Chapter 13

LSTM-Based Multivariate Deep Neural Networks for Stock Price Forecasting

Selim Serin[1]

Gülder Kemalbay[2]

Abstract

The inherent complexity and dynamic nature of financial markets present substantial challenges for accurately forecasting price movements. Traditional forecasting models often struggle to capture these intricacies, leading to suboptimal predictive performance. In this context, deep learning offers a promising alternative due to its ability to process and analyze large volumes of financial data. This chapter aims to forecast the one-day-ahead price of a Turkish stock over the period from July 30, 2007, to October 11, 2023, using multivariate inputs, including the closing price of the XU030 index, Brent crude oil price, gold price per ounce, the difference between intraday high and low prices, the difference between intraday closing and opening prices, and the ratio of the 14-day to 60-day moving averages. All price-based variables are denominated in U.S. dollars using the TRY/USD exchange rate to ensure consistency.

Using this multivariate dataset, the study employs a deep neural network framework, incorporating Vanilla Long Short-Term Memory (LSTM), Stacked LSTM, Bidirectional LSTM (Bi-LSTM), Convolutional Neural Network-LSTM (CNN-LSTM), and Convolutional LSTM (ConvLSTM) models, with lagged input variables spanning up to five days. A review of the literature highlights that most existing studies focus primarily on univariate forecasting of stock prices or market indices, while fewer address multivariate time series using deep neural networks. The results indicate that, among the models tested, the Vanilla LSTM achieves the highest accuracy in one-day-ahead price forecasting, with a Mean Absolute Percentage Error (MAPE) of 2.20%, demonstrating its robustness in handling multivariate data.

1 M.Sc., Yildiz Technical University, selim.serin96@gmail.com, 0009-0004-4124-1262

2 Assoc. Prof., Yildiz Technical University, kemalbay@yildiz.edu.tr, 0000-0001-9126-8907

https://doi.org/10.58830/ozgur.pub518.c2135

1. Introduction

Artificial neural networks (ANNs) became popular tools in the late 1980s, introducing new approaches to tackling complex pattern recognition tasks. However, these early networks faced challenges due to their intricate architectures and the extensive tuning they required. Poorly optimized networks often underperformed compared to other methods, highlighting the need for improvements in design and functionality. By the 2010s, neural networks re-emerged as deep learning models, incorporating advancements that revolutionized their applicability across diverse fields such as image and video classification, speech recognition, and text modeling (Hinton et al., 2006), (LeCun et al., 2015), (James et al., 2021).

Deep learning architectures, especially recurrent and convolutional neural networks, have since demonstrated considerable success in handling complex, multivariate, and nonlinear time series forecasting tasks. These models offer automatic feature-learning capabilities, supporting both multivariable inputs and outputs, which is essential for capturing temporal dependencies and intricate relationships in time series data (Hewamalage et al.,2021). Traditional time series methods, such as Autoregressive Integrated Moving Average (ARIMA), have long been used in forecasting but are limited by their linear framework, which often underperforms in complex, dynamic data environments where nonlinear dependencies are crucial (Wu et al., 2023).

Among deep learning models, LSTM networks are particularly adept at modeling sequential data with temporal dependencies, making them highly effective for time series forecasting in financial markets. The current work leverages these strengths by using LSTM models to forecast the one-day-ahead price of a stock on the Turkish stock market, employing multivariate inputs including the XU030 index closing price, Brent oil price, gold price, and relevant technical indicators.

The objective of this chapter is to forecast the next-day closing price of Türkiye Petrol Rafinerileri Anonim Şirketi (TUPRS) stock by utilizing multivariate inputs through advanced deep neural network models, including Vanilla LSTM, Stacked LSTM, Bi-LSTM, CNN-LSTM, and ConvLSTM. Recognizing that much of the existing literature focuses predominantly on univariate stock price or market index forecasting, this research aims to address a gap by implementing multivariate time series data. Using the previous five trading days' closing values, the study evaluates and compares the performance of each algorithm. The research hypothesis posits that, with multivariate inputs and deep neural network architectures such as LSTM variants, the model can forecast the next-day stock price with a MAPE of no more than 10%.

1.1. Literature Review

One of the foremost challenges in finance is predicting future price movements based on historical data, a task that has garnered increased attention with the advancements in deep learning and machine learning models. The collaboration between today's financial and information sectors is aimed at accurately forecasting asset prices and evaluating the relationship between risk and expected returns. Modern technology facilitates the development of models that can uncover complex, non-linear relationships in financial datasets and make highly accurate predictions using training data.

