Comparative Analysis of Machine Learning and Deep Learning Models for Alibaba Stock Price Prediction

Ruoqing Gao, Luxin Xu, Jingjie Ouyang, Mengqing Fan

Abstract

Stock price prediction has become a research hotspot in financial decision-making and risk management due to its complexity and high uncertainty. This study focuses on forecasting the closing price of Alibaba stock using multiple machine learning and deep learning models, including XGBoost, LSTM, GRU, RNN, and 1D-CNN. In the data preprocessing stage, feature normalization was applied, and technical indicators such as RSI and MACD were calculated to enrich the input data. The models were trained and evaluated on the same dataset, and their performance was compared using evaluation metrics. Results show that XGBoost outperformed other models in terms of prediction accuracy and stability, while GRU and RNN showed moderate performance. The findings provide valuable insights into the effectiveness of different models in real-world stock price forecasting tasks.

Index Terms—Stock prediction, financial time series, machine learning models, deep learning, technical indicators, data preprocessing

1. INTRODUCTION

The dataset used in this paper is Alibaba Group (NYSE: BABA) stock data from Kaggle by M Hassan Saboor (https://www.kaggle.com/datasets/mhassansaboor/alibaba-stock-dataset-2025), which extracted using the Yahoo Finance API (yfinance), a powerful Python library for extracting real-time and historical stock market data. The data records the trading records of Alibaba's stock from January 2014 to February 2025. The data contains 7 indicators: 1) The trading date (YYYY-MM-DD); 2) The stock price at market open; 3) The highest stock price of the day; 4) The lowest stock price of the day; 5) The stock price at market close; 6) The adjusted closing price after splits and dividends; 7) The number of shares traded. The total number of samples is 2617. In order to further improve the performance of the model, we use feature engineering to construct a variety of technical indicators based on the original data, including RSI, MACD, Bollinger Bands, Stochastic Oscillator, etc. The training set is divided from the test set in chronological order to ensure that future test data does not leak into the training phase.

The research focuses on two questions: (1) Whether it is possible to predict the future stock closing price based on historical price data and indicators, and compare the performance of different models. (2) Among the extracted features, which indicators plays the most critical role in the model prediction performance. To this end, we construct and compare a variety of models for time series prediction, including sliding window sequence based deep learning models CNN, RNN, LSTM and GRU, and tree model based XGBoost. We use them to evaluate the prediction accuracy and model stability respectively.

Before modeling, we performed preliminary Exploratory data analysis (EDA). We extracted feature importance ranking through the random forest model, and then visualized the normalized metrics using boxplots to examine their distribution, which all provided support for subsequent modeling.

We will begin with section 3 to describe each of the methods we used, and then Section 4 to describe the data processing, parameter setting and final results in detail. In Section 5, we will discuss and compare the methods used in this paper, and at the end of the article, Section 6, we will make a summary and future outlook.

2. LITERATURE REVIEW

The use of deep learning in stock market prediction has gained significant traction in recent years due to its ability to model complex nonlinear and temporal relationships. In 2017 Selvin used a sliding window approach with a 100-minute window and 90-minute overlap to predict stock prices 10 minutes ahead. This method allowed the models to learn recent patterns effectively and generate dense training samples for short-term forecasting [1].

In 2021, Gao with their team compared optimized LSTM and GRU models for stock price prediction, using PCA and LASSO as feature selection techniques. Their experiments on the Shanghai Composite Index showed that both models performed well, with LASSO-based feature selection leading to higher forecasting accuracy in most cases [2].

Another study by Zhenglin Li in 2023, they explored the application of LSTM neural networks to predict stock prices of leading technology companies, including Apple, Google, Microsoft, and Amazon. Their research demonstrated that an LSTM model, composed of two LSTM layers followed by dense layers, effectively captured complex temporal patterns in historical stock price data, achieving promising accuracy [3].

In 2023, Chavhan conducted a comprehensive comparative study evaluating the performance of LSTM, RNN, and GRU models for stock price prediction. Using historical market data, they addressed issues such as volatility and noise through preprocessing and employed multiple metrics like MSE, RMSE, and R^2 to assess model performance. The findings suggest that while all three models are effective, GRU slightly outperforms others in predictive accuracy [4].

In 2023, Zhang proposed a stock price prediction method based on the XGBoost algorithm, emphasizing its ability to capture short-term time series fluctuations through optimized hyperparameter tuning. The study highlighted the use of technical indicators such as MACD and RSI as critical input features, showing that the model can achieve high accuracy in short-term trend forecasting while also offering interpretability through feature importance analysis [5].

