win-Loss Ratio* is a risk indicator, which is calculated by the ratio of the average mean of winnings to the average mean of losses. It measures the profitability for each of the winnings in excess of that for each of the losses, *i.e.*, Return of winning/Return of loss.

3) *Transactions* is an important indicator for high-frequency trading to compare different methods with the same frequency of trades. It is defined as follows: $\#Transactions$ = The number of trades.

4) *Average Return* is a basic and direct index to evaluate the performance of the corresponding strategy in the high-frequency trading scenarios. It represents the average return per transaction or the expected profit of each trade, *i.e.*, Returns/#Transactions.

Evaluation Protocols. Following [39], we adopt a representative trading strategy to evaluate the performance of LARA regarding the actual trading scenarios of the financial market.

• **Trading Signals.** #Transactions is an important criterion to measure the transaction frequency of trading strategies in high-frequency quantitative trading. According to [1, 39], it is a common practice to set the threshold such that the top confident samples are retrieved, and we fairly compare the predictive ability on these selected samples among different methods. Hence, we retrieve the top 100, 200, 500, and 1000 samples with the highest probability, and then we generate the corresponding trading signals.

• **Trading Strategies.** We do not set a position limit for trading the corresponding asset. At time-step t, if the model gives a positive signal, we open a new long position and close this position 1 minute later. Similarly, we open new short positions for the negative signals and close these positions 1 minute later.

Dataset Collection. For ETFs, we conduct experiments over four high-frequency Chinese exchange-traded funds (ETF): 159915.SZ, 512480.SH, 512880.SH, and 515050.SH. All of them are liquid and traded in large volumes in the exchange. We chronologically split the data into three time periods for training (from Jan. 1, 2020 to Apr. 17, 2020), validation (Apr. 20, 2020 to May. 29, 2020), and testing (from Jun. 1 2020 to Jul. 6 2020). We summarize the basic statistics of datasets in Table 1. In our experiment, we collect 40 features based on commonly used alpha factors [2], which are broadly divided into two types (Please refer to our webpage[3] for the complete factor list.). (1) Technical factors, *e.g.*moving average of price and buy/sell pressure indicators [2]. (2) ETF factors, *e.g.*the weighted return among the constituent stocks of the corresponding ETF [1]. Due to space limitations, we only show the experimental results of 512480.SH, which is one of the mostly-used benchmark ETFs. The results for other ETFs can be found in the webpage[3].

For stocks, following Quant Dataset Zoo in Qlib, we use two recommendation datasets: *Alpha158* and *Alpha360*, the constituents of CSI300[4] in China stock market. *158* and *360* represent the different number of pre-calculated alpha factors, which can explain and predict future asset returns. We follow the temporal order to split data in training (from 01/01/2008 to 12/31/2014), validation (from 01/01/2015, 12/31/2016), and testing (from 01/01/2017 to 08/01/2020). Note that the extra splitting gap of return, *i.e.,* (Close[-2]/Close[-1])-1[5], is intentionally introduced to avoid the problem of future information leakage in the daily-frequency dataset.

For cryptocurrencies, we use the high-frequency BTC/USDT trading pair. Each record is one market snapshot captured for approximately every 0.1 second. The training samples used in the experiments come from 7 consecutive trading days (from 01/03/2021 to 07/03/2021), with a total number of 10 million. The validation and testing samples come from the following 7 trading days (from 08/03/2021 to 10/03/2021 and from 11/03/2021 to 14/03/2021). We use the same technical and limit-order book factors to calculate input features as ETFs.

5.2 Implementation Details

We adopt Sparse High-Dimensional Metric Learning (SDML) [26] in the *metric learning module.* We utilize a fast approximate nearest neighbor search method called HNSW [24] to efficiently implement the K-Neighbor/R-Neighbor algorithm in LA-Attention (detailed in Sec. 3.1). The pseudocode for the K-Neighbor/R-Neighbor is detailed in Supplement B. We employ *LightGBM* [18] as the machine learning predictor in the LARA framework since it was reported to be the most commonly-used predictor for financial prediction tasks in several previous work [37, 39]. The combining methods we use in RA-Labeling are *last* and *vote* described in Sec. 3.2. As we study the binary classification problem, we use the (binary) cross-entropy loss in RA-Labeling. For LARA, we optimize the hyper-parameters on the validation set using the grid search method. For reproducibility, we provide the detailed experimental parameters in Supplement A. Moreover, LARA is a model-agnostic and versatile framework, and existing deep learning methods of price movement forecasting can be also seamlessly integrated into LARA.

[4]CSI300 consists of the 300 largest and most liquid A-share stocks, which aims to reflect the overall performance of China A-share market.

[5] $[-x]$ refers to the closing price of the next x consecutive trading days.