Financial forecasting through neural networks and deep learning has gained traction in recent years, with researchers exploring various models for enhanced predictive accuracy. Early work by *Adem et al.* demonstrated the potential of Multilayer Perceptrons (MLP) in forecasting the XU100 index, achieving a 96.92% accuracy by analyzing daily closing prices of international stock indices. Their study established ANNs as powerful tools for financial data analysis in Turkey's stock markets (Adem et al., 2016). Internationally, *Fischer and Krauss* applied LSTM networks to S&P 500 data, achieving notable accuracy over traditional models and emphasizing LSTM's capacity to process sequential data in financial forecasting (Fischer and Krauss, 2018).

By 2018, neural networks had extended into diverse sectors. *Yorulmus et al.* applied LSTM models to intraday electricity price predictions in Turkey, achieving strong results with Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) values of 17.2 and 0.24, respectively, indicating the model's suitability for high-frequency time series in dynamic markets (Yorulmuş et al., 2018). In the energy sector, further research by *Altunkaya and Yılmaz* developed LSTM, RNN, and GRU models to forecast hourly energy demand, with LSTM proving most effective among these models, showing the growing applicability of deep learning in demand forecasting (Altunkaya and Yılmaz., 2020).

During this period, applications of LSTM in financial markets also gained attention. For example, *Alpay* used LSTM to predict the USD/TRY exchange rate with data spanning from 2000 to 2017, demonstrating the model's ability to capture essential temporal dependencies in financial time series (Alpay, 2020). Another study by *Nelson et al.* explored LSTM's effectiveness for Bitcoin price prediction, comparing its performance to ARIMA and Bayesian models and finding superior results with LSTM for capturing long-term dependencies in cryptocurrency (Nelson et al., 2017). Additionally, *Şişmanoğlu et al.* highlighted advanced processing power as a key factor in deep learning's success in stock market predictions, with a comparison of LSTM, GRU, and Bi-LSTM models showing Bi-LSTM to be particularly promising with a 63.54% accuracy, underscoring the benefits of parallel computing in model performance (Şişmanoğlu et al., 2020).

Recent studies have shown deep learning's adaptability to various markets and timeframes. In a study on XU100 volatility prediction, *Aker* found that LSTM outperformed Facebook Prophet, highlighting LSTM's accuracy in financial volatility prediction (Aker, 2022). Similarly, *Üntez and İpek* applied LSTM and ARIMA to forecast silver/ounce exchange rates, revealing that LSTM excelled in daily closing predictions while ARIMA performed better in trend estimations (Üntez and İpek, 2022).

Expanding beyond Turkish markets, *Toprak et al.* forecasted PETKM stock prices using multivariate inputs, including the USD/TRY exchange rate and the XKMYA index, with both LSTM and Random Forest Regression (RFR) models surpassing CNN in predictive accuracy (Toprak et al.,2023). This research was complemented by *Nirob and Hasan*, who applied LSTM to predict the prices of major U.S. stocks like Google and Netflix, achieving a 96.65% accuracy rate, which outperformed simpler methods such as SMA and EMA, showcasing LSTM's robustness in high-dimensional forecasting (Nirob and Hasan, 2023). Another study by *Livieris et al.* found that LSTM models combined with CNN layers can enhance predictive accuracy for stock markets, suggesting the value of hybrid deep learning models in financial forecasting (Livieris et al., 2020).

Further diversifying applications, *Özden* demonstrated CNN's superior performance over LSTM and RFR in predicting the prices of Turkish dietary staples with minimal error (Özden, 2023). Deep learning models were also applied by *İlkuçar* to predict Turkish Airlines stock prices, where LSTM and GRU achieved a 99% success rate despite pandemic-related volatility, validating the reliability of deep learning for financial forecasting under unstable conditions (İlkuçar, 2023).

In the realm of emerging markets, *Akbulut and Adem* investigated the impact of global economic factors on the XU100 index using LSTM, achieving high accuracy metrics and affirming deep learning's role in global financial predictions (Akbulut and Kemal, 2023). Complementing these studies, *Tanışman et al.* used LSTM and ARIMA to forecast Bitcoin prices, finding that LSTM excelled in both short- and long-term predictions, while ARIMA proved more effective for short-term accuracy, illustrating the versatility of deep learning across asset classes (Tanışman et al., 2021).

Collectively, these studies illustrate the growing role of deep learning, particularly LSTM, in financial forecasting. By addressing nonlinear dependencies and leveraging complex, multivariable data, deep learning models enable more accurate and adaptable forecasting frameworks across diverse financial contexts. These studies exemplify the efficacy of deep learning models in capturing the temporal dependencies in financial time series and support the development of advanced, multivariate forecasting approaches in finance.

2. Methodology

LSTM is a type of Recurrent Neural Network (RNN) designed to retain information over long sequences, enhancing training on extended data series and addressing the "vanishing and exploding gradient" issues common in traditional RNNs. Developed to overcome challenges in standard RNN training, LSTM utilizes a unique memory cell capable of storing information over extended periods. Through a series of gates, the LSTM cell controls data flow, improving its performance by regulating the amount of information passed on. The recurrent connections in LSTM provide the network with an intrinsic memory, enabling it to learn patterns within input sequences and adjust outputs based on previous context. This ability makes LSTM an effective solution for handling long-term dependencies and understanding temporal relationships, positioning it as one of the most effective algorithms for predicting sequential data. LSTMs are well-suited for the prediction, classification, and generation of sequential data, where a sequence represents an ordered set of observations rather than an unordered collection. Examples include test sets ordered by timestamps and videos composed of sequential frames or audio clips (Manaswi et al., 2018).