Hu, Zhao, and Khushi conducted a comprehensive survey of deep learning techniques applied to both Forex and stock price prediction in 2021. The authors classified 86 reviewed papers based on deep learning architectures such as CNN, RNN, LSTM, DNN, reinforcement learning, and other hybrid models. Their analysis revealed that LSTM-based models were the most widely used due to their effectiveness in capturing long-term dependencies in sequential data. They also found that combining LSTM with attention mechanisms or CNN could further enhance predictive performance. Moreover, the study noted that deep learning methods significantly outperformed traditional statistical models in financial forecasting tasks and that the trend of using such methods has increased sharply since 2017 [6].

Building on these insights, our research aims to systematically compare CNN, RNN, LSTM, GRU, and XGBoost models using the Alibaba stock dataset. Maintaining the Integrity of the Specifications.

3. METHODOLOGY

In recent years, many studies have applied various deep learning models for stock prediction. To compare the performance of currently popular approaches in this field, we have selected the following five models.

3.1. CNN

Lecun et al. introduced the Convolutional Neural Network (CNN) model in 1998, which consists of three major components: the convolutional layer, the pooling layer, and the fully connected layer. Recent advances in time series analysis have increasingly incorporated deep learning models. Among these, CNNs have demonstrated strong performance in time series forecasting and have been widely applied to stock price prediction in the financial domain [7].

In this report, a one-dimensional Convolutional Neural Network (1D-CNN) model was constructed as illustrated in Figure 1, which includes an input layer, convolutional layer, pooling layer, dropout layer, fully connected layer, and a single-neuron dense output layer. The model was trained using the Mean Squared Error (MSE) as the loss function and optimized with an adaptive learning algorithm. During training, both single-layer and two-layer convolutional structures were explored, and the results indicated that the single-layer architecture yielded more accurate predictions.

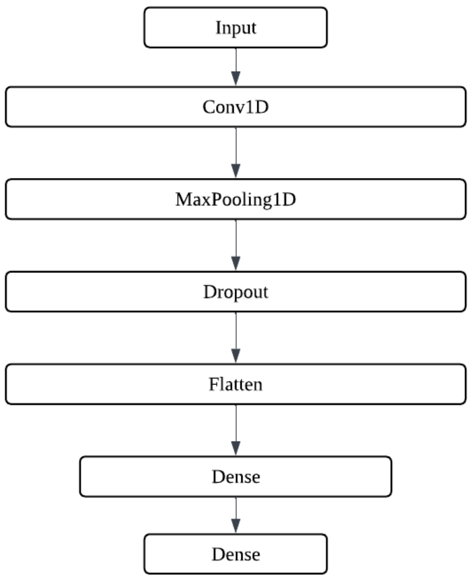

Fig. 1. CNN Architecture.

3.2. RNN

Recurrent neural network (RNN) is a kind of neural network model specializing in processing sequence data, which can effectively handle sequential data by introducing temporal dependencies between observations. Different from traditional feedforward neural networks, RNNs maintain a hidden state and can effectively capture the temporal dependencies in time series data, thus providing context information for time series prediction tasks such as stock price prediction [8]. In this study, we adopt the structure of a simple RNN and use a sliding window of historical features as input to model temporal patterns in stock price data.

The core computation in a simple RNN follows the formulation:

$$ht = gWxt + Uht - 1 + b \tag{1}$$

where h_t is the hidden state at time t, x_t is the input vector, W and U are weight matrices, b is the bias, and g is a non-linear activation function, typically the hyperbolic tangent (tanh) [9]. However, simple RNNs suffer from certain limitations, including their inability to learn long-term dependencies effectively due to issues like vanishing gradients and short memory span [10]. Despite this, they are computationally efficient and serve as a good baseline for sequence learning.

In our implementation, a single-layer RNN model consisting of 64 hidden units and tanh activation function is constructed, followed by a dropout layer to prevent overfitting, and finally the prediction results are output through a fully connected layer. To optimize the model configuration, Grid Search combined with five-fold Time Series Cross-Validation is used to tune hyperparameters such as window size, number of units, dropout rate, and learning rate.

3.3. XGBoost

Extreme Gradient Boosting (XGBoost) is a powerful ensemble learning method based on the Gradient Boosting Decision Tree (GBDT) framework. It boosts the performance of weak learners through iterative residual learning [11].

The XGBoost model can be represented as:

$$yi = k = 1Kfkxi, \quad fk \in F \tag{2}$$

where $F = \{f(x) = w_{q(x)}\}$ means a set of regression trees, q is the structure mapping the input x to a leaf, and w is the score of that leaf. To control model complexity, XGBoost introduces regularization through the following objective function:

$$Obj = i = 1nyi - yi2 + \gamma T + 12\lambda j = 1Twj2 \tag{3}$$

where γ and λ are regularization parameters, and T is the number of leaves in the tree. The second term helps mitigate overfitting by penalizing overly complex models [12][13].