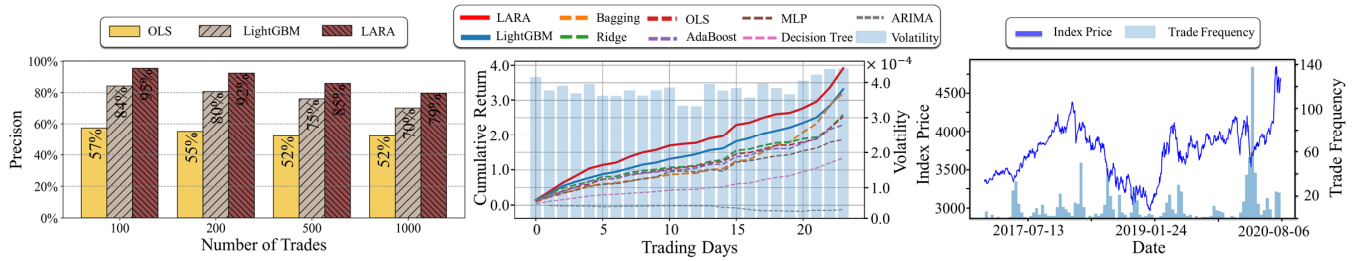


Figure 5: *Left:* Comparisons with the different number of trades among three methods over Precision on 512480.SH. *Middle:* Quantitative comparisons over cumulative return between LARA and baselines on 512480.SH. The right y-axis illustrates the volatility [5] on each trading day. *Right:* The trade distribution in the China's A-share market and the corresponding index price. The left (right) y-axis illustrates the index price (the trade frequency).

Table 2: Quantitative comparisons among different methods on 512480.SH. Best results are in bold.

Methods		Precision(%)	Win-Loss Ratio	Average Return	#Transactions
Time Series	OLS	$52.22_{\pm 0.21}$	$2.430_{\pm 0.017}$	$1.27\text{E-}3_{\pm 4\text{E-}6}$	1000
	ARIMA	$12.57_{\pm 2.77}$	$0.919_{\pm 0.073}$	$-6.00\text{E-}5_{\pm 1\text{E-}4}$	1000
Machine Learning	Ridge	$53.58_{\pm 0.22}$	$2.754_{\pm 0.009}$	$1.29\text{E-}3_{\pm 1\text{E-}5}$	1000
	Decision Trees	$34.48_{\pm 1.44}$	$1.487_{\pm 0.111}$	$6.70\text{E-}3_{\pm 5\text{E-}5}$	1000
	Bagging Regressor	$62.14_{\pm 1.56}$	$2.187_{\pm 0.183}$	$1.61\text{E-}3_{\pm 4\text{E-}5}$	1000
	AdaBoost	$51.32_{\pm 3.23}$	$1.935_{\pm 0.055}$	$1.15\text{E-}3_{\pm 8\text{E-}5}$	1000
	MLP	$43.53_{\pm 17.89}$	$1.838_{\pm 0.293}$	$9.30\text{E-}4_{\pm 5\text{E-}4}$	1000
	LightGBM	$70.16_{\pm 0.89}$	$2.872_{\pm 0.106}$	$1.66\text{E-}3_{\pm 2\text{E-}5}$	1000
LARA	+ LA-Attention	$78.36_{\pm 1.14}$	$\mathbf{2.934_{\pm 0.241}}$	$1.96\text{E-}3_{\pm 4\text{E-}5}$	1000
	+ RA-Labeling	$71.03_{\pm 0.60}$	$2.913_{\pm 0.097}$	$1.68\text{E-}3_{\pm 1\text{E-}5}$	1000
	+ LA-Attention + RA-Labeling	$\mathbf{79.53_{\pm 0.67}}$	$2.933_{\pm 0.221}$	$\mathbf{1.98}\text{E-}3_{\pm 2\text{E-}5}$	1000

For all experiments, we use the default hyper-parameter settings of *LightGBM* and the *metric learning module* to ensure a fair comparison. All experiments are repeated 5 times, and we report the average results with standard deviations to reduce the influence of randomness. In what follows, we always record *the average results of long and short positions.*

5.3 LARA Trades ETFs

A series of experiments on ETFs show the following results: (1) LARA framework outperforms 8 traditional time-series analysis methods and a set of machine learning algorithms (Sec. 5.3.2). (2) LA-Attention significantly improves the performance when applied in both the training and the testing phases (Sec. 5.3.3). (3) LARA is insensitive to the tuning of hyper-parameters (Sec. 5.3.4).

5.3.1 Compared Methods. **(1) Time-series analysis methods [40]:** ordinary least squares (OLS) and autoregressive integrated moving average (ARIMA). **(2) Machine learning methods [15]:** ridge regression, decision trees, bagging regressor, AdaBoost, multi-layer perceptron (MLP), and LightGBM. **(3) LARA Variants:** in order to validate the effectiveness of the two main components, *i.e.,* LA-Attention (Sec. 3.1) and RA-Labeling (Sec. 3.2), we ablate them separately and conduct the following experiments: *i) LA-Attention*; *ii) RA-Labeling*; *iii) LA-Attention + RA-Labeling.*

[6]Short selling is not allowed in China's A-share market without considering margin trading and also in the cryptocurrency market.

