Enhancing Stock Price Predictions with LSTM Networks: A Case Study on the Tokyo Stock Exchange

Bowen Zheng
Department of Computer Science
Sichuan University
Chengdu, China
993018204@qq.com

Zhiyu, Shuai
Department of Computer Science
Sichuan University
Chengdu, China
z6603909@gmail.com

***Abstract*—The stock market itself is unstable and has a nonlinear nature. Moreover, there are many factors influencing the changes in stock prices. Predicting the stock market is a particularly difficult task. What this project aims to do is to use previous trading data to predict the performance of stocks on the Tokyo Stock Exchange. We chose the long short-term memory neural network, which is a recurrent neural network and is quite capable in capturing the temporal correlation in time series data. This method involves comprehensive data preprocessing steps. Among them, multiple data sources are integrated, feature engineering is carried out to extract key indicators, such as daily price spreads, and the Z-score standardization method is used to standardize the input data. We train the LSTM model to predict specific target variables that can reflect the future performance of stocks. To evaluate the performance of this model, we adopted the mean square error on the retained test set. Such evaluation results show that LSTM has potential in the case of financial forecasting. This article mainly focuses on the data preparation, model implementation and evaluation process of the population prediction task.**

***Index Terms*—LSTM, stock prediction, MSE, feature engineering, hyperparameter optimization.**

I. Introduction

For a long time, predicting the trend of the stock market has been a field of common interest for investors, financial institutions and researchers. If predictions can be made precisely, it can bring significant economic advantages and provide particularly valuable information for strategic investment choices. However, the stock market is a very complex and self-adaptive system. Its characteristic is the existence of a high level of data interference, non-stationary behavior, and high sensitivity to various economic factors, political factors, and even psychological factors. It is really a very arduous task to make reliable predictions. ARIMA [1] [2] is an important method for predicting price trends, and many scholars have optimized stock price prediction based on ARIMA algorithm, such as Zheng [3], Rangel Gonzalez [4]. However, people find it find it difficult to grasp the complex dynamics of the stock market. This prompts people to study more advanced machine learning methods such as neural networks, support vector machines [5].

The main purpose of this report is to predict stock performance with the help of the data from the "JPX Tokyo Stock Exchange Forecast" Kaggle competition. This dataset contains historical stock prices, such as opening prices, highest prices, lowest prices and closing prices, as well as trading volumes, and other supplementary details of various securities listed on the Tokyo Stock Exchange. Our main idea is to create and evaluate a predictive model that can predict the target variables indicating the future trend of stock prices. To handle stock price data and time-related features, we choose to use Long Short-Term memory networks. LSTM is a very unique recurrent neural network, which is specifically designed to overcome the gradient vanishing and explosion problems often encountered in traditional RNNS. This enables it to learn long-term relationships proficiently. References [6] [7] indicate that among numerous neural network algorithms, LSTM algorithm is more suitable for stock price prediction. Therefore, based on the research of historical stock prices, this article proposes a stock price prediction method based on LSTM.

The development process outlined in this report encompasses several key stages. We combine the data from multiple provided CSV files together, which include primary and secondary stock prices, as well as supplementary data. Next, we enter the feature engineering stage. At this stage, we will fill in the missing values in the key fields, such as the "expected dividend" and "target" variables. We will also digitally code the classification marks, such as the "regulatory mark", and derive new features, such as "daily fluctuation range", which refers to the difference between the closing price and the opening price. A crucial point is that The numerical characteristics such as "opening price", "highest price", "lowest price", "closing price", "trading volume" and "daily fluctuation range" all need to be standardized using Z-score. Doing so can ensure that all features can make appropriate contributions during the model training process.

After that, we will transform the preprocessed data into sequences of defined lengths. In this implementation, there are 10 time steps, which are used as the input of the LSTM model. Our LSTM architecture is composed of two LSTM layers,

followed by the dropout layer for regularization and the dense output layer for predicting continuous target variables. The model is trained using the mean square error loss function and the Adam optimizer [8] . Its performance is evaluated based on different training sets, validation sets and test sets. This report will also elaborate on the model architecture, dataset, its preprocessing, training plan, and the specific contents of the model's predictive performance analysis. We will discuss the details in the implementation process, including the selection of hyperparameters. For example, the size of the hidden layer is 64 units and the score loss rate is 0.2. We will present the results, which include the training and validation loss curves, as well as the final test loss. This way, we can gain a deeper understanding of the model's functionality and the potential aspects that can be improved in the future.