In this study, XGBoost was used to predict the closing stock price of Alibaba. In order to introduce temporal dependencies, we add lag features (1–3 days) to several technical indicators such as Close, Volume, RSI, MACD, forming a sliding window structure. And hyperparameters such as learning rate, tree depth, and sample subsampling rate are tuned using GridSearchCV.

3.4. GRU

The Gated Recurrent Unit (GRU) [14] proposed by Cho et al. is a variant of Recurrent Neural Networks (RNN) designed to efficiently capture temporal dependencies in continuous data. GRU introduces two gating mechanisms - update gates and reset gates - to regulate the flow of information. These gates allow the model to retain relevant temporal information in long sequences while discarding irrelevant data at appropriate points. The update gate controls the retention of current information, while the reset gate determines the effect of the previous state on the current state [15]. GRU combines the unitary and hidden states into a single vector, which simplifies the network structure, reduces computational complexity, and improves training efficiency.

The computations of the update and reset gates are defined as follows:

$$\text{Update gate:} \quad z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{4}$$

$$\text{Reset gate:} \quad r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{5}$$

The model used in this paper consists of a GRU layer, a Dropout layer and two fully connected layers, where the GRU layer adopts the standard package module provided by TensorFlow Keras, and the overall model is trained with the mean square error as the loss function and the Adam optimiser, which is used to deal with the task of regression of time series with a fixed window length.

3.5. LSTM

Long Short-Term Memory (LSTM) networks are an advanced type of recurrent neural network specifically designed to address the vanishing gradient problem that commonly arises in traditional RNNs when handling long sequences. By introducing a memory cell (cell state) and three gating mechanisms (forget gate, input gate, and output gate), LSTM can retain crucial long-term dependencies during information processing, making it particularly suitable for complex time series modeling tasks such as stock price prediction [16].

In this study, we constructed a single-layer LSTM model with 64 hidden units and a tanh activation function. A Dropout layer with a rate of 0.3 was applied after the main network to prevent overfitting. The input sequence had a window length of 30 time steps and included daily opening price, high, low, volume, and several technical indicators (e.g., RSI, MACD, Stochastic_K), along with time-related features encoded using sine and cosine transformations for weekday and month [17].

The core mechanism of LSTM is reflected in its gating structure, particularly the forget gate and output gate. The computation for the forget gate is as follows:

$$f_t = \sigma\left(W_f \cdot [h_t{}^{-1}, x_t] + b_f\right) \tag{6}$$

This gate controls how much of the previous state Ct−1 information should be retained at the current time step.

The output gate determines the final hidden state output, calculated as:

$$o_t = \sigma(W_o \cdot [h_t{}^{-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \odot tan\,h(C_t) \tag{8}$$

The model was trained using the Adam optimizer, and hyperparameters were optimized through grid search combined with five-fold time series cross-validation. This structure has been shown effective in previous works for capturing financial time series dynamics and handling nonlinear relationships in stock data [18].

4. RESULTS AND EVALUATIONS

4.1. Data analysis

This Dataset contains historical stock prices for Alibaba Group (NYSE: BABA) from January 2014 to February 2025. It has seven columns, including the date, opening price, highest price, lowest price, closing price, adjusted closing price, and daily trading volume. The closing price was selected as the target variable for prediction.

4.2. Data Pre-processing

4.2.1. Data Cleaning and Splitting

The dataset was initially loaded and examined for missing values and incorrect data types. The Date column was converted to datetime format to ensure consistency in time-based operations. After these steps, the dataset was checked for duplicate dates and entirely duplicated rows. Finally, the data was sorted by date to facilitate time-based splitting for model training and evaluation.

Before model training, the database was split into two time-based subsets: a training set (data from September 19, 2014 to January 29, 2024) and a test set (data from January 30, 2024 to February 13, 2025). This temporal division was intended to simulate practical time series forecasting scenarios and to prevent data leakage from future information.

4.2.2. Feature Engineering

4.2.2.1. Date cyclical encode

To improve the model's ability to capture temporal patterns, the year, month, and day were extracted from the Date column, and an additional column indicating the weekday was created. Moreover, sinusoidal (sin) and cosinusoidal (cos) encodings were applied to represent the cyclical nature of time-related features.