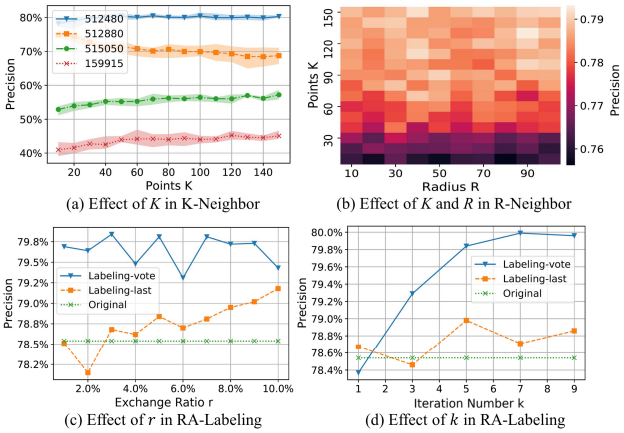
5.3.2 Comparison Experimental Results. First, we quantitatively evaluate LARA compared with traditional time-series analysis and machine learning algorithms to demonstrate the effectiveness of our proposed framework. For a fair comparison, we generate trading signals according to the top 1000 probability values. As shown in Table 2, LARA achieves the highest performance of all three indicators under the same #Transactions, *i.e.*, Precision, Win-Loss Ratio, and Average Return. We improve the performance with a great margin by up to 8% over precision compared with other time-series and machine learning algorithms. Combined with RA-Labeling, the precision further improves by 1-2% and the variance among different seeds significantly decreases, which suggests RA-Labeling can empirically reduce noise of financial datasets.

Second, in order to investigate the effect of #Transactions, we choose three representative methods of each category with the highest precision in Table 2, *i.e.*, OLS in traditional time-series analysis, LightGBM in machine learning, and our proposed LARA framework. As shown in Fig. 5 left, LARA consistently outperforms OLS and LightGBM in terms of precision under the different #Transactions.

Third, to further evaluate our LARA framework's effectiveness, we illustrate the performance of the back-testing strategy in terms of cumulative return (Fig. 5 middle). We choose the top 1000 signals with the highest predicted probability for fair comparison and do not set position limits. For the sake of simplicity, transaction costs are ignored since #Transaction is the same over all methods, which does not affect the experimental conclusion. It is obvious that LARA always outperforms other baseline methods throughout the test

Table 3: Ablation studies with LA-Attention applied in the training and testing phase. Best results are in bold. ✓(-) means we evaluate w (w/o) LA-Attention in the corresponding phase.

Training Phase	Testing Phase	Precision(%)	Win-Loss Ratio	Average Return(%)	#Transactions
-	-	$70.16_{\pm 0.89}$	$2.872_{\pm 0.106}$	$0.166_{\pm 0.002}$	1000
-	✓	$70.39_{\pm 1.07}$	$2.757_{\pm 0.184}$	$0.166_{\pm 0.002}$	1000
✓	-	$37.82_{\pm 11.09}$	$1.712_{\pm 0.188}$	$0.062_{\pm 0.037}$	1000
✓	✓	$\mathbf{78.58_{\pm 0.67}}$	$\mathbf{2.957_{\pm 0.259}}$	$\mathbf{0.196_{\pm 0.003}}$	1000



(a) Effect of K in K-Neighbor (b) Effect of K and R in R-Neighbor (c) Effect of r in RA-Labeling (d) Effect of k in RA-Labeling

Figure 6: Hyperparameter study. (a)(b) Effect of K and R in K-Neighbor and R-Neighbor. (c)(d) Effect of k and r in RA-Labeling. Results shown in (b,c,d) are obtained on 512480.SH.

period, which demonstrates that LARA significantly improves the prediction ability of the asset price trend.

5.3.3 Ablation Experimental Results. We explore whether LA-Attention can boost performance when concentrating on the *samples with high* $p_{\boldsymbol{x}}$. We consider whether to use LA-Attention in the training and testing phases in terms of different performance metrics. The result is shown in Table 3. LA-Attention applied in both the training and testing phases outperforms other ablation competitors on all performance metrics, which is in line with our expectation that LA-Attention can capture much more reliable trading opportunities with a significantly higher average return. If we merely use LA-Attention in the testing phase, the performance decreases slightly by about 8% on precision. However, merely using LA-Attention in the training phase can lead to significant performance degradation (about 40% on precision). We postulate the reason is that the distributions of datasets in the training and testing phases greatly deviate from each other, which suggests that we should pay attention to the *samples with high* $p_{\boldsymbol{x}}$ in both the training and testing phases to capture more reliable trading opportunities.