In an LSTM network, there are several core components: the forget gate, input gate, memory cell, candidate memory cell, output gate, and hidden state. During model training, input data flows through the network. Each of the gates—forget, input, and output—applies weights and reduces data using a sigmoid function, outputting values between 0 and 1. Upon passing to the candidate memory cell, the data is processed with a tanh function, which normalizes values between -1 and 1. If the forget gate outputs a value close to 1 and the input gate outputs a value close to 0, the information in the memory cell is retained in the subsequent flow. This structure is an ideal approach to address the vanishing gradient problem. The data output from the input gate is multiplied by the tanh function's output and passed to the candidate memory cell. If a value already exists in memory, it is multiplied by the forget gate output and added to the candidate memory cell's result. This summation then passes through a tanh function and is multiplied by the output gate value. When the output gate yields a value close to 1, it fully activates, allowing all the information within the memory cell to flow forward into the hidden state, making this complete information available for predictions. On the other hand, if the output gate value is closer to 0, it restricts the flow, meaning only a small fraction of the memory cell's information is carried forward, limiting the predictor's access to this memory. This iterative process continues as the hidden state cell and input data combine, restarting the cycle for new inputs (Brownlee, 2018). Figure 1 illustrates the flow of a sample network (Zhang et al., 2021).

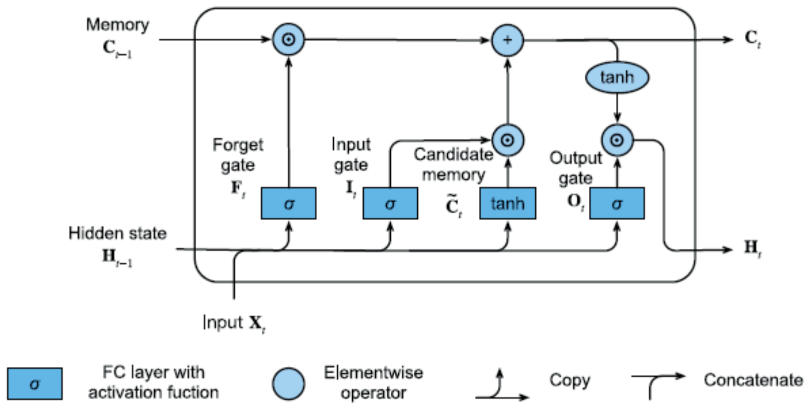

Memory $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$
FC layer with activation fuction
Elementwise operator
Copy
Concatenate

Figure 1. LSTM network cell structures (Zhang et al., 2021).

The LSTM algorithm's stages are defined by the following formulas:

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{1}$$

$$i_t = \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{2}$$

$$o_t = \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{3}$$

$$\overline{c}_t = \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \overline{c}_t \tag{5}$$

$$h_t = o_t \odot \sigma_h\left(c_t\right) \tag{6}$$

In these equations, d and h represent the number of input elements and hidden units, respectively:

$x_t \in R^d$: Input vector.

$f_t \in (0, 1)^h$: Forget gate vector.

$i_t \in (0, 1)^h$: Input gate vector.

$o_t \in (0, 1)^h$: Output gate vector.

$h_t \in (-1, 1)^h$: Hidden output vector of the LSTM unit.

$\overline{c}_t \in (0, 1)^h$: Candidate memory vector.

$c_t \in R^h$: Memory vector.

Additionaly:

$W \in R^{h*d}$, $U \in R^{h*h}$, and $b \in R^h$: represent weight and bias matrices.

σ : Activation function.

$\odot$: Hadamard product (element-wise multiplication) (Brownlee, 2018; Zhang et al., 2021).

This formulation provides a structured approach to understanding the LSTM cell's flow of information and highlights its ability to address vanishing gradient issues, making it an optimal choice for time-dependent sequence modeling.

2.1. Variants of LSTM

There are structural variations of LSTM networks developed for different applications. Below is a summary of these types.

2.1.1. Vanilla LSTM

The Vanilla LSTM network is the simplest variant, consisting of a single hidden layer used for predictions. The layers of this model are as follows:

- Input layer,
- LSTM layer,
- Dense layer,
- Output layer.

2.1.2. Stacked LSTM

Stacked LSTM networks are neural network models with multiple hidden layers. These layers create a hierarchical structure where each layer represents the data at a higher level, making them particularly effective for working with sequential data like time series. Stacked LSTMs provide greater learning capacity but require more computational power and data. The layers are as follows:

- Input layer,
- LSTM layer,
- LSTM layer,
- Dense layer,
- Output layer.