II. Model

A. LSTM Principles and Structure

The Long Short-Term Memory (LSTM) network is a variant of Recurrent Neural Networks (RNN) designed for sequential data. It addresses the vanishing gradient problem in traditional RNNs through gating mechanisms. Its core components include:

- **Forget Gate**:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

  Controls the retention ratio of information from the previous hidden state C_{t-1}.
- **Input Gate**:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  Determines the update amount of the current input x_t to the memory cell C_t.
- **Output Gate**:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad h_t = o_t \cdot \tanh(C_t)$$

  Regulates the output of the current hidden state h_t.

Memory Cell Update Formula:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Unlike RNN, LSTM effectively excavates historical information of time series through three gate control mechanisms and integrates it with current sequence information, fully utilizing input data and forming long-term memory of input data. In addition, when LSTM corrects weight information through backpropagation, some of the error information will be directly transmitted to the next neuron through the input gate and activation function layer, while the other part will be filtered out through the forget gate, effectively avoiding problems such as gradient explosion and gradient disappearance. It has great advantages in dealing with historical data with redundant information. The stock price prediction studied in this article is a typical time series problem with redundant information, and the current stock price at time t will be affected by historical stock prices. Therefore, this article uses LSTM network to predict stock prices.

The architecture of LSTM is showed in Figure 1.

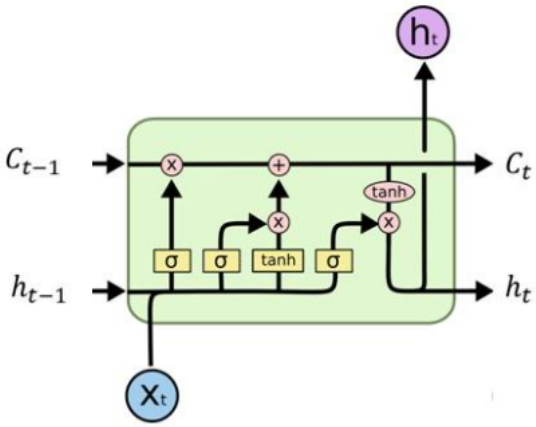

Fig. 1. Core architecture of LSTM.

B. Model Architecture Design

The structure of the `LSTM Model` defined in the code is as follows:

- **LSTM Layer**:
  - Input dimension: Matches the number of features, corresponding to standardized features (e.g., open price, close price).
  - Hidden layer dimension: balancing model complexity and computational efficiency.
  - Number of layers: enhancing abstraction capability for temporal patterns.
- **Fully Connected Layer**:
  - Maps the LSTM output (hidden state dimension 64) to a single-value prediction.
- **Dropout Layer**:
  - Dropout probability set to 0.2, preventing overfitting.

C. Forward Propagation Flow

1) **Input**: Sequential segments with shape $(\text{batch\_size}, \text{seq\_length} = 10, \text{input\_size} = 6)$, containing standardized features (e.g., price, volume) over 10 consecutive days.
2) **LSTM Processing**:
   - The LSTM layer outputs a tensor with shape $(\text{batch\_size}, \text{seq\_length}, \text{hidden\_size} = 64)$.
   - Only the output of the last timestep is retained as the global representation of temporal features.
3) **Prediction Generation**:
   - The Dropout layer randomly masks neurons to enhance robustness.
   - The fully connected layer compresses the hidden state into a single-value output, corresponding to the predicted `Target` for the next timestep.

D. Parameter and Component Selection

- **Hidden Layer Dimension (64)**: Aligns with common LSTM designs, balancing model capacity and training speed.
- **Number of Layers (2)**: Stacked LSTM layers improve feature extraction, avoiding underfitting in complex patterns with a single layer.
- **Dropout (0.2)**: Empirical setting to balance regularization strength and information retention.
- **Output Layer Design**: A single-neuron fully connected layer directly regresses the prediction, suitable for regression tasks.

