MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 3, pp. 674–681 (2024)

A Comparison of LSTM, GRU, and XGBoost for forecasting Morocco's yield curve

Jeaab K.[1], Saoudi Y.[2], Falloul M. E. M.[1]

[1] *Sultan Moulay Slimane University, Economics and Management Laboratory, Khouribga, Morocco*
[2] *Ibn Tofail University, Advanced Systems and Engineering Laboratory, Kenitra, Morocco*

(Received 22 December 2023; Revised 5 August 2024; Accepted 15 August 2024)

The field of time series forecasting has grown significantly over the past several years and is now highly active. In numerous application domains, deep neural networks are exact and powerful. They are among the most popular machine learning techniques for resolving big data issues because of these factors. Historically, there have been numerous methods for accurately predicting the subsequent change in time series data. The time series forecasting problem and its mathematical underpinnings are first articulated in this study. Following that, a description of the most popular deep learning architectures used to date with success in time series forecasting is provided, emphasizing both their benefits and drawbacks. Feedforward networks, recurrent neural networks (such as Elman networks), long- and short-term memory (LSTM), and gated recurrent units (GRU) are given special consideration. Furthermore, the advantages of the XGBoost boosting tree method have shown its superiority in numerous data mining competitions in recent years. The high coefficients of the metric measures indicate that the proposed XGBoost model provides good predictive performance, according to the results.

Keywords: *forecasting; yield curve, deep learning; long short term memory; gated recurrent unit; eXtreme gradient boosting.*

2010 MSC: 68T20, 62M10, 62P05, 91G30, 68T07 **DOI:** 10.23939/mmc2024.03.674

1. Introduction

Time series forecasting is an important area of research that has been successfully used in several application areas [1, 2]. Time series forecasting involves making predictions behavior of a system using data about its past and present states. Due to their high dimensionality, complex market dynamics, and extremely high noise levels, the extraction of financial time series is difficult, which complicates the determination of similarity. Recent developments in the field of machine and deep learning have led to the development of several machine-learning techniques for the analysis of financial time series. In order to circumvent the drawbacks of conventional forecasting techniques, this problem has recently attracted the interest of machine learning researchers. We will focus on three machine learning models in this research. To mimic financial time series, including stock prices, interest rates, and stock market indices, long-term memory (LSTM) [3, 4] has long been used. In addition, Gated Recurrent Unit (GRU) networks [5] outperform traditional Recurrent Neural Networks (RNN) when learning long-term dependencies, overcoming the challenges posed by gradient fading and bursting. The eXtreme Gradient Boosting (XGBoost) method is more frequently employed than support vector networks and random forests as machine learning approaches [6]. It offers considerable advances in terms of accuracy, speed, and the ability to take into account data attributes. The rest of the paper is organized as follows: we describe the mathematical modeling of time series. Then we describe the architectures of three models. We carry out a practical study of the Moroccan Treasury bill reference rate as our time series. In addition, the experimental results of this study. And we end with a conclusion.

2. Yield curve

What exactly is a yield curve?

A yield curve is a graphical representation of the relationship between interest rates and the maturity of debt securities. It is a valuable tool for understanding market sentiment, predicting economic trends, and making informed investment decisions.

In practice, there are several yield curves on the markets. We can distinguish two families:

— Market curves, i.e., those constructed directly from market quotations:
  - swap rate curve;
  - bond yield curve for the government;

— Implied curves, which are created by extrapolating market quotations for securities like bonds and swaps:
  - zero-coupon yield curve;
  - forward rate curve;
  - forward rate curve that is immediate;
  - curve of yield at par.

For many financial and economic choices, yield curve forecasting is crucial. In addition to being important for investment management, monetary policy, risk management, and economic planning, it gives significant information on market predictions of future interest rates.

3. Time series

A time series (or chronological series) is a sequence of observations $x_1, x_2, \ldots, x_n$ indexed by time. It will be assumed that this is a realization of a process X, i.e. a sequence $\{X_i\}$ of random variables.

A time series is generally made up of several elements:

— Trend: represents the long-term evolution of the series (interannual scale). Examples of economic growth, and long-term climatological trends (cyclical or non-cyclical).

— Seasonality: changes occurring regularly every year, month, or week, every week. Examples: in meteorology, there are lower temperatures in the winter than in the summer. In economics, seasonality is induced by vacation periods, climate, etc.

— Stationary (or residual) component: what remains once the other components have been removed. Describe the short-term evolution of the series (daily scale).