5.3.4 Hyperparameter Study. **Analysis of LA-Attention.** For two implementations of LA-Attention: K-Neighbor and R-Neighbor (Sec. 3.1), we study the effect of points K and radius R. As shown in Fig. 6(a), when the number of points in the K-Neighbor algorithm increases, the precision index of most ETFs gradually increases, except 512880.SH. The same situation occurs in Fig. 6(b). The larger neighborhood it attends to, the better performance we can obtain. However, it costs more time when attending to more neighbors, which is a trade-off between time and performance. Besides, when we vary radius R from 10 to 100 using a fixed K, R-Neighbor achieves relatively stable performance as shown in Fig. 6(b). In all, these results demonstrate LA-Attention does not rely on the heavy tuning of hyperparameters to achieve satisfactory performance.

Analysis of RA-Labeling. We conduct experiments to analyze the robustness of two combination schemes in RA-Labeling (Sec. 3.2) with the different iteration number k and exchange ratio r. In Fig. 6(c), *labeling-last* (C_{Last}) performs better as r increases. Moreover, precision further improves by adopting *labeling-vote* (C_{Vote}), which suggests the effectiveness of RA-Labeling. A similar phenomenon appears in Fig. 6(d). As k increases, *Labeling-vote* achieves better performance on precision and reaches the saturated performance when k is larger than 7.

5.4 LARA Trades Stocks and Cryptocurrencies

To further qualify LARA in different trading markets with longer periods and various granularities, we transplant our proposed LARA framework to a recently launched Qlib platform [38] to evaluate its performance in China's A-share market and OKEx with a set of deep learning based competitors. Extensive experiments demonstrate the effectiveness of LARA against a set of competitors.

5.4.1 Compared Models. We compare our LARA framework with the following baseline models: Linear, XGBoost [8], LightGBM [18], CatBoost [25], GRU [9], LSTM [17], ALSTM [27], GATs [32], SFM [40], TFT [21], TabNet [3], DoubleEnsemble [39], and TCTS [35]. Due to space limitations, we only list the competitive models in Table 4. More comparisons are detailed on the webpage[3].

5.4.2 Experimental Results. Table 4 shows the results in China's A-share stock market and OKEx. LARA obtains 59.1% precision and outperforms the best performing baseline models by up to 4.1% in the Alpha158 dataset. As for the average return, LARA is slightly better than (in the Alpha158 dataset) and on par with (in the Alpha360 dataset) several deep learning competitors under the 1000 #Transactions with the top predicted probability. Moreover, LARA achieves superior performance over all three metrics in the cryptocurrency market. They all suggest the effectiveness of LARA on the real-world markets executed on the Qlib platform.

In Fig. 5 right, we plot the trade distribution of LARA in China's A-share market and the corresponding index price. We can clearly see that LARA trades more when the index price fluctuates sharply, which also meets our expectation that there will be more opportunities when the market is highly volatile. These results further demonstrate that LARA can be used to guide real-world quantitative investments to pick up reliable trading opportunities.

6 RELATED WORK

Price Movement Forecasting. There are two main research threads in price movement forecasting based on machine learning algorithms. The first focuses on improving prediction by introducing extra financial data sources and performing feature engineering

Table 4: Quantitative comparisons among different methods on the China's A-share market and the cryptocurrency market. - means that it is not implemented in Qlib. Best results are in bold. ·/· denotes the results of Alpha158 / Alpha360.

Methods		Precision(%)		Win-Loss Ratio		Average Return		#Transactions
		Stock	Crypto	Stock	Crypto	Stock	Crypto	
Qlib Platform [38]	XGBoost [8]	11.1 / 53.3	53.3	1.182 / 1.138	0.678	1.04E-3 / 5.22E-3	-3.0E-5	1000
	LightGBM [18]	55.0 / 54.0	51.0	1.331 / 1.235	0.890	7.26E-3 / 5.38E-3	1.4E-5	1000
	GRU [9]	51.3 / 50.7	44.4	1.114 / 1.254	1.013	3.02E-3 / 4.29E-3	3.7E-5	1000
	LSTM [17]	55.0 / 53.2	43.8	1.106 / 1.200	1.053	5.53E-3 / 4.62E-3	1.9E-5	1000
	ALSTM [27]	51.9 / 49.9	47.9	1.133 / 1.110	0.933	3.24E-3 / 2.62E-3	5.9E-5	1000
	TabNet [3]	51.8 / 50.3	51.0	1.299 / 1.146	0.890	5.14E-3 / 2.83E-3	1.4E-5	1000
	TCTS [35]	- / 55.6	-	- / 0.913	-	- / 2.09E-3	-	1000
	Transformer	53.0 / 53.2	38.7	1.064 / 1.230	0.888	3.34E-3 / 5.73E-3	-5.0E-5	1000
	Localformer	52.9 / 52.7	49.3	1.116 / **1.309**	**1.177**	4.64E-3 / 5.73E-3	1.3E-4	1000
	DoubleEnsemble [39]	14.4 / 54.0	-	**1.379** / 1.225	-	2.23E-3 / **5.75E-3**	-	1000
LARA	+ LA-Attention	56.6 / 51.4	51.2	1.142 / 1.031	0.826	5.27E-3 / 1.54E-3	9.5E-5	1000
	+ RA-Labeling	55.2 / 52.8	56.2	1.038 / 1.152	1.034	3.53E-3 / 2.71E-3	1.4E-4	1000
	+ LARA	**59.1** / **56.0**	**57.8**	1.274 / 1.275	1.059	**7.79E-3** / 5.46E-3	**1.5E-4**	1000