4.2.2.2. Financial Technical indicators

Since the original dataset contained only basic stock market records, additional financial technical indicators were introduced to enable a more comprehensive analysis of cyclical and trend patterns in the time series. Inspired by the market-based technical indicators mentioned in the reports of Htun [19] and Chanrungmaneekul [20], The following indicators were included in the model:

- Relative Strength Index (RSI): A 14-day momentum indicator reflecting the magnitude of recent price changes.

$$RSI = 100 - \left(\frac{100}{1 + \frac{Gains}{Losses}}\right) \tag{9}$$

- Stochastic Oscillator: A 14-day window indicator used to identify potential price reversal signals.

$$\%K = 100 \times \frac{Close - Low}{High - Low} \tag{10}$$

- Moving Average Convergence-Divergence (MACD): Calculated with a short-term window of 12, long-term window of 26, and signal window of 9, providing the MACD Line, Signal Line, and Histogram.

$$MACDLine = EMA_{12} - EMA_{26} \tag{11}$$

$$SignalLine = EMA_{9} \tag{12}$$

$$Histogram = MACD - Signal \tag{13}$$

- Bollinger Bands: The upper, middle, and lower bands were calculated using a 20-window size and added to quantify price volatility.

$$MiddleBand = SMA_{20} \tag{14}$$

$$UpperBand = SMA_{20}(P) + 2\sigma_{20}(P) \tag{15}$$

$$LowerBand = SMA_{20}(P) - 2\sigma_{20}(P) \tag{16}$$

Because all of these technical indicators all rely on rolling time windows, the absence of sufficient historical data at the beginning of the series results in NaN values. Given that the dataset spans over ten years, we addressed this issue by removing the first 26 rows that contained NaN values. These indicators complement each other and provide valuable insights into underlying market trends within the time series data.

4.2.3. Random Forest-Based Feature Selection

In regression tasks, Random Forest is one of the most popular feature selection methods. It ranks features by calculating feature importance scores using the Mean Decrease in Impurity (MDI) method [21]. To identify the features that contribute most significantly to stock prediction and to mitigate risks such as overfitting, the Random Forest algorithm was employed to rank feature importance and evaluate model performance across different feature subsets.

During the preprocessing stage, an initial experiment was conducted by retaining only the top four most important features. However, given the dataset spans over a decade and contains a large volume of data, using only a few features was insufficient for the deep learning model to effectively capture the complex relationships between variables. Therefore, the least informative features were considered for exclusion, including the original date-related features, which were removed and have been replaced with their cyclically encoded versions. All remaining features were used for model training.

4.2.4. Data normalisation

Given the sensitivity of deep learning models to the scale of input features, all numerical variables were standardized using min-max normalization to constrain their values within a uniform range. The feature ranges after data normalization are shown in Figure 2.

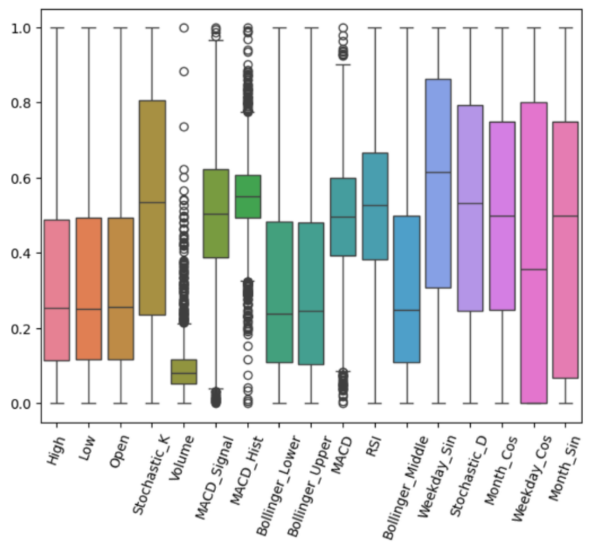

Fig. 2. Data Normalisation.

4.3. Experimental Design

To investigate the predictive capabilities of different algorithmic paradigms, we implement and compare a series of representative models, including RNN, LSTM, GRU, 1D-CNN, and XGBoost. Experiments were conducted around these models in parallel to evaluate their forecasting capabilities under the same data and experimental settings. To enhance the robustness of time series modeling, this study adopts time series segmentation along with cross-validation strategies. The models in this study follow a similar construction process. First, data preprocessing is performed, For the deep learning models (GRU, LSTM, RNN, and 1D-CNN), a sliding window approach is used to construct input sequences from historical data points. Each model is trained using the respective algorithm's architecture: RNNs and their variants (GRU, LSTM) use recurrent layers to capture temporal dependencies, while 1D-CNN utilizes convolutional layers for feature extraction. XGBoost is applied as a gradient boosting model for comparison. Hyperparameter optimization for all models is performed using grid search combined with time series cross-validation to ensure model generalization. The performance of the models is quantitatively evaluated by MSE and R² metrics, and the forecasting effect of each model is demonstrated visually to verify the applicability and performance differences of different models in financial time series forecasting tasks.