Time series analysis can be further categorized as univariate or multivariate. The single observation is the only variable in the univariate time series. Multiple observations gathered over time make up the multivariate time series.

4. Machine learning algorithms

Algorithms are systematic operations used in various disciplines, including computer science, to teach computers how to perform specific tasks. Machine learning algorithms are widely used in data science due to their ability to learn independently from data, enabling them to quickly improve with practice, unlike other algorithms that operate without input. These algorithms are a narrow class of algorithms.

Algorithms for machine learning are employed in both data analysis and prediction. There are various machine learning algorithms, including [7]:

— supervised Learning;

— unsupervised Learning;

— reinforcement Learning.

5. Recurrent neural networks (RNNs)

Recurrent neural networks (RNNs) are powerful machine learning models for analyzing data sequences, such as text, speech, or time series. These networks enable machines to "remember" past information and use it to make decisions in real time.

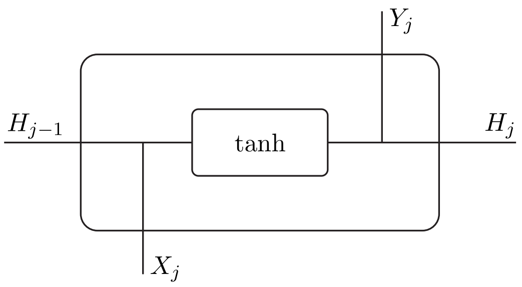

Fig. 1. A simple RNN unit.

Using hidden state vectors to hold historical data in sequential data is the basic notion behind RNNs. Figure 1 displays a basic RNN unit, referred to as a vanilla RNN, where H_{j-1} is identified as the hidden state vector at time t_{j-1}. At time t_j, X_j, and Y_j are the inputs and outputs, respectively. Updates to the hidden state vector and outputs are made by

$$H_j = g(W_{HH}H_{j-1} + W_{HX}X_j + b_H),$$
$$Y_j = g(W_{YH}H_j + b_Y),$$

where g is the activation function, such as the sigmoid function; W_{HH}, W_{YH}, b_H, and b_Y are trainable weights and biases to be determined by the training process using the training dataset by minimizing the loss function.

6. Long short-term memory (LSTM)

The LSTM network is a type of recurrent neural network that is used to describe long-term dependencies [8]. Long short-term Memory (LSTM) is a series of memory cells that store and update information in significant increments. Each cell contains three gates: input, output, and forget. The input gate adds information, the output gate outputs it, and the forget gate discards it. The network learns the gates based on input and hidden state [4]. The gates have the following equations:

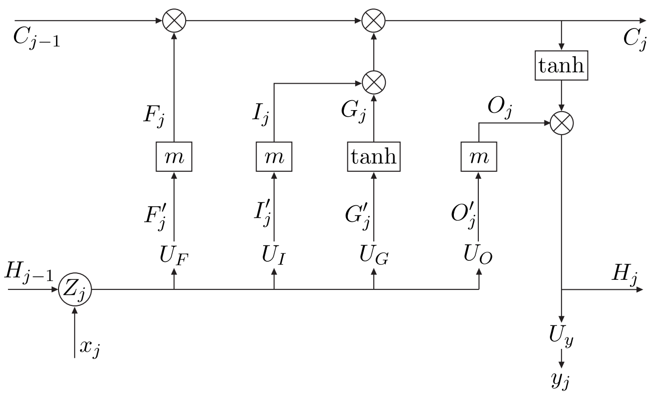

Fig. 2. An LSTM cell diagram.

input gate:

$$I_j = m(U_I H_{j-1} + U_I H_j);$$

forget gate:

$$F_j = m(U_F H_{j-1} + U_F H_j);$$

output gate:

$$O_j = m(U_O H_{j-1} + U_O H_j);$$

intermediate cell state:

$$G_j = \tanh(U_G H_{j-1} + U_G H_j);$$

cell state (nest memory input):

$$C_j = (I_j * G_j) + (F_j * C_{j-1});$$

new state:

$$H_j = O_j * \tanh(C_j);$$

x_j is input vector; H_j is output vector.

The next parameter in our LSTM model in Keras is in Table 1.

Table 1. Hyperparameters for LSTM model.