2.1.3. Bidirectional LSTM

Bidirectional LSTM (Bi-LSTM) networks allow for training in both forward and backward directions. By enabling two-way learning, the model processes inputs from both past-to-future and future-to-past, which can be advantageous in time series forecasting. The layers in this model are as follows:

- Input layer,
- Forward LSTM layer,
- Backward LSTM layer,
- Dense layer,
- Output layer.

2.1.4. CNN – LSTM

The CNN-LSTM model is a hybrid algorithm combining CNN and LSTM techniques. It is specifically designed to work with two-dimensional image data but can also be effective for learning sequential data such as time series. Using CNN and LSTM in a hybrid structure can enhance predictive power. The layers are as follows:

- Input layer,
- CNN layer,
- LSTM layer,

- Dense layer,
- Output layer.
- Output layer.

2.1.5. Convolutional LSTM

The Convolutional LSTM (ConvLSTM) variant integrates the convolutional operation directly into each LSTM cell. Unlike the CNN-LSTM model, where data is first processed by the CNN layer before passing to the LSTM cell, the ConvLSTM algorithm processes the data within the LSTM cell itself. The layer structure is as follows:

- Input layer,
- ConvLSTM layer,
- Flatten layer,
- Dense layer,
- Output layer (Brownlee, 2018), (Zhang et al., 2021), (Tekchandani and Kumar, 2023).

2.2. Performance Evaluation Metrics

Several metrics are used to evaluate the performance of forecasting models. Below are some of the key metrics commonly employed to test the neural networks used in this study.

- Mean Absolute Error (MAE), calculated by dividing the sum of absolute differences between actual and predicted values by the sample size:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{7}$$

- Mean Absolute Percentage Error (MAPE) measures the accuracy of model predictions in percentage terms:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{8}$$

- The Mean Squared Error (MSE) is the mean squared difference between actual and predicted values, used to quantify the average error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{9}$$

- Root Mean Squared Error (RMSE) is the square root of the average squared differences between observed and predicted values, providing an interpretable error metric in the same units as the data:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{10}$$

These metrics provide a comprehensive evaluation of model accuracy, enabling a robust comparison of forecasting performance across different neural network architectures (Hyndman and Koehler, 2006).

3. Experimental Result

In this section, we utilize data obtained from Yahoo Finance (finance.yahoo.com) on TUPRS stock covering the period from July 30, 2007, to September 29, 2023. Using multivariate input data, the objective is to predict the next day's stock price based on information from the previous five days and to evaluate the performance of various LSTM algorithm variants. The analysis was conducted on Google Colaboratory using the Python programming language.

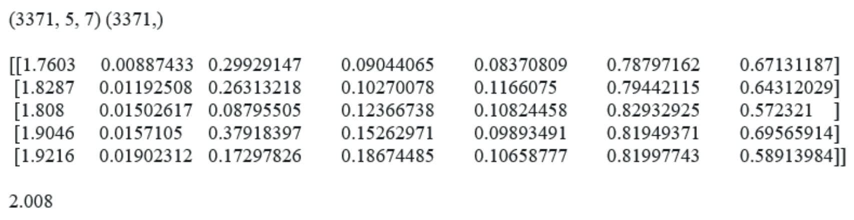
TUPRS stock was selected as the target variable due to its status as Turkey's largest and most influential oil refinery, playing a pivotal role in the Turkish economy and maintaining strong ties to global markets and economic indicators. The target variable is the closing price of TUPRS stock, while the independent variables include the XU030 index, Brent crude oil price, and gold ounce price. Additionally, through feature engineering, the analysis incorporates the difference between the intraday high and low prices, the difference between the intraday closing and opening prices, and the ratio of the 14-day moving average to the 60-day moving average. To ensure consistency in currency, the closing prices of these independent variables were converted to USD using the daily USD/TRY exchange rate. After preprocessing, the data will be structured for model input, and the findings from various models will be derived and analyzed. The variables used in the analysis are summarized in Table 1 as follows:

Table 1. Variables and descriptions used in the analyis.

Variable Name	Description
Close_usd	Closing price of TUPRS stock in USD.
Bist30_close_usd	Closing price of the XU030 index in USD.
Brentoil_close	Closing price of Brent crude oil.
Gold_close	Closing price of gold per ounce.
High_low_dif	Difference between the intraday high and low prices of TUPRS stock.
Close_open_dif	Difference between the intraday closing and opening prices of TUPRS stock.
14_over_60	Ratio of the 14-day moving average to the 60-day moving average of TUPRS stock prices.

