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:
1 $\mathcal{N}(\boldsymbol{x})$ = top $K$ points with minimum $d_{\boldsymbol{M}}(z, \boldsymbol{x})$, $z \in X$.
2 Return: $\hat{p}_{\boldsymbol{x}} = \sum_{z_1 \in \mathcal{N}(\boldsymbol{x})} y_{z_1} \cdot \frac{k_i(z_1, \boldsymbol{x})}{\sum_{z_2 \in \mathcal{N}(\boldsymbol{x})} k_i(z_2, \boldsymbol{x})}$.

R-Neighbor Algorithm:
1 $\mathcal{N}_1(\boldsymbol{x})$ = top $K$ points with minimum $d_{\boldsymbol{M}}(z, \boldsymbol{x})$, $z \in X$.
2 $\mathcal{N}_2(\boldsymbol{x}) = \{z : z \in \mathcal{N}_1(\boldsymbol{x}),\ d_{\boldsymbol{M}}(\boldsymbol{x}, z) < R\}$.
3 Return: $\hat{p}_{\boldsymbol{x}} = \sum_{z_1 \in \mathcal{N}_2(\boldsymbol{x})} y_{z_1} \cdot \frac{k_r(z_1, \boldsymbol{x})}{\sum_{z_2 \in \mathcal{N}_2(\boldsymbol{x})} k_r(z_2, \boldsymbol{x})}$.
```

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.
- **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.

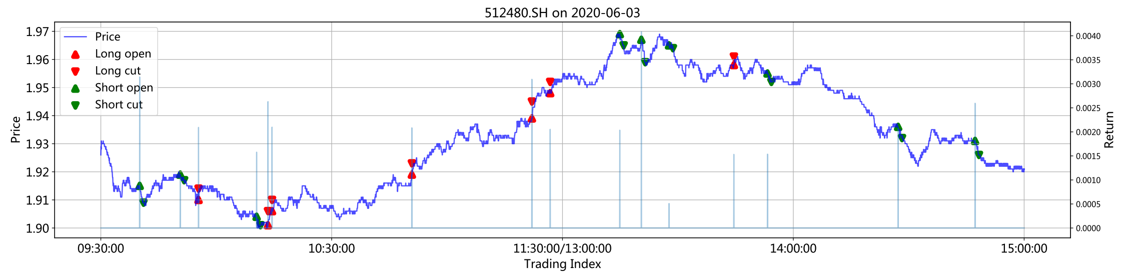

Figure 7: Overview of open-close positions on 512480.SH.

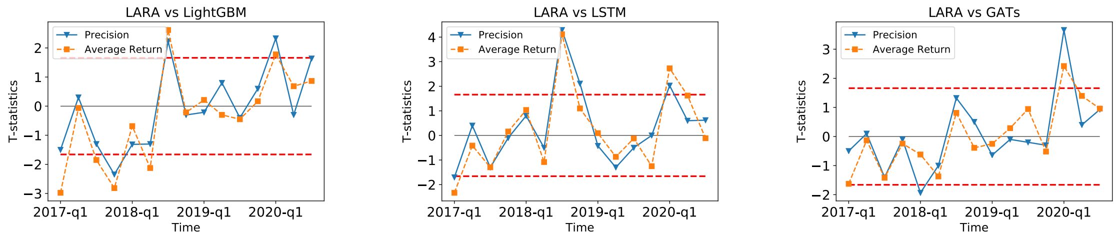

Figure 8: Statistical tests of Precision and Average Return on China's A-share market quarterly. If the curve is above the dotted red line (black line), our proposed framework LARA is significantly better (slightly better) than the corresponding method, and vice versa (significance level α equals to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

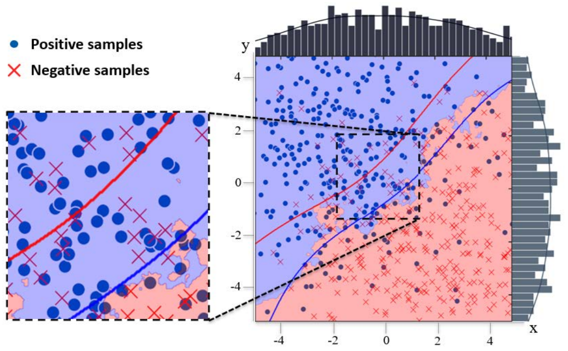

Figure 9: A running example of the *masked attention scheme*. Randomly sample 400 samples from $\mathcal{N}((-2,2),8I_2)$ and $\mathcal{N}((2,-2),8I_2)$ as positive and negative samples, respectively. The blue region denotes the *samples with high* p_x calculated by the *masked attention scheme*. The 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.

- **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.

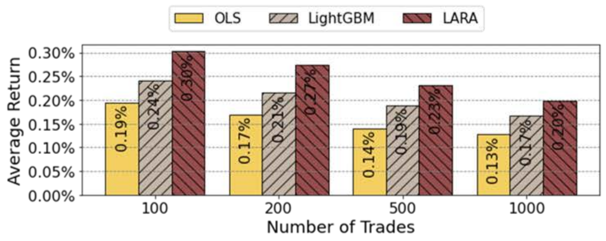

Figure 10: Comparisons with the different number of trades among three methods over Average Return on 512480.SH.

- **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.