Parameter	Epochs	Batch size	Optimizer	Loss	Layer	Dense
Value	50	32	rmsprop	mse	5	1

7. Gated recurrent unit (GRU)

GRU, or Gated Recurrent Unit, is a simplified type of LSTM with two gates [5]: a reset gate and an update gate. The reset gate keeps the previous concealed state, whereas the update gate includes fresh input. The concealed state serves as both the cell state and output, eliminating the need for a separate output gate. GRU is less complicated to construct and has fewer parameters than LSTM [9]. The GRU has the following equations:

update gate:

$$L_j = m(W_L H_{j-1} + U_L X_j);$$

reset gate:

$$K_j = m(W_K H_{j-1} + U_K X_j);$$

cell state:

$$C_j = \tanh(W_C(H_{j-1} * K) + U_C X_j);$$

new state:

$$H_j = (L * C)((1 - L) * H_{j-1}).$$

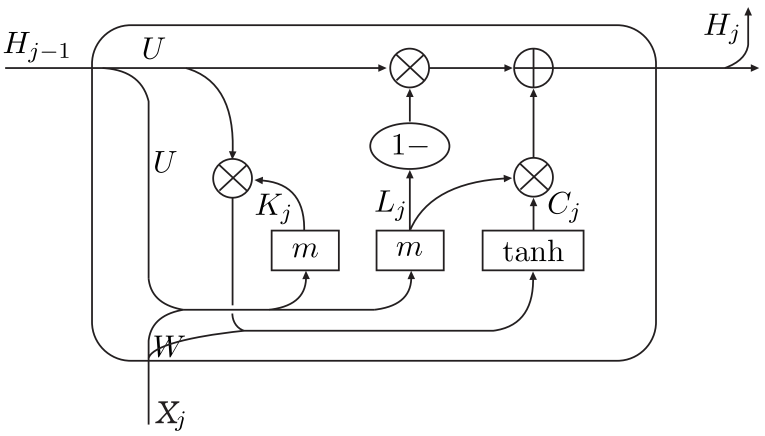

Fig. 3. Schematic for GRU Cell.

The next parameter in our GRU model in Keras is in Table 2.

Table 2. Hyperparameters for GRU model.

Parameter	Epochs	Batch size	Learning rate	Momentum	Loss	Layer	Dense
Value	50	32	0.01	0.9	mse	5	1

8. eXtreme gradient boosting (XGBoost)

There are two primary reasons why the XGBoost algorithm was used for this study's benchmark rate prediction of Moroccan treasury bills. First off, for the gradient boosting machine (GBM), XGBoost is among the most well-liked boosting tree algorithms. The industry has made extensive use of it because of its great performance in solving problems and low-feature engineering needs. Second, XGBoost is acknowledged as being more user-friendly for small datasets running on the CPU when compared to deep learning algorithms.

The Classification and regression tree (CART) is a decision tree used in stimulus tree algorithms to divide a dataset into two groups based on a variable's limit until the maximum tree depth is reached.

It can be described as below:

$$R_1 = \{x | x^j \leqslant s\} \quad \text{and} \quad R_2 = \{x | x^j \geqslant s\}.$$

Each leaf node's mean squared error is computed:

$$\text{MSE}_{node} = \sum_{i \in node} (\hat{y}_{node} - y^{(i)}),$$

$$\hat{y}_{node} = \frac{1}{m_{node}} \sum_{i \in node} \hat{y}^i,$$

where the number of occurrences in a node is denoted by m_{node}. The following is an expression for the cost function for CART regression:

$$J(k, t_k) = \frac{m_{left}}{m} \text{MSE}_{left} + \frac{m_{right}}{m} \text{MSE}_{right}.$$

The algorithm minimizes the cost function by finding the best solutions for variable bounds, resulting in the average target value of each instance in a single subset. However, CART trees can overfit without regularization, so ensemble bagging is used to address this issue.

XGBoost continuously adds and trains new trees to address previous iterations' mistakes, providing predicted values by aggregating scores from all corresponding leaves,

$$\hat{y} = \phi(x_i) = \sum_{k=i}^{k} f_k(x_i),$$

$$f_k(x_i) = w_q(x), \quad f_k \in \mathcal{F},$$

where k is the number of trees, $w_q(x)$ is the score for each leaf node, $q(x)$ is the number of leaf nodes, and $\mathcal{F}$ is an assembly of all related functions f_k. The result of input x_i for the k-th tree is represented by $f_k(x_i)$.

The regularization and the training error make up two components of XGBoost's objective function, which is expressed as:

$$\mathrm{obj}(\theta) = \sum L(\theta) + \sum \Omega(\theta).$$

The loss function (L) measures the difference between actual and projected values, while the regularization function (Ω) measures the training model's complexity to prevent overfitting,

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda \|w\|^2,$$

where ω is the score assigned to each leaf node and T is the total number of leaf nodes. Controlling factors like γ and λ are used to prevent overfitting.