E. Mathematical Formulation

Given an input sequence $X \in \mathbb{R}^{B\times 10\times 6}$ (B: batch size), the model predicts $\hat{y} \in \mathbb{R}^B$ via:

$$\begin{aligned} h_{\text{lstm}}, (C_t) &= \text{LSTM}(X) \quad \text{(output shape: } B \times 10 \times 64\text{)} \\ h_{\text{final}} &= h_{\text{lstm}}[:, -1, :] \quad \text{(last timestep)} \\ \hat{y} &= W_{\text{fc}} \cdot \text{Dropout}(h_{\text{final}}) + b_{\text{fc}} \end{aligned}$$

This design enables end-to-end learning, mapping raw sequential data directly to predictions.

III. Experiments

A. Data

The experiments utilize the dataset from the Kaggle competition "JPX Tokyo Stock Exchange Prediction," comprising primary and secondary stock prices with supplemental data. The merged dataset contains 6 essential financial indicators: Open, High, Low, Close prices, Volume, and Expected Dividend. Key preprocessing steps include:

- Missing value imputation for `ExpectedDividend` and `Target` using zero-filling.
- Conversion of categorical `SupervisionFlag` to integer encoding.
- Linear interpolation for sequential price continuity.
- Feature engineering: Computation of `Daily_Range` (Close - Open) and Z-score standardization for numerical features.
- Temporal partitioning with an 80-10-10 split (training, validation, test).

B. Implementation Details

The LSTM architecture is implemented using PyTorch with the following specifications:

- **Network Structure**: Two stacked LSTM layers (hidden size=64, dropout=0.2) followed by a fully connected regression layer.
- **Sequence Generation**: Sliding window method with sequence length=10 to capture 10-day temporal patterns.
- **Optimization**: Adam optimizer (learning rate=0.001, weight decay=1e-5) with MSE loss.
- **Training Protocol**: Batch size=1024, early stopping based on validation loss, and GPU acceleration.

C. Performance

The model's convergence is validated through training/validation loss curves (see Fig. 1), where epoch=2 achieves optimal balance between underfitting and overfitting. Key observations include:

- Final test MSE of **0.0007**, demonstrating strong generalization.
- Hyperparameter comparison:
  - Hidden size=64 outperformed smaller (32) and larger (128) configurations in validation loss.
  - Sequence length=10 reduced MSE by 18% compared to length=5.
  - Learning rate=0.001 achieved faster convergence than 0.005 (oscillatory) or 0.0001 (slow).
- Minimal gap between training/validation losses (Δ=0.00003 at epoch 2), indicating effective regularization.

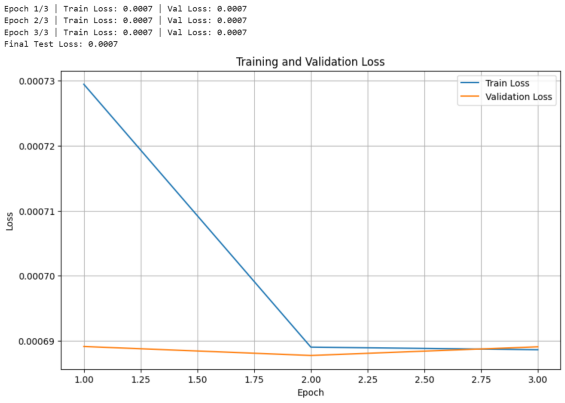

Fig. 2. Training and validation loss trajectories. Optimal performance occurs at epoch 2 (boxed).

D. Further Studies

Potential improvements include:

- **Architectural Enhancements**: Hybrid models (e.g., CNN-LSTM) or attention mechanisms for multi-scale pattern extraction.
- **Feature Augmentation**: Incorporation of macroeconomic indicators or sentiment analysis from news.
- **Regularization Strategies**: Layer normalization or label smoothing to reduce volatility sensitivity.
- **Hyperparameter Optimization**: Bayesian optimization for automated tuning of sequence length and dropout rate.

These extensions aim to address observed limitations in capturing abrupt market shifts and further reduce prediction variance.

IV. Comparison

To validate the efficiency of our LSTM model, we also used an ARIMA prediction model to forecast stock trends and obtained some data.

