Advances in Industrial Engineering, June 2024, 58(1): 237-249
DOI: 10.22059/aie.2024.372470.1889

RESEARCH PAPER

Deep Learning Approach for Stock Price Prediction: Comparing Single and Hybrid Models

Asgar Noorbakhsh[1]*, Mostafa Shaygani[2]

[1]*Assistant Professor, Department of Financial Management, Faculty of Management and Accounting, Farabi Campus of University of Tehran, Qom, Iran.*
[2]*M.Sc., Department of Financial Management, Faculty of Management and Accounting, Farabi Campus of University of Tehran, Qom, Iran.*

Received: 11 February 2024, Revised: 16 March 2024, Accepted: 31 March 2024

Abstract

This study utilizes deep-learning models for stock price prediction, focusing on data from five companies listed on the Tehran Stock Exchange over the period 2001 to 2022. Five models are employed, including two hybrid models and three single models. The hybrid CNN-LSTM model serves as the primary model, with its predictive accuracy compared against the other four models. Results indicate that the CNN-LSTM model demonstrates superior performance relative to the others, although the CNN-GRU hybrid model also yields satisfactory results. Interestingly, among the single models, the CNN model surpasses both the LSTM and GRU models, defying initial expectations. The accuracy of the models is notably impacted by factors such as volatility, which increases uncertainty. This research, which exclusively relies on technical indicators, suggests that achieving optimal results hinges not only on selecting the right neural network but also on determining the appropriate number of layers in each model. Overall, the CNN-LSTM model delivers the best performance across four of the five stocks, with the CNN-GRU model slightly outperforming it for one stock. Among the single models, the CNN model consistently outperforms the others.

Keywords:
Deep Learning, Convolutional Neural Network (CNN), Long Short-Term Memory Neural Network (LSTM), Gated Recurrent Unit (GRU), Stock Price Prediction

Introduction

Financial markets are considered one of the most captivating innovations of the present era. In such markets, the success of an investor relies heavily on the quality of information and the speed of decision-making. Therefore, accurate predictions are crucial for investors, as they enable informed decision-making and risk reduction. According to the Efficient Market Hypothesis and the Random Walk Theory in financial markets, no investor can outperform the market's average rate of return using historical and current data, rendering prediction impossible [1]. However, contrary to these theories, proponents of the Efficient Market argue both theoretically and empirically that financial variables, such as stock prices, stock market indices, and prices of financial derivatives, are to some extent predictable [2]. Hence, they have presented and tested models related to stock market behavior, emphasizing the importance of attention and interest.

Among various modeling approaches, artificial neural networks have gained popularity due

---

* Corresponding author: (Asgar Noorbakhsh)
Email: anoorbakhsh@ut.ac.ir

to their ability to handle the irregular, unstable, noisy, and non-parametric nature of financial markets. Their intrinsic capability to accurately estimate non-linear relationships, conduct multi-variable analyses without predefined assumptions, and facilitate generalization has contributed to their relative success in modeling and predicting various segments of financial markets [3].

The movement of stock prices is consistently identified as a significant challenge in the economic domain. Stock prices are influenced by various internal and external factors, including the domestic and international economic environment, global conditions, industry outlook, financial data of listed companies, and stock market performance [4].

Traditional analytical methods in economics and finance, primarily based on fundamental and technical analysis, have been widely used. Fundamental analysis emphasizes the intrinsic value of stocks and qualitatively analyzes external factors affecting stocks, such as interest rates, exchange rates, inflation, industry policies, and financial affairs of listed companies, while technical analysis primarily focuses on the direction of stock prices, transaction volumes, and psychological expectations of investors. Currently, fundamental and technical analysis are the most common methods used by many organizations and individual investors [5].

Artificial neural networks, with their ability to analyze multi-variable non-linear data and facilitate the generalization of advanced learning algorithms, have become a versatile and dominant tool in financial and economic studies. Given that the performance of neural networks is directly related to the quality of features, and placing an excessive number of independent variables in regression is one of the biggest obstacles in their scientific application, feature selection plays a crucial role in prediction [6].

Since predicting and modeling prices has always been a challenging and inseparable task for investor analyses, and the use of neural networks in financial matters has seen a growing expansion, employing various neural network models, especially composite models, can create powerful models in this field. Predicting prices using new models can enhance accuracy in analysis and prediction, thereby improving investment performance. Additionally, increased accuracy in price prediction can assist regulatory organizations in identifying illegal price movements [7]. In this research, we aimed to investigate the prediction accuracy of various deep learning models in the Tehran Stock Exchange. The use of five deep learning models, including two hybrid models and three single models, as well as the use of various technical variables to accurately prediction of prices in the stock exchange, are among the innovations of this research.

Literature Review

Given the significant importance of price prediction in the stock market, investors utilize various methods for this purpose. These methods can broadly be categorized into two approaches: some are retrospective, attempting to better predict markets by following price patterns, while others are prospective, aiming to analyze various scenarios.

Jian Cao et al. [8]conducted a study titled " Financial time series forecasting model based on CEEMDAN and LSTM " on four indices: DAX[†], HSI[‡], S&P500, and SSE[§], using daily closing prices for prediction. The results of this composite model indicate lower errors compared to LSTM, SVM, CEEMDAN-SVM, and CEEMDAN-MLP models based on MAE, MAPE, and RMSE metrics. The model's optimization capability is highlighted as it only uses closing prices as input, but the number of inputs can be increased by incorporating the highest and lowest

[†] Dax performance-index
[‡] Hang Seng Index
[§] SSE Composite index

prices, trading volume, and other price-related features.