The ratio of short-term to long-term moving averages provides valuable insights into identifying "golden cross" and "death cross" patterns. A golden cross occurs when the short-term moving average crosses above the long-term moving average, signaling a potential upward trend. Conversely, a death cross forms when the short-term moving average crosses below the long-term moving average, indicating a potential downward trend. Before training the models, the dataset was split into training and test sets, with the first 80% of observations (from July 30, 2007, to July 6, 2020) used for training, and the remaining 20% (from July 7, 2020, to September 29, 2023) reserved for testing, preserving the chronological order of the data. Independent variables were scaled between 0 and 1 using MinMax scaling to standardize the input values.

The dataset was then structured for the deep learning problem, where each example consists of values from the previous five days of independent variables to predict the TUPRS stock price for the following day. For this purpose, the data was organized in a three-dimensional tuple format, structured as (num_samples, timesteps, num_features), which corresponds to (3371, 5, 7) in this study. Here, num_samples represents the total number of days (3371), timesteps denotes the 5-day lag, and num_features indicates the 7 variables used. In Figure 2, for instance, the closing price of TUPRS stock was observed to be \$1.7603 five days prior, \$1.9216 one day prior, and \$2.008 on the day in question.



```
(3371, 5, 7) (3371,)

[[1.7603    0.00887433  0.29929147    0.09044065    0.08370809    0.78797162    0.67131187]
 [1.8287    0.01192508  0.26313218    0.10270078    0.1166075     0.79442115    0.64312029]
 [1.808     0.01502617  0.08795505    0.12366738    0.10824458    0.82932925    0.572321  ]
 [1.9046    0.0157105   0.37918397    0.15262971    0.09893491    0.81949371    0.69565914]
 [1.9216    0.01902312  0.17297826    0.18674485    0.10658777    0.81997743    0.58913984]]

2.008
```

Figure 2. The structure of the dataset for deep learning with 5 day-lag and 7 variables.

To gain an initial understanding of the target variable's behavior over time, Figure 3 presents the historical trend of the TUPRS stock closing price between 2007 and 2023.

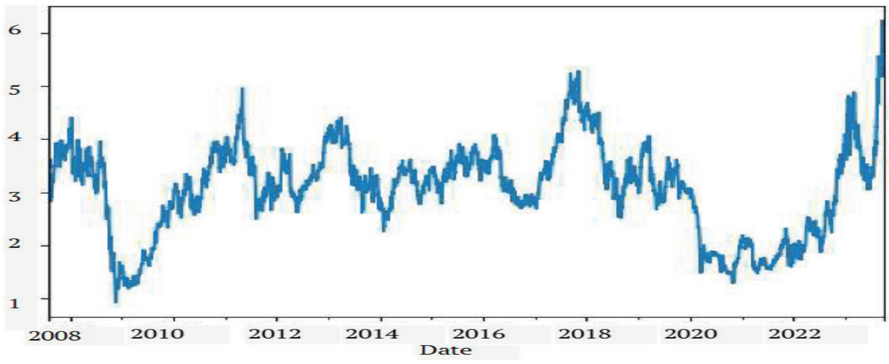


Figure 3. Time series plot of TUPRS stock closing price (2007–2023).

As observed in the Figure 3, the TUPRS stock price has undergone significant fluctuations, with notable peaks and troughs corresponding to various economic cycles. The price demonstrates periods of both rapid increase and decline, with a particularly sharp rise observed from 2021 onward. This volatility suggests that the stock is sensitive to both domestic and international market conditions, which may be influenced by factors such as commodity prices, exchange rates, and broader economic indicators. This trend justifies the inclusion of multivariate inputs in the modeling process to capture the complex factors affecting the stock's performance.

In this work, a total of 42 deep neural network models were run, utilizing five types of LSTM architectures: Vanilla LSTM, Stacked LSTM, Bi-LSTM, CNN-LSTM, and ConvLSTM. The model parameters used in the analysis are presented in Table 2 The success scores obtained from each algorithm will be examined individually by varying the number of neurons within each model. The number of epochs has been kept constant throughout, as increasing the epochs enhances model performance but also significantly extends training time.

Table 2. Default parameters for the models.

Parameter	**Value**
Batch size	32
Epoch count	200 or 500
Optimization algorithm	Adam
Activation function	ReLU
Loss function	MSE

The default parameters used in training the deep neural network models are presented in Table 2. The batch size was set to 32, and the number of epochs was either 200 or 500, depending on the model (ConvLSTM and CNN-LSTM utilized 500 epochs for enhanced performance). The Adam optimizer was employed for parameter updates, with ReLU as the activation function. Mean Squared Error (MSE) was selected as the loss function, suitable for regression-based predictive modeling tasks.

To evaluate the predictive performance of various LSTM architectures, a total of 30 models were run with different configurations in terms of epoch numbers and neuron counts. Table 3 presents the results, ranked according to MAPE, with the best-performing models highlighted. For clarity and ease of comparison, key performance metrics, including MAE, MSE, Root RMSE, and MAPE, are displayed for each model. This table serves as a comprehensive overview of model performance, allowing for the identification of the most effective configurations within each LSTM type.