techniques [2, 15, 20, 28]. Gu et al. [15] used sentiment analysis to extract extra financial information from multiple textual data sources. Besides, there also exists much literature (*e.g.,* [2, 20]) studying new methods to construct more informative features for financial data which can be effectively applied to deep learning models. The second seeks to construct well-designed prediction models to achieve more accurate price prediction [33, 34, 37, 39, 40]. Zhang et al. [39] proposed a new ensemble method based on sample reweighting and feature selection for financial data analysis. Xu et al. [37] developed an adaptive regularized model for unexpected revenue prediction. Zhang et al. [40] introduced the state frequency memory recurrent networks for stock price prediction. Despite increasing efforts in price movement forecasting, few of them explicitly utilize auxiliary label information to denoise the noisy financial data. Inspired by this, we propose LARA to first extract the potentially profitable samples and then adaptively refine the labels of *noisy samples*.

Handling Label Noise. Learning with noisy labels is inevitable in the realistic scenario [6, 7, 13, 30]. Van Rooyen et al. [30] proposed the so-called unhinged loss to cope with synthetic labels considering noise. In contrast, our proposed RA-Labeling method is designed to adaptively refine the similar samples with opposite labels to boost the performance of predictors in the quantitative trading scenarios.

7 CONCLUSION

We study the problem of price movement forecasting and present LARA (Locality-Aware Attention and Iterative Refinement Labeling). In LARA, we propose LA-Attention to extract potentially profitable samples and RA-Labeling to adaptively denoise labels of noisy samples. Extensive experiments demonstrate its superior performance over traditional time-series analysis and a set of machine learning based competitors on three real-world financial markets: ETFs, stocks and cryptocurrencies. Besides, we also illustrate how each component works by comprehensive ablation studies, which indicates that LARA indeed captures more reliable trading opportunities. We expect our work to pave the way for price movement forecasting in more realistic quantitative trading scenarios.

REFERENCES

[1] David Abner. 2010. The ETF handbook. *How to Value and Trade Exchange Traded Funds John Wiley & Sons Inc., New Jersey* (2010).
[2] Steven B Achelis. 2001. *Technical Analysis from A to Z.* McGraw Hill New York.
[3] Sercan O Arık and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *Proc. of AAAI.*
[4] Robert Bloomfield, Maureen O'hara, and Gideon Saar. 2009. How noise trading affects markets: An experimental analysis. *The Review of Financial Studies* (2009).
[5] John Y Campbell, John J Champbell, John W Campbell, Andrew W Lo, Andrew W Lo, and A Craig MacKinlay. 1997. *The econometrics of financial markets.* princeton University press.
[6] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* (2002).
[7] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Nature/Scientific reports* (2018).
[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proc. of KDD.*
[9] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*
[10] Marcos Lopez De Prado. 2018. *Advances in financial machine learning.* John Wiley & Sons.
[11] William De Vazelhes, CJ Carey, Yuan Tang, Nathalie Vauquier, and Aurélien Bellet. 2020. metric-learn: Metric learning algorithms in python. *Journal of Machine Learning Research* (2020).
[12] Eugene F Fama. 1960. Efficient market hypothesis. *Diss. PhD Thesis, Ph. D. dissertation* (1960).
[13] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* (1997).
[14] Sanford J Grossman and Joseph E Stiglitz. 1980. On the impossibility of informationally efficient markets. *The American economic review* (1980).
[15] Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2018. *Empirical asset pricing via machine learning.* Technical Report.
[16] Campbell R Harvey, Yan Liu, and Heqing Zhu. 2016. … and the cross-section of expected returns. *The Review of Financial Studies* (2016).
[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).
[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Proc. of NeurIPS* (2017).
[19] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the Loss Landscape of Neural Nets. In *Proc. of NeurIPS.*
[20] Zhige Li, Derek Yang, Li Zhao, Jiang Bian, Tao Qin, and Tie-Yan Liu. 2019. Individualized indicator for all: Stock-wise technical indicator optimization with stock embedding. In *Proc. of KDD.*
[21] Bryan Lim, Stefan Zohren, and Stephen Roberts. 2019. Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science* (2019).