Hiransha et al. [9] in their article "Stock Market Prediction Using Deep Learning Models" employed four deep learning architectures, namely MLP, RNN, LSTM, and CNN, to predict stock prices from two different markets, India and the United States. The study favored the CNN model for achieving better performance compared to other models.

Omkar Khare et al. [10] presented "Short-Term Stock Price Prediction Using Deep Learning," using one year of minute-level data from ten companies on the New York Stock Exchange. The study employed a feedforward neural network (Multi-Layer Perceptron) and a short-term memory neural network (LSTM) for comparison, showing that the multi-layer perceptron outperformed the short-term memory neural network.

Chi-Yu Lee et al. [11] discussed "Stock Price Prediction with Financial News Based on Recurrent Convolutional Neural Networks," utilizing a combined model of a convolutional neural network and a short-term memory neural network. The study demonstrated that the combined model had better performance than individual models, especially the short-term memory neural network.

Peter T. Yamak et al. [12] compared ARIMA, LSTM, and GRU models for time series prediction in their article "A Comparison of ARIMA, LSTM, and GRU Models for Time Series Prediction." The study, focusing on digital currency data, revealed that the ARIMA model outperformed the other models, with GRU performing better than LSTM.

Seyedra Mahatab et al. [13] conducted research titled "Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models," utilizing NIFTY50 index data from the National Stock Exchange of India. The study employed four regression models based on deep learning using LSTM, demonstrating that the single-variable LSTM model using one-week historical input for predicting the future opening price achieved the highest accuracy.

Nan Jing et al. [14] proposed a model titled "A Hybrid Model Integrating Deep Learning with Investor Sentiment Analysis for Stock Price Prediction." This combined approach used a convolutional neural network for sentiment classification extracted from a large stock forum. The study integrated sentiment analysis results with a long short-term memory neural network for technical analysis of stock market indicators. Real experiments on six key industries in the Shanghai Stock Exchange validated the effectiveness and applicability of the proposed model.

Jingyi Shen et al. [15] conducted research titled " Short-term stock market price trend prediction using a comprehensive deep learning system". They conducted comprehensive evaluations on frequently used machine learning models and conclude that their proposed solution outperforms due to the comprehensive feature engineering that they built. The system achieves overall high accuracy for stock market trend prediction.

These studies showcase the diversity of approaches, combining various techniques such as deep learning, sentiment analysis, and technical analysis to enhance stock price prediction accuracy.

Data and Methodology

Data

This research delves into the examination of results obtained from five deep learning models. The data for this study spans 21 years (from 2001 to 2022) and pertains to the companies Iran Khodro, Naft-e- Behran, Cimane Tehran, Tose'e Ma'aden Rooye Iran, and Alborz Darou.

The methodology employed in this research solely revolves around historical and technical features. The utilized features include the opening price, the lowest price, the highest price, the closing price, trading volume, transaction value, price changes (closing price), and percentage changes in price (closing price).

Long Short -Term Memory (LSTM)

The short-term memory neural network, first introduced by Hochreiter and Schmidhuber in 1997, has undergone significant improvements by numerous individuals in subsequent years. Although the LSTM network is relatively old, it is still widely used across a spectrum of problems and continues to enjoy high popularity [16].

The short-term memory neural network is a type of deep learning architecture specifically designed for modeling sequential data, such as texts, speech, time-series signals, and other types of sequential data. The primary components of a short-term memory neural network unit include:

1. Short-Term Memory: In this component, information that is not needed for final decision-making steps from past sequence steps is discarded.
2. Long-Term Memory: This component retains important and meaningful information from the sequence so that it can be utilized in future steps.
3. Gates: Gates, such as the input gate, forget gate, and output gate, are present in an LSTM unit. These gates control which information enters the short-term memory, which information is forgotten, and which information is read from the long-term memory.
4. Activation Functions: Activation functions, such as hyperbolic tangent (tanh), are used to calculate and control the flow of information within the LSTM unit.

Fig. 1 shows the LSTM structure diagram (Fig. 1).

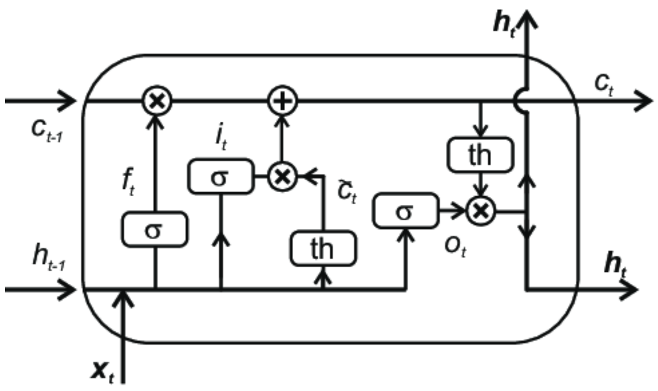

Fig 1. LSTM structure diagram.

Eq. 1 to 7, respectively, represent the input gate, forget gate, output gate, memory cell candidate, memory cell, hidden state, and cell output in the architecture of an LSTM model (Eq. 1, 2, 3, 4, 5, 6 and 7).

In the equations below, the lowercase variables represent vectors. Matrices $\mathbf{W_q}$ and $\mathbf{U_q}$ contain, respectively, the weights of the input and recurrent connections, where the subscript $_q$ can either be the input gate i, output gate o, the forget gate f or the memory cell c, depending on the activation being calculated.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$
$$\tilde{c}_t = tanh(W x_t + U h_{t-1} + b) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$
$$y_t = h_t \quad (7)$$

Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) were first developed around the 1980s and found application primarily in recognizing handwritten digits. Initially, their main utility was in postal services for reading postal codes, PIN codes, and similar applications. An important note regarding any deep learning model is the substantial need for a large volume of training data and significant computational resources. In the early days, CNNs were constrained to postal applications, but with the advent of powerful hardware and increased computational capabilities of machines, they gained popularity, especially in computer vision [17].