Table 3. Performance metrics of LSTM-based models ranked by MAPE.

Rank	Model Type	Epoch	Neurons	MAE	MSE	RMSE	MAPE
1	Vanilla LSTM	200	25	0,0582	0,0085	0,0922	0,0220
2	Stacked LSTM	200	25 - 25	0,0594	0,0085	0,0923	0,0226
3	Vanilla LSTM	200	28	0,0592	0,0081	0,0898	0,0226
4	Bi-LSTM	200	25	0,0602	0,0083	0,0909	0,0233
5	Vanilla LSTM	200	21	0,0603	0,0084	0,0915	0,0233
6	ConvLSTM	500	50	0,0618	0,0088	0,0937	0,0236
7	Bi-LSTM	200	21	0,0609	0,0084	0,0914	0,0236
8	Vanilla LSTM	200	50	0,0622	0,0099	0,0994	0,0239
9	Stacked LSTM	200	50 - 50	0,0636	0,0110	0,1051	0,0239
10	Stacked LSTM	200	50 - 25	0,0625	0,0083	0,0909	0,0242
11	Stacked LSTM	200	28 - 14	0,0652	0,0089	0,0945	0,0257
12	Bi-LSTM	200	14	0,0651	0,0092	0,0959	0,0260
13	Bi-LSTM	200	28	0,0648	0,0087	0,0933	0,0261
14	ConvLSTM	500	18	0,0667	0,0098	0,0991	0,0261
15	Bi-LSTM	200	75	0,0672	0,0094	0,0970	0,0265
16	Vanilla LSTM	200	75	0,0682	0,0101	0,1007	0,0270
17	Vanilla LSTM	200	14	0,0689	0,0099	0,0997	0,0271
18	ConvLSTM	500	25	0,0702	0,0109	0,1046	0,0272
19	Stacked LSTM	200	25 - 12	0,0678	0,0095	0,0976	0,0273
20	Bi-LSTM	200	50	0,0688	0,0101	0,1007	0,0274
21	ConvLSTM	500	64	0,0759	0,0116	0,1079	0,0307
22	ConvLSTM	500	32	0,0809	0,0127	0,1128	0,0328
23	Stacked LSTM	200	32 - 32	0,0799	0,0113	0,1065	0,0332
24	CNN-LSTM	500	18 - 14	0,1014	0,0261	0,1615	0,0366
25	CNN-LSTM	500	32 - 14	0,1013	0,0229	0,1514	0,0385
26	ConvLSTM	500	82	0,1043	0,0171	0,1308	0,0455
27	CNN-LSTM	500	82 - 52	0,1354	0,0456	0,2136	0,0502
28	CNN-LSTM	500	64 - 25	0,1273	0,0331	0,1819	0,0502
29	CNN-LSTM	500	64 - 50	0,1348	0,0362	0,1903	0,0544
30	CNN-LSTM	500	32 - 25	0,1405	0,0370	0,1924	0,0604

As observed in Table 3, Vanilla LSTM and Stacked LSTM models tend to perform better in terms of MAPE, especially when using lower neuron counts, as evidenced by the leading scores of 0.0220 and 0.0226. The Bi-LSTM model also yields competitive results, achieving a MAPE of 0.0233 with 25 neurons. Notably, increasing the number of neurons does not always improve performance and, in some cases, may lead to slight increases in error metrics, possibly due to overfitting or higher computational requirements without proportional gains in prediction accuracy.

The CNN-LSTM models, while effective for capturing spatial and temporal dependencies, showed relatively higher error rates in this study, likely due to the complexity of the data and the sensitivity of CNN layers to hyperparameter tuning in time series forecasting. Consequently, Vanilla LSTM appears to be the most efficient architecture for this dataset, balancing accuracy and computational efficiency.

Following the detailed examination of individual performance metrics in Table 3, Table 4 presents a summary of the best-performing configurations for each LSTM-based model variant, ranked by the MAPE metric. This summary table allows for a streamlined comparison, highlighting each algorithm's optimal settings in terms of epoch number, neuron configuration, and corresponding performance scores.

Table 4. Summary of optimal performance configurations for LSTM-based models.

Rank	Model Type	Epoch	Neurons	MAE	MSE	RMSE	MAPE
1	Vanilla LSTM	200	25	0,0582	0,0085	0,0922	0,0220
2	Stacked LSTM	200	25 - 25	0,0594	0,0085	0,0923	0,0226
4	Bi-LSTM	200	25	0,0602	0,0083	0,0909	0,0233
6	ConvLSTM	500	50	0,0618	0,0088	0,0937	0,0236
24	CNN-LSTM	500	18 - 14	0,1014	0,0261	0,1615	0,0366

Table 4 effectively consolidates the top results from various configurations, showcasing the best scores achieved across different model types. For instance, Vanilla LSTM achieves the lowest MAPE score of 0.0220 with 25 neurons, followed closely by Stacked LSTM with a MAPE of 0.0226 at a neuron configuration of 25-25. Both models demonstrate a strong balance between complexity and predictive accuracy, suggesting that simpler LSTM structures may be more effective for this dataset than more complex architectures.