[22] Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. 2021. Learning Multiple Stock Trading Patterns with Temporal Routing Adaptor and Optimal Transport. *arXiv preprint arXiv:2106.12950* (2021).

[23] Andrew W Lo. 2019. *Adaptive markets: Financial evolution at the speed of thought*. Princeton University Press.

[24] Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[25] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Proc. of NeurIPS*.

[26] Guo-Jun Qi, Jinhui Tang, Zheng-Jun Zha, Tat-Seng Chua, and Hong-Jiang Zhang. 2009. An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization. In *Proc. of ICML*.

[27] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W Cottrell. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *Proc. of IJCAI*.

[28] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. 2021. Exploring the Scale-Free Nature of Stock Markets: Hyperbolic Graph Learning for Algorithmic Trading. In *Proc. of WWW*.

[29] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* (2008).

[30] Brendan Van Rooyen, Aditya Krishna Menon, and Robert C Williamson. 2015. Learning with symmetric label noise: The importance of being unhinged. *arXiv preprint arXiv:1505.07634* (2015).

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *Proc. of ICLR*.

[33] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. 2019. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proc. of KDD*.

[34] Qingsong Wen, Zhe Zhang, Yan Li, and Liang Sun. 2020. Fast RobustSTL: Efficient and Robust Seasonal-Trend Decomposition for Time Series with Complex Patterns. In *Proc. of KDD*.

[35] Xueqing Wu, Lewen Wang, Yingce Xia, Weiqing Liu, Lijun Wu, Shufang Xie, Tao Qin, and Tie-Yan Liu. 2021. Temporally Correlated Task Scheduling for Sequence Learning. In *Proc. of ICML*.

[36] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. 2002. Distance metric learning with application to clustering with side-information. In *Proc. of NeurIPS*.

[37] Jin Xu, Jingbo Zhou, Yongpo Jia, Jian Li, and Xiong Hui. 2020. An adaptive master-slave regularized model for unexpected revenue prediction enhanced with alternative data. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*.

[38] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. 2020. Qlib: An AI-oriented Quantitative Investment Platform. *arXiv preprint arXiv:2009.11189* (2020).

[39] Chuheng Zhang, Yuanqi Li, Xi Chen, Yifei Jin, Pingzhong Tang, and Jian Li. 2020. DoubleEnsemble: A New Ensemble Method Based on Sample Reweighting and Feature Selection for Financial Data Analysis. In *Proc. of ICDM*.

[40] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *Proc. of KDD*.

SUPPLEMENTAL MATERIAL

A HYPERPARAMETERS SETTINGS

For the reproducibility of our proposed LARA framework, we describe the hyperparameter settings in detail here. In order to compare different methods fairly, we first search hyper-parameters on the validation set, and then select the best hyperparameters based on their performance. We list the detailed values of K in K-Neighbor, K and R in R-Neighbor, k and r in the RA-Labeling method, and the combination scheme C in Table 5.

Table 5: Hyperparameters of each ETF

ETF	K-Neighbor/ R-Neighbor	K	R	k	r	C
Long Positions						
159915.SZ	K-Neighbor	150	-	7	0.05	C_{Vote}
512480.SH	R-Neighbor	90	100	9	0.04	C_{Vote}
512880.SH	R-Neighbor	150	50	7	0.07	C_{Vote}
515050.SH	K-Neighbor	120	-	9	0.05	C_{Vote}
Short Positions						
159915.SZ	K-Neighbor	100	-	9	0.07	C_{Vote}
512480.SH	R-Neighbor	150	30	9	0.03	C_{Vote}
512880.SH	K-Neighbor	150	-	9	0.10	C_{Vote}
515050.SH	K-Neighbor	150	-	7	0.10	C_{Vote}

B IMPLEMENTATIONS OF LA-ATTENTION

As introduced in Sec. 3.1, we consider two kinds of attention weights: the identical weight k_i to implement the K-Neighbor algorithm and the reciprocal of distance k_r to implement the R-Neighbor algorithm. The HNSW algorithm can be directly adopted in K-Neighbor. As for R-Neighbor, we first query the nearest neighbors with the moderate value of K and then select potentially profitable samples located in a fixed radius R. The pseudocodes are illustrated in Algorithm 3.

Algorithm 3: K-Neighbor and R-Neighbor algorithm

Input: Training data $(\boldsymbol{X}, \boldsymbol{y}) \in \mathbb{R}^{N\times d} \times \mathbb{R}^N$, the query sample $\boldsymbol{x}$, the number of the nearest neighbors K, and the radius R.

Output: $\hat{p}_{\boldsymbol{x}} \in \mathbb{R}^N$.

K-Neighbor Algorithm:

$\mathcal{N}(\boldsymbol{x})$ = top K points with minimum $d_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{x})$, $\boldsymbol{z} \in X$.