Convolutional Neural Networks (CNNs) represent one of the most crucial architectures in deep learning, inspired by the pattern recognition in the visual cortex of the brain. Initially patterned after the Visual Cortex, individual neurons in small sections work independently, and their combination forms interconnected networks that contribute to perceiving a specific area. One-dimensional CNNs are a type of deep neural network architecture designed for processing one-dimensional sequential data, such as time-series signals and temporal data. They are also employed in certain cases for text processing, aiding in the detection of patterns and significant features in sequential data.

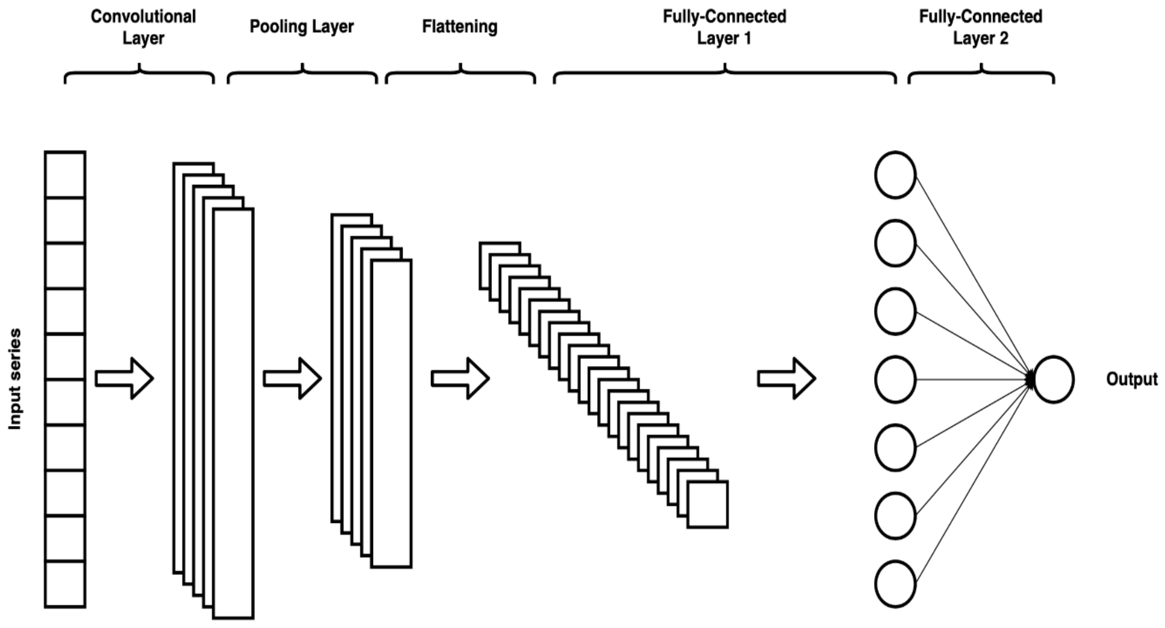

Fig 2. CNN structure diagram.

The key components of a one-dimensional Convolutional Neural Network include (Fig. 2):
1. One-Dimensional Convolutional Layer: This layer employs one or more one-dimensional filters to scan input data. The filters have a specific length and width, move across the data, and perform the convolution operation. This operation aids in detecting patterns and significant features in the data.
2. Pooling Layers: Similar to pooling layers in two-dimensional convolutional neural networks, pooling layers in one-dimensional convolutional neural networks are used to reduce dimensions and the number of features. These layers play a role in down-sampling and simplifying the representation of the input data.

Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride, and padding size:

The depth of the output volume controls the number of neurons in a layer that connect to the same region of the input volume. These neurons learn to activate for different features in the input.

Stride controls how depth columns around the width and height are allocated. If the stride is 1, then we move the filters one pixel at a time. This leads to heavily overlapping receptive fields between the columns, and to large output volumes. For any integer S>0, a stride S means that the filter is translated S units at a time per output. In practice, S>0 is rare. A greater stride means smaller overlap of receptive fields and smaller spatial dimensions of the output volume.

Sometimes, it is convenient to pad the input with zeros (or other values, such as the average of the region) on the border of the input volume. The size of this padding is a third hyperparameter. Padding provides control of the output volume's spatial size. In particular, sometimes it is desirable to exactly preserve the spatial size of the input volume, this is commonly referred to as "same" padding. The spatial size of the output volume is a function of the input volume size ***W*** the kernel field size ***K*** of the convolutional layer neurons, the stride ***S***, and the amount of zero padding ***P*** on the border. The number of neurons that "fit" in a given volume is then (Eq. 8):

$$\frac{W-K+2P}{S}+1 \tag{8}$$

If this number is not an integer, then the strides are incorrect and the neurons cannot be tiled to fit across the input volume in a symmetric way.

In general, setting zero padding to be $P=\frac{(K-1)}{2}$ when the stride is $S=1$ ensures that the input volume and output volume will have the same size spatially. However, it is not always completely necessary to use all of the neurons of the previous layer.

Gated Recurrent Unit

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a gating mechanism to input or forget certain features, but lacks a context vector or output gate, resulting in fewer parameters than LSTM. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. There are several variations on the full gated unit, with gating done using the previous hidden state and the bias in various combinations, and a simplified form called minimal gated unit [18]. Fig. 3 shows the GRU structure diagram (Fig. 3).