4.3.1. GridSearchCV

Grid search is utilized in this experiment to perform hyperparameter optimization by exhaustively evaluating all candidate parameter combinations, thereby identifying the most effective model configuration.

4.3.2. Time Series Segmentation Strategys

The core idea of the time series segmentation strategy is to strictly follow the chronological order in the modelling process to ensure the temporal consistency of the model training and testing data, and to avoid the problem of future information leakage that may be brought by the traditional random segmentation [22]. In the time series prediction task, the future information cannot be 'seen' by the model in advance in the training stage, otherwise the prediction ability of the model will lose its practical significance. In this experiment, the sliding window method is used for time series segmentation. Specifically, the training set is always located before the test set, so that the time horizon of each round of training set is earlier than the corresponding test set, which ensures that the model can only use the past data to learn during each training. Under the sliding window strategy, the position of the test set is gradually slid backwards as the training set expands, allowing the model to be validated on different time frames and thus evaluate its performance more comprehensively.

4.3.3. Time series cross-validation

Time series cross-validation further enhances the robustness of model assessment based on the time series segmentation described above. It ensures that the model can adapt to data from different time windows through a cycle of multiple rounds of training and validation, enhancing the reliability and generalisation of model evaluation. The traditional cross-validation method of randomly partitioning the dataset is not applicable in time series tasks, as it may result in future data appearing in the training set, generating information leakage. This experiment uses the TimeSeriesSplit method in conjunction with a sliding window strategy for cross-validation. TimeSeriesSplit performs training and validation by dividing the data into multiple time periods. For each validation, the model is trained with a segment of historical data and then tested in the immediately following time period. With each round of validation, the training set is gradually increased while the test set is kept constant or slid backwards, ensuring that the data from each time window can be used as a test set for validation, enhancing the stability and reliability of the model.

4.4. Parameter Settings

The optimal combination of hyperparameters for each model was determined by combining grid search with time series cross-validation. Key hyperparameters considered for models such as GRU, LSTM, RNN, 1D-CNN included units, dropout rate, learning rate, and for XGBoost, max depth, and n_estimators. The optimal hyperparameter configurations, identified through grid search in combination with time series cross-validation, are summarized in Tables I and II:

TABLE I. HYPERPARAMETER SETTINGS

Model	**Hyperparameter settings**				
	Window Size	***Units***	***Dropout Rate***	***Learning Rate***	***Optimizer***
GRU	30	64	0.3	0.001	Adam
LSTM	30	64	0.3	0.005	Adam
RNN	14	64	0.2	0.001	Adam
1D-CNN	5	64	0.2	0.001	Adam

TABLE II. XGBOOST HYPERPARAMETER SETTINGS

Hyperparameter	**Value**
Learning Rate	0.05
Max Depth	3
n_estimators	200
Subsample	1.0

4.5. Evaluation metrics

In order to scientifically evaluate the performance of the constructed model in the stock price prediction task, two widely used evaluation metrics for regression problems are selected in this paper: Mean Squared Error (MSE) and Coefficient of Determination (R-squared, R^2). Both of them quantitatively evaluate the prediction effect of the model from the perspectives of error magnitude and goodness-of-fit. Since the neural network is deterministic in the prediction phase (model.predict), as long as no inference behaviours with randomness such as Dropout or BatchNorm are added, the result is the same no matter how many times predict is run, so we have not averaged the MSE of the test set.

4.5.1. Mean Squared Error

MSE is a commonly used measure of the prediction error of a regression model, calculated as the average of the squares of the differences between the predicted and true values. A lower MSE value indicates that the model's prediction error is small and is suitable for assessing the model's accuracy in stock price forecasting.The formula for Mean Squared Error is defined as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{17}$$

4.5.2. R-Squared

The R^2 value reflects how well the model fits the data, with values ranging from 0 to 1 [23]. The closer the value is to 1, the better the model's ability to explain stock price fluctuations.R^2 can effectively assess the model's performance in capturing the trend of the data movement. The formula for R^2 is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{18}$$

4.6. Results

In this experiment, the closing price of the stock is used as the target variable, and the time series segmentation strategy of sliding window is adopted to convert continuous historical data into a fixed-length input sequence to adapt to the structural requirements of the time series model. Multiple technical indicators are combined to construct the feature inputs and systematic evaluation is carried out around the model. All models are trained and tested under a unified data processing process and evaluation criteria to ensure fair and scientific comparisons.