Return: $\hat{p}_{\boldsymbol{x}} = \sum_{\boldsymbol{z}_1 \in \mathcal{N}(\boldsymbol{x})} y_{\boldsymbol{z}_1} \cdot \frac{k_i(\boldsymbol{z}_1, \boldsymbol{x})}{\sum_{\boldsymbol{z}_2 \in \mathcal{N}(\boldsymbol{x})} k_i(\boldsymbol{z}_2, \boldsymbol{x})}$.

R-Neighbor Algorithm:

$\mathcal{N}_1(\boldsymbol{x})$ = top K points with minimum $d_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{x})$, $\boldsymbol{z} \in X$.

$\mathcal{N}_2(\boldsymbol{x}) = \{\boldsymbol{z} : \boldsymbol{z} \in \mathcal{N}_1(\boldsymbol{x}),\ d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{z}) < R\}$.

Return: $\hat{p}_{\boldsymbol{x}} = \sum_{\boldsymbol{z}_1 \in \mathcal{N}_2(\boldsymbol{x})} y_{\boldsymbol{z}_1} \cdot \frac{k_r(\boldsymbol{z}_1, \boldsymbol{x})}{\sum_{\boldsymbol{z}_2 \in \mathcal{N}_2(\boldsymbol{x})} k_r(\boldsymbol{z}_2, \boldsymbol{x})}$.

C MORE EXPERIMENTAL RESULTS

C.1 Trading Transactions

We plot the open-close positions in Fig. 7 to illustratively explain how our proposed LARA framework works. LARA achieves favorable profit in both long and short positions. Even though the forecasting target is 0.1%, more than half of the transactions earn 0.2% return, and some even reach 0.4%, which shows that LARA has indeed captured satisfactory and profitable trading opportunities. Besides, we can find that on up-trend period, we open long positions (red triangles in Fig. 7); in the down-trend period, we open short positions (blue triangles in Fig. 7). Obviously, it is a rational way to open positions when there is a stronger tendency in the price of ETF and do nothing when the price fluctuates around value.

As shown in Fig. 8, we identify the top three machine learning models with the highest precision in Table 4 on the Qlib platform, *i.e.,* LightGBM, LSTM, and GATs. For each method, we select the top 200 signals with the highest predicted probability value in each quarter and perform the t-statistical test of Precision and Average Return between LARA and the corresponding method quarterly (significance level α equals to 0.05). If the curve is above the dotted red line, it represents that LARA is significantly better than the corresponding model, and vice versa. We can empirically find that from 2019-q1 to 2020-q2, LARA steadily outperforms other machine learning baseline models. From 2017-q1 to 2019-q1, LARA is on par with other baseline models. The reason could be attributed to the violent fluctuation of assets price during this period [10].

C.2 Masked Attention Scheme

We run the *masked attention scheme* on a simplified example for further study in Fig. 9. We can obtain *samples with high* $p_{\boldsymbol{x}}$ in the blue region calculated by *masked attention scheme*, whose labels are indeed dominated by the positive ones. We optimize an SVM classifier with the Gaussian kernel on the whole data, which leads to the blue decision boundary located in the mixing regions. However, if we optimize the classifier only on the *samples with high* $p_{\boldsymbol{x}}$, we can get the red decision boundary located in the blue region which is far from the middle mixing regions. Thus it can be more robust to the noise of datasets and generate more profitable signals.

C.3 Ablation Study on Average Return

Following the settings in Fig. 5 left, we perform experiments on LARA and the corresponding machine learning methods in terms of average return under different #Transactions in Fig. 10

D COMPARED METHODS

The main characteristics of all baselines are listed below:

- **Ordinary Least Squares (OLS)** fits a linear model to minimize the residual square sum among observed targets.
- **Autoregressive Integrated Moving Average (ARIMA)** is a class of models that "explains" a given time series based on its own lags and the lagged forecast errors.
- **Ridge** regression minimizes a penalized residual sum of squares by imposing a penalty on the size of the coefficients.
- **Decision Trees** create a model that predicts target values by learning simple decision rules inferred from features.

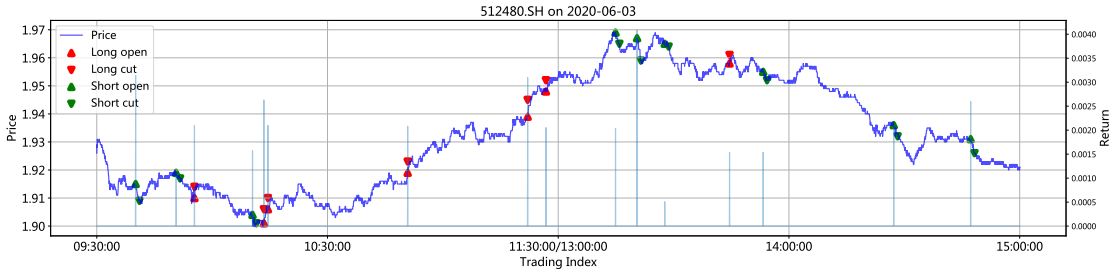


Figure 7: Overview of open-close positions on 512480.SH.