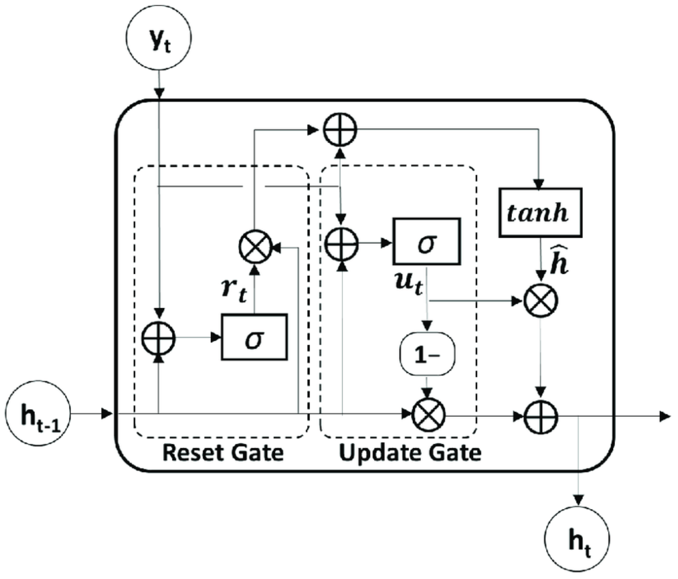

Fig 3. GRU structure diagram.

Eq. 9 to 12, respectively, represent the update gate vector, reset gate vector, candidate activation vector, output vector, in the architecture of an GRU model (Eq. 9, 10, 11 and 12).

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{9}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{10}$$
$$\hat{h}_t = tanh(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \tag{11}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{12}$$

1. Update Gate (z): The update gate determines the extent to which the previous hidden state should be combined with the hidden state of the new candidate. This gate assists the model in deciding whether to update or ignore a portion of the past information.
2. Reset Gate (r): The reset gate decides which portions of the previous hidden state should be discarded and forgotten in the computation of the new candidate hidden state.
3. Candidate Hidden State ($\hat{h}$): This candidate hidden state, calculated based on the input data and the previous hidden state, and created considering the reset gate, specifies the new hidden state. It combines information from the current input and the past hidden state.

Research Process

In all three models, the data is initially entered into the Python software in a specific format.

The first step after entering the data is the data scaling process, which can be done in two ways: normalization and standardization. In this research, standardization has been employed. The reason for choosing this method is the existence of features with non-uniform units.

Normalization Method: Min-Max scaling, or Min-Max Scaler, is a common method in normalizing data. It scales the data to a specified range between two minimum and maximum values. This method ensures that all data are scaled to a specific range, such as [0, 1].

Standardization Method: Standardization is another method for normalizing data, transforming them into a distribution with a mean of zero and a variance of one. In this method, the mean and variance of the data are used to scale and transform the data units. This process helps machine learning models to handle data more effectively and accurately.

$$y_i = \frac{x_i - \bar{x}}{\sigma} \tag{13}$$

In Eq. 13, y_i represents the standardized value, x_i denotes the input data, $\bar{x}$ signifies the mean of the input data, and σ indicates the standard deviation (Fig. 3).

The second stage involves dividing the data into two categories: Train and Test. In this research, for all three models, we assume 80% of the data as training data and 20% as testing data. Due to the limited number of data points, testing data has also been used in evaluating the model's performance.

The third stage focuses on transforming time series data into a supervised learning format. Converting time series data into a supervised learning format is a crucial step for using regression models in predicting and analyzing time series data. In this format, time series data is transformed into pairs of input and corresponding output. In other words, past samples are used to predict future values. In this research, the following command is employed to transform the data from time series to supervised learning format. This process is typically performed using time windows (time_step), where previous features serve as input, and the value at the next time step serves as the output.

First, we determine the time step (time_step). The time step specifies how many previous time steps to use for predicting the label. In this case, a time step of 22 is employed, equivalent to one working month. Subsequently, we construct the matrices x_train, y_train, x_test, and y_test.

The fourth stage involves fitting and evaluating the model. Fitting the model refers to the alignment of the model with the training data. This alignment can occur in three different ways: good fit, overfitting, and underfitting. These three conditions indicate the quality of the model's training concerning both the training data and new data.

- **Good Fitting**: In this situation, the model accurately responds to the training data and also demonstrates good capability in estimating new data. The model neither responds excessively to the training data nor to an extent that it fails to extract general information about the problem.
- **Overfitting**: In this scenario, the model responds excessively to the training data and has poor ability to estimate new data. The model becomes sensitive to small details and noise in the training data, preserving the representation of the training data to a considerable extent. However, it may provide incorrect results when faced with new data.
- **Underfitting**: In this condition, the model neither accurately responds to the training data nor possesses good capability in estimating new data. The model responds to the simplicity and discontinuity of the data and lacks the ability to generalize to new data.

To achieve good fitting, it is necessary for the model to have an appropriate architecture, utilize a suitable cost function, and determine an optimal number of training epochs.

To address the issue of overfitting, various methods can be employed, and in this context, we are utilizing the Regularization method.

Regularization is a general technique in machine learning applied to reduce both overfitting and underfitting of models, including neural networks. This technique, by adding regularization terms to the model's cost function, controls the impact of weights, helping the model learn important and meaningful information from the training data.

There are two main types of regularization:

1. **Lasso Regularization**: In Lasso regularization, a regularization term is added to the cost function equal to the absolute values of the weights. This encourages weights with small values to approach zero, resulting in the removal of features associated with those weights. This process helps the model select only important and meaningful features.
2. **Ridge Regularization**: In Ridge regularization, a regularization term is added to the cost function equal to the sum of the squares of the weights. This causes the weights to be slightly closer to smaller values, enhancing the model's inclination to learn important and more generalizable features.