The evaluation results of each model on the test set are shown in the Table III, and the evaluation metrics include mean square error (MSE) and coefficient of determination (R²).

TABLE III. EVALUATION METRICS RESULT

Model	Evaluation metrics	
	MSE	***R^2***
XGBoost	5.9835	0.9471
GRU	8.5353	0.9295
RNN	9.2684	0.9200
LSTM	10.4459	0.8965
1D-CNN	11.3570	0.9062

In order to show more visually how well the model predicts on the test set, figures below show how the models predict the closing price of a stock compared to the true value.

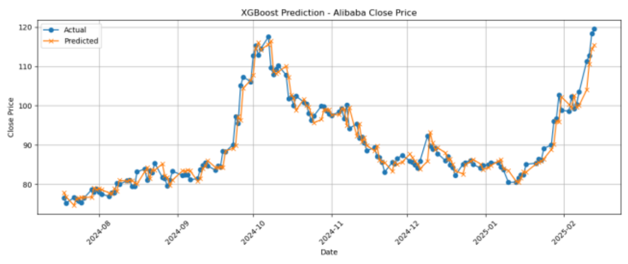

Fig. 3. Predicted vs. Actual Closing Prices of XGBoost

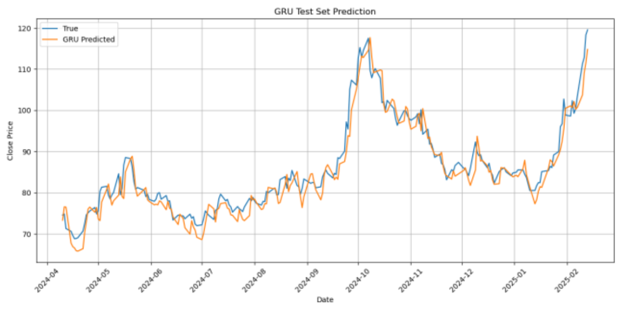

Fig. 4. Predicted vs. Actual Closing Prices of GRU

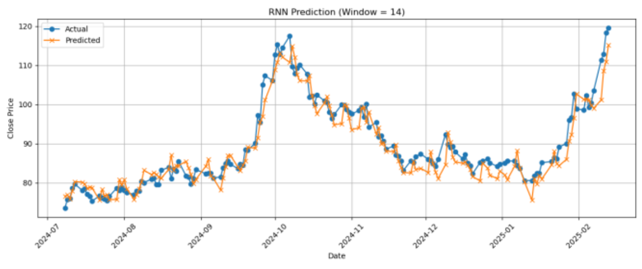

Fig. 5. Predicted vs. Actual Closing Prices of RNN

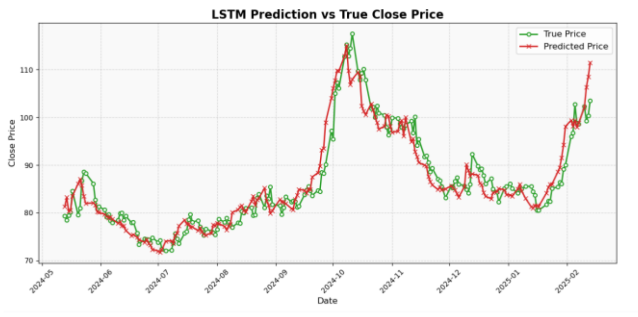

Fig. 6. Predicted vs. Actual Closing Prices of LSTM

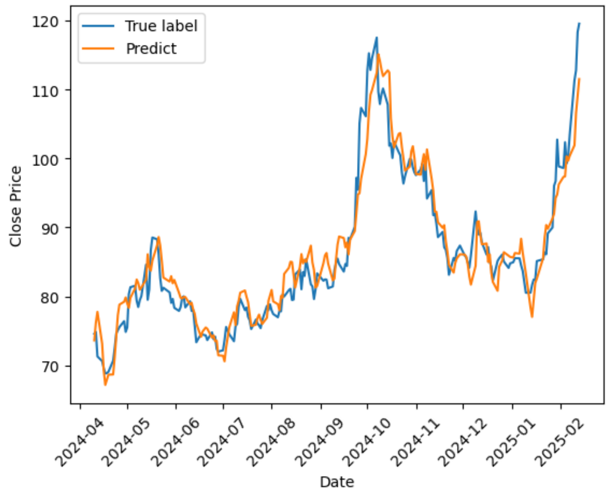

Fig. 7. Predicted vs. Actual Closing Prices of 1D-CNN

5. DISCUSSION

As for the question of which extracted feature played the most critical role in the model's predictive performance, this can already be addressed in the data preprocessing section. According to the feature importance ranking produced by the Random Forest algorithm, the feature High achieved an importance score of 0.72, far exceeding the contributions of other features. Notably, the feature Low also obtained a relatively high importance score of 0.27. If evaluated solely based on importance scores, these two features could account for approximately 99% of the model's predictive power. However, in practice, deep learning models are designed to capture complex relationships among features and can identify hidden patterns and underlying dynamics through self-learning processes. Therefore, relying on only two variables is insufficient. When the dataset is sufficiently large, retaining all relevant features enables the model to automatically learn richer patterns, which is often more effective. Consequently, a total of 17 features were ultimately retained, including several cyclical indicators drawn from previous studies, which are particularly effective in capturing time-series dynamics.

It is evident that the data preprocessing stage involved some redundant steps, representing an area for potential optimization. Another point worth noting is that after feature normalization, some features exhibited noticeable outliers. Unfortunately, we did not have an appropriate approach to address these outliers, which remains a limitation of the preprocessing procedure.

In the modeling and prediction section, short-term (30-day) future prediction using the sliding window approach yields varying results across different deep learning models, Specifically, architectural differences impact their ability to capture temporal dependencies of varying lengths and complexities. According to the results shown in Table III, XGBoost and the LSTM variant GRU achieved relatively high predictive accuracy. By contrast, the MSE of LSTM and RNN was significantly higher than that of the other methods. As mentioned by Selvin, one possible explanation is that LSTM and RNN models rely more heavily on lagged information to predict future outcomes. Moreover, in the process of model selection and design, it is important to recognize that the stock market is a large-scale, highly nonlinear, and dynamic system, where patterns and information continuously evolve over time. This helps explain why RNN and LSTM models may struggle to accurately capture such dynamic variations, ultimately leading to poorer predictive performance [24].