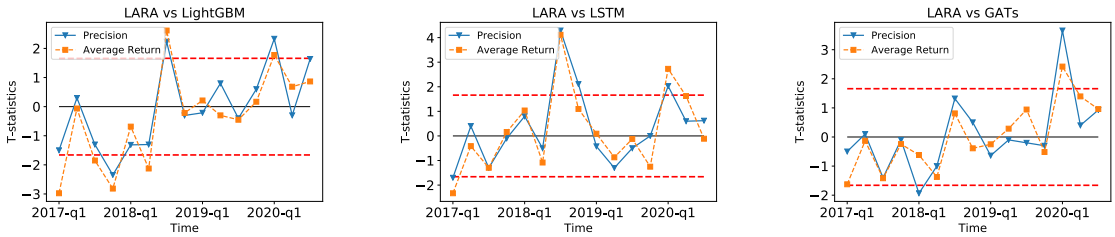


Figure 8: Statistical tests of Precision and Average Return on China's A-share market quarterly. If the curve is above the dotted red line (black line), our proposed framework LARA is significantly better (slightly better) than the corresponding method, and vice versa (significance level α equals to 0.05).

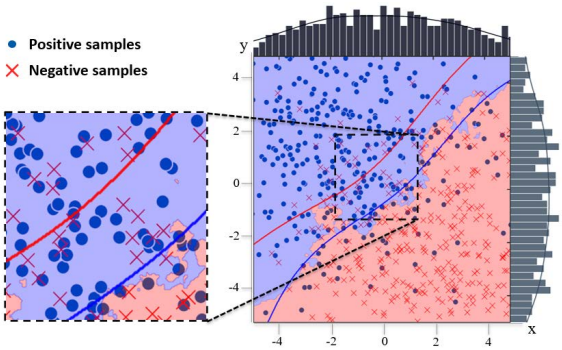


Figure 9: A running example of the *masked attention scheme*. Randomly sample 400 samples from $\mathcal{N}((-2,2),8I_2)$ and $\mathcal{N}((2,-2),8I_2)$ as positive and negative samples, respectively. The blue region denotes the *samples with high* p_x calculated by the *masked attention scheme*. The blue curve is the decision boundary of an SVM classifier with the Gaussian kernel training on the whole set, and the red curve is that training only on the *samples with high* p_x. The left figure is the enlarged view of the area near the decision boundary.

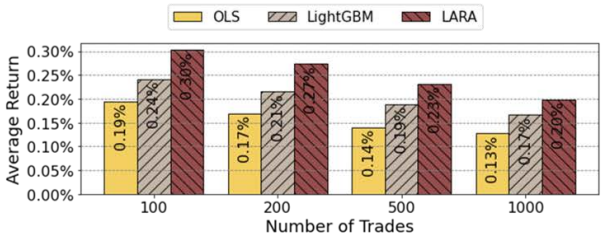


Figure 10: Comparisons with the different number of trades among three methods over Average Return on 512480.SH.

- **AdaBoost [13]** fits a sequence of weak learners on repeatedly modified sample weight of the data and then combines through a weighted majority vote to produce the final prediction.
- **Bagging Regressor** is an ensemble estimator that fits base regressors on random subsets of original datasets and then aggregates individual predictions to form a final prediction.
- **Multi-Layer Perceptron (MLP)** trains iteratively *w.r.t* the loss function to update the parameters. This model optimizes the squared-loss using the stochastic gradient descent.
- **XGBoost [8], LightGBM [18], and CatBoost [25]** are non-linear models based on gradient boosting trees.
- **LSTM [17]** is the vanilla LSTM model, which obtains a sequential embedding; and then an FC layer is used to make the final prediction of return.
- **GRU [9]** is an extended version of LSTM, which has a forget gating mechanism to control the information flow.
- **ALSTM [27]** adds an external attention layer into the vanilla model to adaptively aggregate hidden states' information of previous timestamps.
- **GATs [32]** utilizes the Graph Neural Networks (GNNs) to model the relationship between different stocks, and the attention scheme is incorporated into GNNs.
- **SFM [40]** redesigns the recurrent neural networks by decomposing hidden states into multiple frequency components to model multi-frequency trading patterns.
- **TFT [21]** introduces Deep Momentum Networks to simultaneously learn both trend estimation and position sizing in a data-driven manner.

- **TabNet [3]** is a deep tabular data learning architecture, which uses sequential attention to choose which features to reason from at each decision step.
- **DoubleEnsemble [39]** proposes an ensemble framework leveraging learning trajectory based sample reweighting and shuffling based feature selection.
- **TCTS [35]** introduces a learnable scheduler and adaptively selects auxiliary tasks with the main task.