The application of regularization involves adding a regularization term to the main cost function. The regularization factor, determined as a hyperparameter, is a parameter that specifies how much impact regularization has on the model. By tuning this hyperparameter, the degree of regularization influence on the model can be adjusted.

Eq. 14 and Eq. 15 represent Lasso and Ridge respectively (Fig. 14 and 15).

$$Loss = Error(y, \hat{y}) + \lambda \sum_{j=1}^{p} |\beta_j| \tag{14}$$

$$Loss = Error(y, \hat{y}) + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{15}$$

In this research, a combination of both Lasso and Ridge regularization is employed, known as Elastic Net.

The Lasso method is not highly sensitive to outliers and tends to reduce some weights to exactly zero. However, certain weights may still remain large. On the other hand, the Ridge method, being more sensitive to outliers, shrinks the weights, but they never become exactly zero.

Elastic Net combines these two regularization techniques, aiming to address the limitations of both.

Evaluation

In order to evaluate and measure the amount of error between the actual value and the predicted value, three error criteria of mean absolute error, root mean square error and coefficient of determination have been used.

The mean absolute error is a measurement criterion that is used in various fields, especially in statistics and machine learning, to measure the amount of difference between the predicted values and the actual values of a variable. The mean absolute error is less sensitive to large and outlier errors. This means that if a number of predictions are in error, the mean absolute error is less affected and changes little. The mean absolute error is easy to interpret.

Eq. 16 shows how to calculate the mean absolute error (Fig. 16).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - ty_i| \tag{16}$$

The Root Mean Square Error (RMSE), like the Mean Absolute Error, is a measurement metric used across various fields, including statistics and machine learning, to quantify the difference between predicted and actual values. It is succinctly referred to as the square root of the mean squared error.

As a metric that calculates differences in a squared manner, RMSE places importance on both larger and smaller errors. This interpretation of accuracy implies that even small differences have a significant impact on the final results. When confronted with larger errors, RMSE assigns greater importance to them compared to Mean Absolute Error.

In summary, RMSE provides a comprehensive assessment of the model's predictive performance, considering both the magnitude and squared differences between predicted and actual values.

Eq. 17 shows how to calculate the root mean square error (Fig. 17).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{17}$$

The coefficient of determination, also known as R-squared, is a metric that measures the extent to which the dependent variable's variation is explained by the independent variables in a model. This metric is commonly used in statistics and machine learning to indicate how much of the variability in the dependent variable is justified by changes in the independent variables. The coefficient of determination is equal to the square of the coefficient of correlation between the dependent variable and the independent variables. In essence, R-squared illustrates the proportion of the dependent variable's changes that can be attributed to changes in the independent variables.

The value of the coefficient of determination ranges from 0 to 1, where 0 indicates no justification for changes in the dependent variable by the independent variables, and 1 signifies complete justification of changes. The coefficient of determination provides insight into how well your model explains the actual variations. If the R-squared value is high, it indicates that your model aligns well with the data.

Eq. 18 shows how to calculate the coefficient of determination (Fig. 18).

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

Result

In Table 1, the results and performance of all five models for each of the five companies are observable. The results from the evaluation metrics illustrate how the models performed.

The results indicate that the combined CNN-LSTM model has shown better and more accurate performance compared to other models. Only in the case of the Khodro symbol, the

CNN-GRU model performed slightly better than the CNN-LSTM model. Among the single models, the CNN model, despite initial predictions favoring the other two models (LSTM and GRU), demonstrated better performance. It is also noteworthy in this research that the single CNN model shows performance comparable to that of the combined models in some symbols, indicating its predictive strength.

Table 1. Performance of the Models.

Ticker	Model	RMSE	MAE	R-Square
Shebehran	CNN	2596	1050	98.79
	LSTM	6283	2344	92.89
	GRU	5015	2024	95.47
	CNN-GRU	2489	964	98.88
	CNN-LSTM	2199	818	99.13
Dalber	CNN	1276	696	97.24
	LSTM	3381	1694	80.63
	GRU	1657	803	95.35
	CNN-GRU	841	470	98.80
	CNN-LSTM	623	409	99.34
Khodro	CNN	458	152	96.05
	LSTM	592	304	94.10
	GRU	536	213	95.17
	CNN-GRU	396	125	97.36
	CNN-LSTM	417	125	97.08
Kerooy	CNN	2604	1447	96.65
	LSTM	6432	2903	79.53
	GRU	7362	3684	73.18
	CNN-GRU	2889	1255	95.87
	CNN-LSTM	1733	1062	98.51
Setran	CNN	1221	731	99.45
	LSTM	3068	1158	96.52
	GRU	2572	1045	97.55
	CNN-GRU	1310	722	99.37
	CNN-LSTM	1193	667	99.47

The performance results of the models can be seen in Figures 4 to 8 (Fig. 4, 5, 6, 7 and 8).