The experimental results further indicate that during hyperparameter tuning, the optimal window size for RNN was found to be 14, while that for CNN was 5. This may partly explain why these two models performed worse than XGBoost, as we standardized the window size to 30 across all models, which may have negatively impacted the performance of RNN and CNN to some extent.

Overall, the experimental findings confirm the feasibility of using historical price data and indicators to predict future stock closing prices and to compare the performance of different models. However, there is still considerable room for improvement in predictive accuracy. Recent literature suggests that researchers are increasingly inclined to combine multiple deep learning models for predictive analysis. In our study, using these deep learning models individually did not yield highly accurate results, indicating that model ensemble approaches represent a promising direction for future improvement.

6. CONCLUSION

In this study, we compared various models, including XGBoost, GRU, LSTM, RNN, and 1D-CNN, for predicting Alibaba's stock closing price. The results indicate that XGBoost performed the best, with superior performance in terms of both MSE and R^2 value. On the other hand, LSTM and GRU performed slightly worse.

XGBoost's strength lies in its powerful nonlinear modeling ability, which effectively captures complex relationships and handles high-dimensional features, making it particularly suitable for small sample data. While deep learning models like LSTM and GRU are capable of capturing long-term dependencies, their performance in this study was hindered by overfitting and longer training times.

XGBoost demonstrates great potential for stock prediction, especially when the sample size is small. Deep learning models are more suitable for large-scale time-series data. However, the study's limitations include a small dataset size, limited feature selection, and the absence of macroeconomic factors.

Future research could explore more advanced models, such as Transformer, expand the dataset, and integrate external data sources (e.g., financial news) to further improve prediction accuracy.

We recommend continuing to use XGBoost as the main model, particularly when the data size is small, while exploring additional features, models, and data sources to enhance prediction accuracy.

REFERENCES

[1] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, “Stock price prediction using LSTM, RNN and CNN-sliding window model,” in 2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017, Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 1643–1647. doi: 10.1109/ICACCI.2017.8126078.

[2] Y. Gao, R. Wang, and E. Zhou, “Stock Prediction Based on Optimized LSTM and GRU Models,” Scientific Programming, vol. 2021, 2021, doi: 10.1155/2021/4055281.

[3] Z. Li, H. Yu, J. Xu, J. Liu, and Y. Mo, “Stock Market Analysis and Prediction Using LSTM: A Case Study on Technology Stocks,” Innovations in Applied Engineering and Technology, pp. 1–6, Nov. 2023, doi: 10.62836/iaet.v2i1.162.