The Table 4 also provides insights into the trade-offs among models. Notably, CNN-LSTM, despite its hybrid structure designed to capture both spatial and temporal features, yields a significantly higher MAPE of 0.0366. This indicates that while CNN-LSTM may excel in certain domains, its complexity does not necessarily translate into improved accuracy for time series forecasting in this specific financial context.

In summary, Table 4 not only highlights the best configurations but also provides a practical takeaway: simpler models like Vanilla and Stacked LSTM might outperform more complex variants like CNN-LSTM and

ConvLSTM in this financial forecasting task. This observation is valuable for future research directions, as it suggests that the additional layers or hybrid structures of advanced models do not always yield substantial accuracy improvements and may introduce unnecessary computational overhead.

These findings provide insights into the suitability of different LSTM architectures for financial time series data, emphasizing the importance of model simplicity and strategic selection of neuron counts for optimal performance. Only the best-performing model, Vanilla LSTM with 25 neurons, is further visualized in Figure 4 to illustrate the predictive accuracy achieved by this configuration.

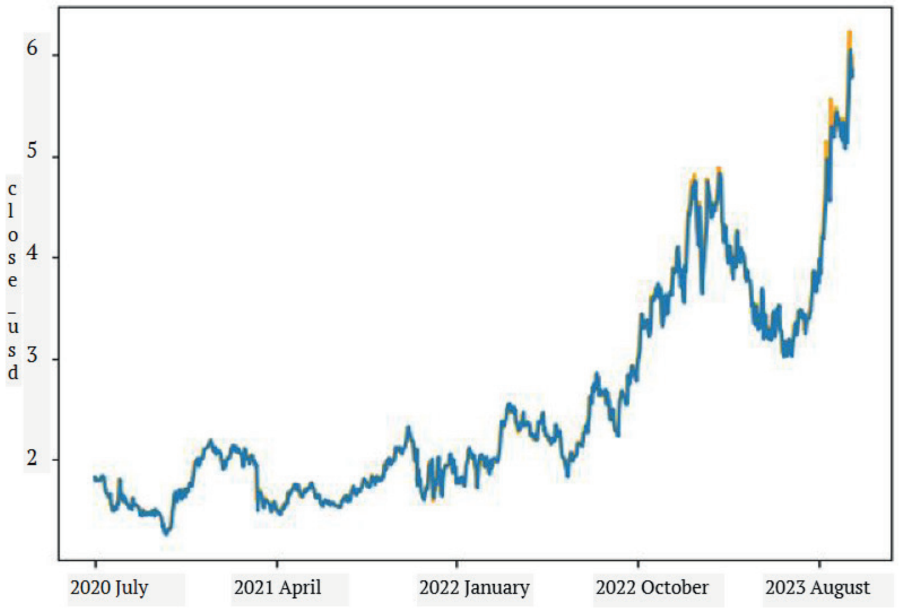


Figure 4. Vanilla LSTM model with 25 neurons prediction plot for TUPRS stock prices.

In Figure 4, the performance of the Vanilla LSTM model with 25 neurons in predicting TUPRS stock prices is displayed over the time frame from July 2020 to September 2023. The blue line represents the actual stock prices, while the orange points indicate the predicted values by the model. The model captures the general trend of the stock price movement, including the major upward trend that begins in 2021 and continues through 2023. Although there are minor deviations between the predicted and actual values, the Vanilla LSTM model demonstrates strong predictive accuracy in tracking the stock's volatile behavior, particularly in the upward and downward swings. This performance underscores the model's capability to learn and generalize temporal patterns effectively.

4. Conclusion

This chapter presents a comprehensive analysis of TUPRS stock price prediction using deep learning models, including Vanilla LSTM, Stacked LSTM, Bi-LSTM, CNN-LSTM, and ConvLSTM architectures. By utilizing multivariate input data, including the XU030 index, Brent crude oil prices, gold prices, and engineered features such as the difference between high and low prices, this research captures the complex relationships influencing TUPRS stock movements. The objective was to evaluate and compare the performance of these models, with a particular focus on the MAPE as a primary metric.

The results indicate that Vanilla LSTM with 25 neurons achieved the highest accuracy among all models, demonstrating the model's strength in capturing both long-term trends and short-term fluctuations in stock price. The CNN-LSTM and ConvLSTM models, while performing adequately, showed slightly higher MAPE values, likely due to the complex convolutional structures that are better suited for spatial data rather than sequential financial data. Stacked LSTM and Bi-LSTM models also showed robust performance, yet their added complexity did not result in significantly improved predictive accuracy over the simpler Vanilla LSTM.