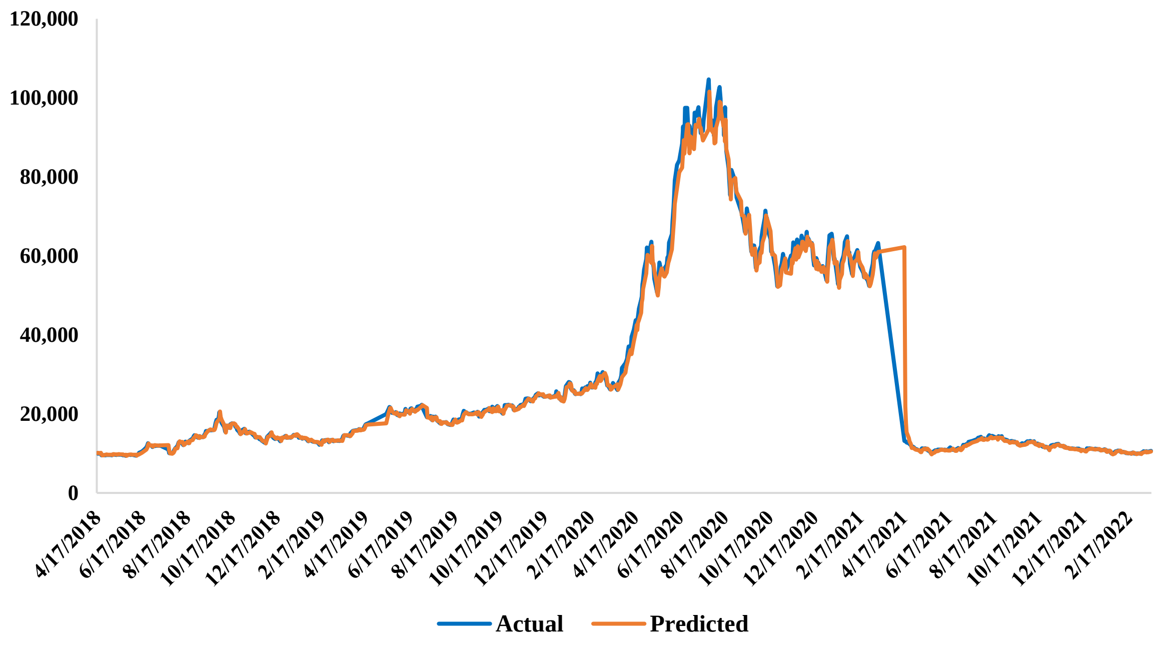

Fig 4. Shebehran (CNN-LSTM).

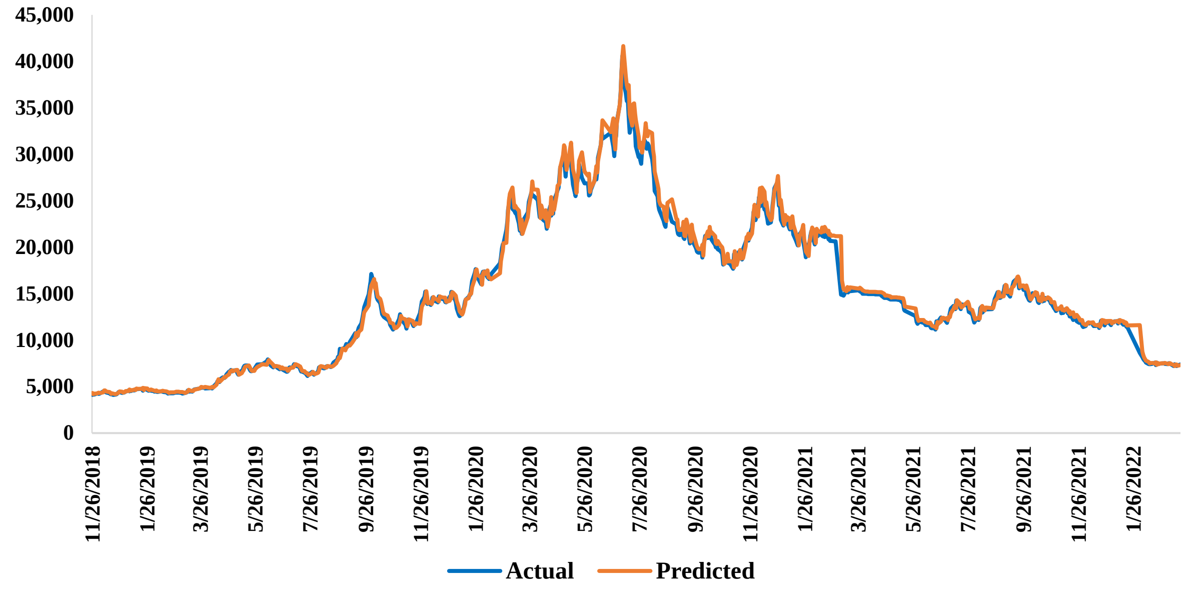

Fig 5. Dalber (CNN-LSTM).