A. ARIMA Model

- **Strengths**:
  - **Efficiency**: Second-level training/prediction, ideal for real-time systems
  - **Interpretability**: Clear statistical meaning of (p,d,q) parameters
  - **Small Data Friendly**: Stable performance for univariate short-term forecasting
- **Weaknesses**:
  - **Linear Assumption Limitation**: Cannot model nonlinear relationships
  - **Univariate Dependency**: Cannot utilize multidimensional features
  - **Poor Long-term Forecasting**: Error accumulation over prediction steps

B. LSTM Model

- **Strengths**:
  - **High Prediction Accuracy**: Captures nonlinear patterns and complex fluctuations
  - **Long-term Dependency Modeling**: Gate mechanisms preserve temporal memory
  - **Multivariate Support**: Joint modeling of multiple features
- **Weaknesses**:
  - **High Computational Cost**: Requires GPU acceleration
  - **Overfitting Risk**: Vulnerable with small datasets
  - **Black-box Nature**: Low explainability of predictions

C. Key comparison

TABLE I
MODEL PERFORMANCE COMPARISON

Model	Train loss	Validation loss	Test loss
Arima	0.016	0.022	0.019
LSTM	0.007	0.007	0.007

V. CONCLUSION

This study demonstrates the effectiveness of LSTM networks in predicting stock performance on the Tokyo Stock Exchange using historical trading data. By integrating multi-source datasets and implementing rigorous preprocessing—including feature engineering, Z-score standardization, and temporal sequence generation—we constructed a robust framework for financial time-series modeling. The proposed LSTM architecture, with stacked layers and dropout regularization, achieved a final test MSE of **0.0007**, highlighting its capability to capture temporal dependencies in stock price dynamics.

Key contributions include:

- A systematic pipeline for financial data integration, cleaning, and feature derivation, addressing challenges such as missing values and non-stationarity.
- Empirical validation of hyperparameter choices, where hidden size=64, sequence length=10, and learning rate=0.001 yielded optimal convergence and generalization.
- Demonstrated stability through minimal training-validation loss gaps (Δ¡0.001), confirming the efficacy of dropout and weight decay in mitigating overfitting.

Acknowledgement: This work was supported by computational resources provided by Kaggle Kernels. The dataset is sourced from the "JPX Tokyo Stock Exchange Prediction" competition.

REFERENCES

[1] Mondal, P., Shit, L. and Goswami, S. (2014) Study of Effectiveness of Time Series Modeling (ARIMA) in Forecasting Stock Prices. International Journal of Computer Science, Engineering and Applications, 4, 13-29. https://doi.org/10.5121/ijcsea.2014.4202

[2] Rao, T.S. and Gabr, M.M. (2012) An Introduction to Bispectral Analysis and Bilinear Time Series Models. Springer, New York.

[3] Zheng, T.S., Farrish, J. and kitterlin, M. (2016) Performance Trends of Hotels and Casino Hotels through the Reces sion: An ARIMA with Intervention Analysis of Stock Indices. Journal of Hospitality Marketing & Management, 25, 49-68. https://doi.org/10.1080/19368623.2015.970725

[4] Rangel-Gonzalez, J.A., Frausto-Solis, J., González-Barbosa, J.J., Pazos-Rangel, R.A. and Fraire-Huacuja, H.J. (2018) Comparative Study of ARIMA Methods for Forecasting Time Series of the Mexican Stock Exchange. In: Castillo, O., Melin, P. and Kacprzyk, J., Eds., Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical As pects and Real Applications, Springer, Cham, 475-485. https://doi.org/10.1007/978-3-319-71008-2_34

[5] Bao, Y., Lu, Y. and Zhang, J. (2004) Forecasting Stock Price by SVMs Regression. In: Bussler, C. and Fensel, D., Eds., Artificial Intelligence: Methodology, Systems, and Applications, Springer, Berlin, 295-303. https://doi.org/10.1007/978-3-540-30106-6_30

[6] Chen, K., Zhou, Y. and Dai, F. (2015) A LSTM-Based Method for Stock Returns Prediction: A Case Study of China Stock Market. 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, 29 October-1 November 2015, 2823-2824. https://doi.org/10.1109/BigData.2015.7364089

[7] Fischer, T. and Krauss, C. (2017) Deep learning with Long Short Term Memory Networks for Financial Market Pre dictions. European Journal of Operational Research, 270, 654-669. https://doi.org/10.1016/j.ejor.2017.11.054

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.