This research confirms that LSTM-based models, particularly Vanilla LSTM, are effective in predicting stock prices in volatile and complex markets, especially when multivariate data is incorporated. The study also emphasizes the importance of feature engineering, as derived features like the moving average ratio provided valuable insights into trend reversals. Additionally, the findings underline the significance of model parameter tuning, including neuron counts and epoch settings, as these directly influence prediction accuracy and training efficiency.

Future research could explore integrating additional macroeconomic indicators and employing hybrid models that combine LSTM with attention mechanisms to further enhance predictive accuracy. Additionally, extending the forecasting horizon to predict multiple future days or employing ensemble methods could offer deeper insights and potentially improve performance. Overall, this work contributes to the growing body of literature on deep learning applications in financial forecasting and provides a practical framework for researchers and practitioners interested in stock price prediction in emerging markets.

References

Adem, K., Zengin, N., Hekim, M., & Karaca, S. S. (2016). Prediction of the relationship between the bist 100 index and advanced stock market indices using artificial neural network:(2011-2015). *Journal of New Theory*, (13), 86-95.

Akbulut, S., & Kemal, A. D. E. M. (2023). Derin öğrenme ve makine öğrenmesi yöntemleri kullanılarak gelişmekte olan ülkelerin finansal enstrümanlarının etkileşimi ile Bist 100 tahmini. *Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, *12*(1), 1-1.

Aker, Y. (2022). Analysis of price volatility in BIST 100 index with time series: comparison of Fbprophet and LSTM model. Avrupa Bilim ve Teknoloji Dergisi, (35), 89-93.

Alpay, Ö. (2020). LSTM mimarisi kullanarak USD/TRY fiyat tahmini. Avrupa Bilim ve Teknoloji Dergisi, 452-456.

Altunkaya, D., & Yılmaz, B. (2020). Multivariate Short-term Load Forecasting Using Deep Learning Algorithms. *The Eurasia Proceedings of Science Technology Engineering and Mathematics*, *11*, 14-19.

Brownlee, J. (2018). Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, *270*(2), 654-669.

G. James, D. Witten, T. Hastie and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, 2nd Edition, New York: Springer Science+Business Media, 2021. Available: https://doi.org/10.1007/978-1-0716-1418-1

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388-427.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527-1554.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International journal of forecasting, 22(4), 679-688.

İlkuçar, M. (2023). Prediction Turkish airlines BIST stock price through deep artificial neural network considering transaction volume and seasonal values. *Bilişim Teknolojileri Dergisi*, *16*(1), 43-53.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN–LSTM model for gold price time-series forecasting. *Neural computing and applications*, *32*, 17351-17360.

Manaswi, N. K., Manaswi, N. K., & John, S. (2018). *Deep learning with applications using python* (pp. 31-43). Berkeley, CA, USA: Apress.

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017, May). Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419-1426). Ieee.

Nirob, F. A., & Hasan, M. M. (2023). Predicting Stock Price from Historical Data using LSTM Technique. Journal of Artificial Intelligence and Data Science, 3(1), 36-49.

Özden, C. (2023). Comparative analysis of CNN, LSTM and random forest for multivariate agricultural price forecasting. Black Sea Journal of Agriculture, 6(4), 422-426.

Şişmanoğlu, G., Koçer, F., Önde, M. A., & Sahingoz, O. K. (2020). Derin öğrenme yöntemleri ile borsada fiyat tahmini. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, *9*(1), 434-445.

Tanışman, S., Karcıoğlu, A. A., Ugur, A., & Bulut, H. (2021). LSTM sinir ağı ve ARIMA zaman serisi modelleri kullanılarak bitcoin fiyatının tahminlenmesi ve yöntemlerin karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*, (32), 514-520.

Tekchandani, R., & Kumar, N. (2023). Applied Deep Learning: Design and implement your own Neural Networks to solve real-world problems (English Edition). BPB Publications.

Toprak, Ş., Çağıl, G., & Kökçam, A. H. (2023). Stock Closing Price Prediction with Machine Learning Algorithms: PETKM Stock Example In BIST. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 11(2), 958-976.

Üntez, A., & İpek, M. (2022). Developing Financial Forecast Modeling With Deep Learning On Silver/Ons Parity. *Journal of Advanced Research in Natural and Applied Sciences*, *8*(1), 35-44.

Wu, H., Chen, S., & Ding, Y. (2023). Comparison of ARIMA and LSTM for Stock Price Prediction. *Financial Engineering and Risk Management*, *6*(1), 1-7.

Yorulmus, H., Ugurlu, U., & Tas, O. (2018). A long short term memory application on the Turkish intraday electricity price forecasting. *PressAcademia Procedia*, *7*(1), 126-130.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. arXiv preprint arXiv:2106.11342.