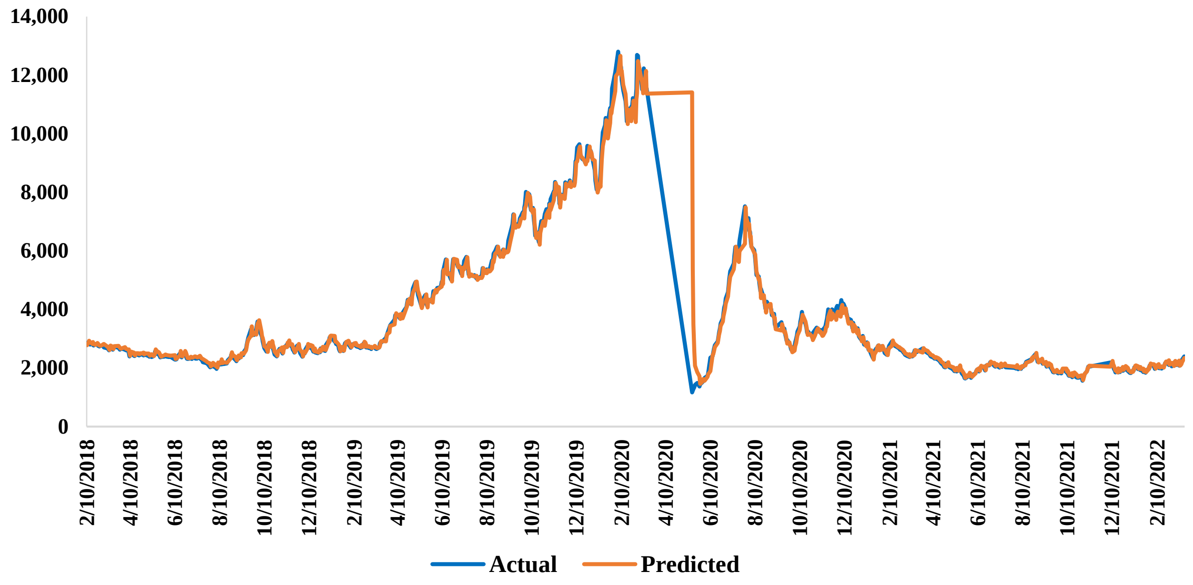

Fig 6. Khodro (CNN-LSTM).

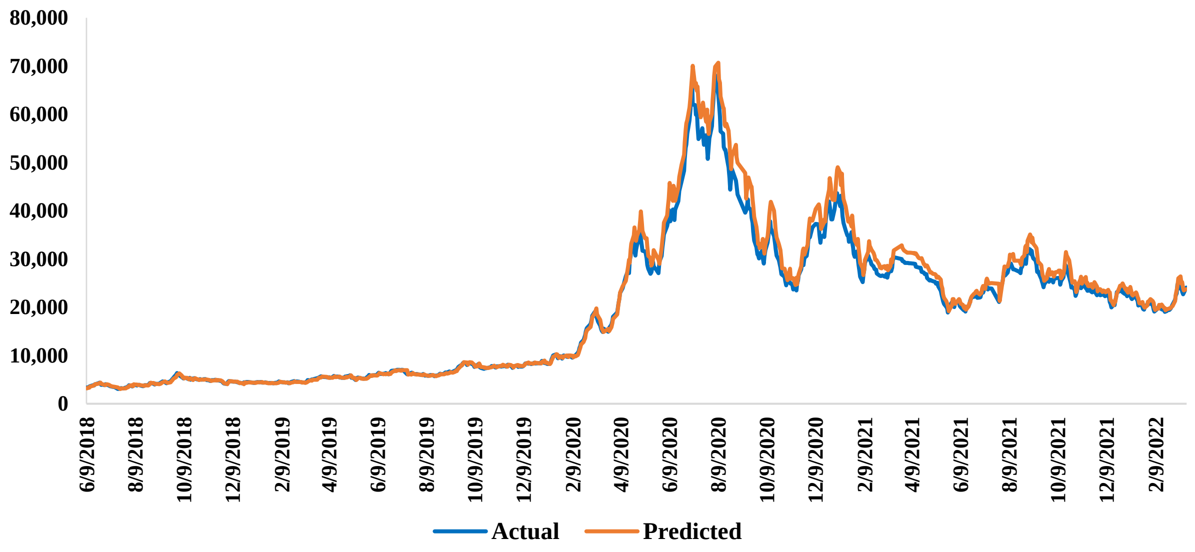

Fig 7. Kerooy (CNN-LSTM).

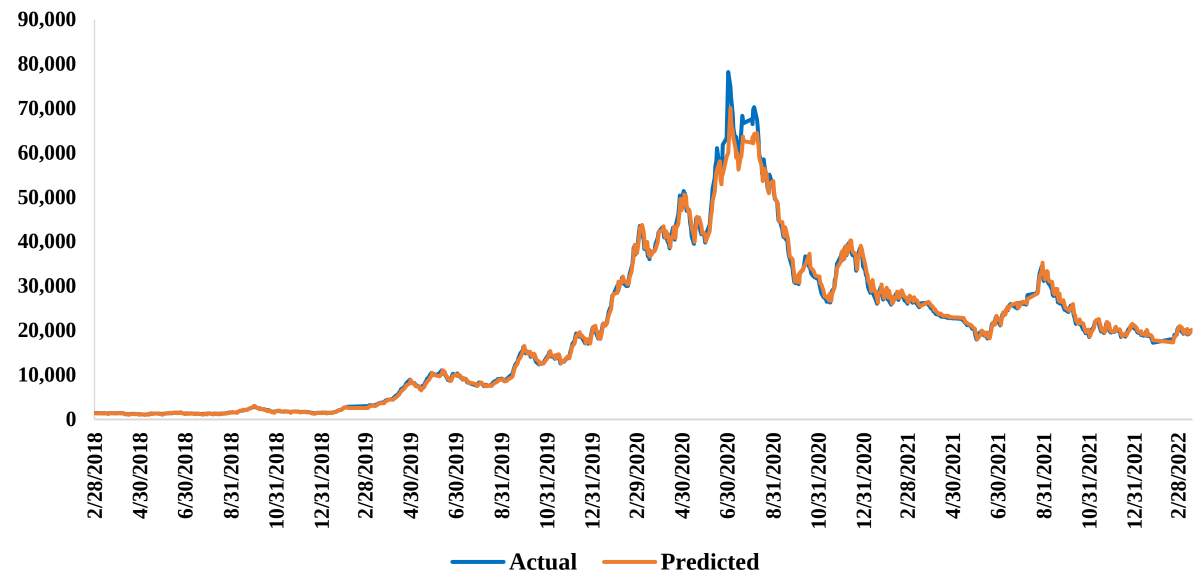

Fig 8. Setran (CNN-LSTM).