[4] S. Chavhan, P. Raj, P. Raj, A. K. Dutta, and J. J. P. C. Rodrigues, “Deep Learning Approaches for Stock Price Prediction: A Comparative Study of LSTM, RNN, and GRU Models,” in 2024 9th International Conference on Smart and Sustainable Technologies, SpliTech 2024, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.23919/SpliTech61897.2024.10612666.

[5] Y. Zhang, “Stock Price Prediction Method Based on XGboost Algorithm,” 2023, pp. 595–603. doi: 10.2991/978-94-6463-030-5_60.

[6] Z. Hu, Y. Zhao, and M. Khushi, “A survey of forex and stock price prediction using deep learning,” Applied System Innovation, vol. 4, no. 1. MDPI AG, pp. 1–30, Mar. 01, 2021. doi: 10.3390/ASI4010009.

[7] W. Lu, J. Li, J. Wang, and L. Qin, ‘A CNN-BiLSTM-AM method for stock price prediction’, May 01, 2021, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00521-020-05532-z.

[8] F. F. Mojtahedi, N. Yousefpour, S. H. Chow, and M. Cassidy, “Deep Learning for Time Series Forecasting: Review and Applications in Geotechnics and Geosciences,” Archives of Computational Methods in Engineering. Springer Science and Business Media B.V., 2025. doi: 10.1007/s11831-025-10244-5.

[9] N. M. Shahani, M. Kamran, X. Zheng, and C. Liu, “Predictive modeling of drilling rate index using machine learning approaches: LSTM, simple RNN, and RFA,” Petroleum Science and Technology, vol. 40, pp. 534–555, 2022, doi: 10.1080/10916466.2021.2003386.

[10] Y. Bengio, P. Simard, and P. Frasconi, “Learning Long-Term Dependencies with Gradient Descent is Difficult,” IEEE Transactions on Neural Networks, vol. 5, pp. 157–166, 1994, doi: 10.1109/72.279181.

[11] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[12] S. Li and X. Zhang, “Research on orthopedic auxiliary classification and prediction model based on XGBoost algorithm,” Neural Computing and Applications, vol. 32, pp. 1971–1979, Apr. 2020, doi: 10.1007/s00521-019-04378-4.

[13] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM,” Nov. 2019.schitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[14] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[15] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[16] D. Zhao, J. Liu, and X. Wang, “A hybrid LSTM-based model for stock price prediction,” *Neural Computing and Applications*, vol. 33, no. 10, pp. 4921–4933, May 2021, doi: 10.1007/s00521-020-05343-w.

[17] A. Patel and S. Thakkar, “Comparative analysis of LSTM and GRU for stock price prediction using multivariate time series,” *Procedia Computer Science*, vol. 207, pp. 2811–2820, 2022, doi: 10.1016/j.procs.2022.09.321.

[18] M. Rezaei and M. Azizi, “Forecasting stock market trends using LSTM and technical indicators,” *Expert Systems with Applications*, vol. 202, p. 117177, Jan. 2022, doi: 10.1016/j.eswa.2022.117177.

[19] H. H. Htun, M. Biehl, and N. Petkov, ‘Survey of feature selection and extraction techniques for stock market prediction’, Dec. 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1186/s40854-022-00441-7.

[20] P. Chanrungmaneekul, J. Phuangbut, and R. Chaysiri, ‘Bridging Bollinger Bands and Machine Learning in SET100 Stock Trading Strategies’, in *KST 2024 - 16th International Conference on Knowledge and Smart Technology*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 51–56. doi: 10.1109/KST61284.2024.10499648.

[21] R. Iranzad and X. Liu, ‘A review of random forest-based feature selection methods for data science education and applications’, 2024, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s41060-024-00509-w.

[22] Lovrić, M., Milanović, M. and Stamenković, M., 2014. Algoritmic methods for segmentation of time series: An overview. Journal of Contemporary Economic and Business Issues, 1(1), pp.31-53. *(references)*

[23] Chicco, D., Warrens, M.J. and Jurman, G., 2021. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *Peerj computer science*, *7*, p.e623.

[24] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, ‘Stock price prediction using LSTM, RNN and CNN-sliding window model’, in *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 1643–1647. doi: 10.1109/ICACCI.2017.8126078.

CRediT STATEMENT

Ruoqing Gao: Data preprocess, CNN model and report writing (methodology, results related to data analysis and preprocessing, and discussion).

Luxin Xu: RNN model, XGBoost model and report writing (introduction, literature review, part of methodology).

Mengqing Fan: LSTM model, and report writing (abstract, composing the conclusion section).

Jingjie Ouyang: GRU model and report writing (part of methodology, results and evaluations except for data preprocessing).