The model introduced in this research exhibits significantly higher accuracy in predicting various cycles of economic downturns and upturns compared to other models. The obtained coefficient of determination in the model indicates that desirable outcomes in stock price prediction can be achieved using technical variables. As mentioned, deep learning models require a large amount of data to enhance prediction accuracy, but an increase in the number of independent variables may lead to overfitting issues. It is important to note that the presence of market volatility diminishes the predictive power of these models.

Conclusion

According to the literature review, neural networks and deep learning play a very important role in stock price prediction. These technologies leverage the analysis of financial and stock market data to help investors make better decisions and make more accurate forecasts.

The results of studies in this field show that neural networks and deep learning can provide accurate predictions about stock prices by analyzing historical data and financial indicators. This high accuracy helps investors make better investment decisions. Considering that price forecasting and modeling have always been challenging and an integral part of investors' analysis, and on the other hand, the use of neural networks in financial affairs has been expanding day by day, the use of different neural network models, especially hybrid models can create powerful models in every aspect in this field.

Price forecasting using new models can increase accuracy in analysis and forecasting, which can improve investment performance. The purpose of this research is to present a hybrid model using neural network called CNN and STLM neural network. The main question and problem that we seek to answer during this research is whether is the hybrid model capable of predicting the Tehran Stock Exchange market or not?

Considering the continuous advancements in deep learning models for predicting various datasets, including financial data, we have endeavored in this research to investigate some deep learning models for estimation and prediction in the Tehran Stock Exchange. Given the constraints and specific characteristics of the Tehran Stock Exchange, such as market volatility, researchers are motivated to explore and forecast this market using machine learning and deep learning models.

The results obtained from this research indicate that deep learning models, when selecting features (independent variables) capable of capturing the most significant interpretations of the causes of market upturns and downturns during periods of market expansion and recession, exhibit acceptable accuracy and prediction capabilities.

For future research, two aspects could be pursued and examined. First, the model's capability and accuracy in predicting longer time periods, and second, incorporating additional independent variables, such as macroeconomic and financial corporate variables, for comparison and analysis of results.

References

[1] Oussar, Y., & Dreyfus, G. (2000). Initialization by selection for wavelet network training. Neurocomputing, 34 (1): 131-143.

[2] Chong, E., Han, Ch., & Park F. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems With Applications, (83): 187-205.

[3] Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. Expert Systems with Applications, 36 (8): 10896-10904.

[4] Vanaga, R., & Sloka, B. (2020). Financial and capital market commission financing: Aspects and challenges. Journal of Logistics, 7 (1): 17-30.

[5] Sousa, J., Montevechi, J., & Miranda, R. (2019). Economic lot-size using machine learning, parallelism, metaheuristic, and simulation. Journal of Logistics, Informatics, and Service Science, 18 (2): 205-216.

[6] Zhang, L., & Kim, H. (2020). The influence of financial service characteristics on use intention through customer satisfaction with mobile fintech. Journal of System and Management Sciences, 10 (2): 82-94.

[7] Noorbakhsh, A., Soltani, R., & Asadi Mafi, M. (2021). "Supply Chain and Predictability of Return". Advances in Industrial Engineering, 55(4), 323-333. doi: 10.22059/aie.2021.330003.1803

[8] Cao, Jian & Li, Zhi & Li, Jian, (2019). "Financial time series forecasting model based on CEEMDAN and LSTM," Physica A: Statistical Mechanics and its Applications, Elsevier, vol. 519(C), pages 127-139.

[9] Hiransha M, Gopalakrishnan E.A., Vijay Krishna Menon, Soman K.P. (2018), "NSE Stock Market Prediction Using Deep-Learning Models", Procedia Computer Science, Volume 132, Pages 1351-1362.

[10] K. Khare, O. Darekar, P. Gupta and V. Z. Attar, (2017), "Short term stock price prediction using deep learning," 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2017, pp. 482-486, doi: 10.1109/RTEICT.2017.8256643.

[11] C. -Y. Lee and V. -W. Soo, (2017). "Predict Stock Price with Financial News Based on Recurrent Convolutional Neural Networks", Conference on Technologies and Applications of Artificial Intelligence (TAAI), Taipei, Taiwan, 2017, pp. 160-165, doi: 10.1109/TAAI.2017.27.

[12] Peter T. Yamak, Li Yujian, and Pius K. Gadosey, (2020). "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting", In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence (ACAI '19). Association for Computing Machinery, New York, NY, USA, 49–55.

[13] Mehtab, S., Sen, J., Dutta, A. (2021). "Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models. In: Thampi, S.M., Piramuthu, S., Li, KC., Berretti, S., Wozniak, M., Singh, D. (eds) Machine Learning and Metaheuristics Algorithms, and Applications". SoMMA 2020. Communications in Computer and Information Science, vol 1366. Springer, Singapore. https://doi.org/10.1007/978-981-16-0419-5_8.

[14] Jing, Nan & Wu, Zhao & Wang, Hefei. (2021). A Hybrid Model Integrating Deep Learning with Investor Sentiment Analysis for Stock Price Prediction. Expert Systems with Applications. 178. 115019. 10.1016/j.eswa.2021.115019.

[15] Shen, J., Shafiq, M.O. Short-term stock market price trend prediction using a comprehensive deep learning system. J Big Data 7, 66 (2020). https://doi.org/10.1186/s40537-020-00333-6.

[16] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[17] Rawat, Waseem & Wang, Zenghui. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation. 29. 1-98. 10.1162/NECO_a_00990.

[18] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014 Workshop on Deep Learning, December 2014.