The expected score for the t-th tree can be stated as follows when a new tree is constructed to accommodate residual errors of the previous iteration:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i).$$

Thus, the objective function is recast as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\big(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\big) + \Omega(f_t). \tag{1}$$

For each suitable function f_t, the second-order Taylor polynomial of $f_t = 0$ is substituted. As a result, the objective function can be roughly represented by:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[l\big(y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

If h_i represents the second-order derivative and g_i is the first-order derivative:

$$g_i = \partial \hat{y}_i^{(t-1)} l\big(y_i, \hat{y}_i^{(t-1)}\big), \quad h_i = \partial^2 \hat{y}_i^{(t-1)} l\big(y_i, \hat{y}_i^{(t-1)}\big).$$

As a result, Eq. (1) can be reduced as follows since the residual errors (y) of earlier ($t-1$) trees have little bearing on how the objective function is modified,

$$\bar{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t).$$

All examples belonging to the same leaf node can be reconstructed as follows once each instance is finally classified into a single leaf node:

$$\mathrm{obj}^{(t)} = \sum_{i=1}^{T} \left[\Big(\sum_{i \in I_j} g_i \Big) w_j + \frac{1}{2} \Big(\sum_{i \in I_j} g_i + \lambda \Big) w_j^2 \right] + \gamma.$$

Consequently, the objective function and optimal w are obtained as follows:

$$w_j^* = \frac{G_j}{H_j + \lambda} \mathrm{obj} = -\frac{1}{2} \sum_{j=1}^{t} \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

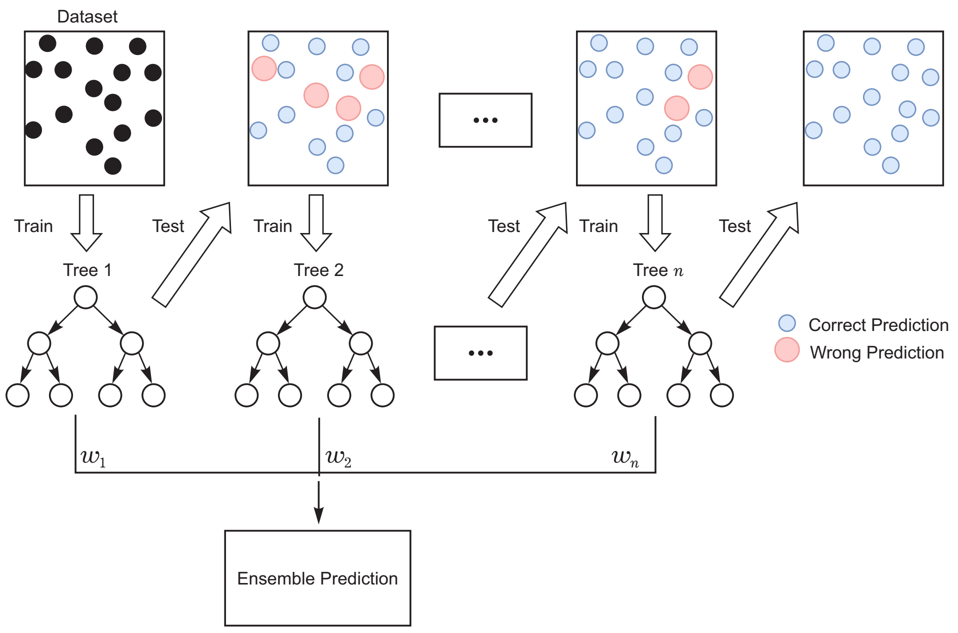

Fig. 4. Schematic of XGBoost Trees.

XGBoost is a powerful algorithm for tree creation due to its dependable objective function, useful functions like dividing threshold and maximum depth, and overfitting-avoidance techniques like column subsampling and shrinking. Shrinkage limits the influence of a single tree, while column subsampling creates a tree with only a portion of attributes, similar to a random forest. The next parameter in our XGBoost model in Keras is in Table 3.

Table 3. Hyperparameters for XGBoost model.

Parameter	Base score	Booster	N estimators	Objective	Max depth	Learning rate	Verbose
Value	0.5	gbtree	15000	reg:linear	3	0.01	100

9. Measure metric

Through the use of measures like mean absolute error (MAE), mean absolute percentage error (MAPE), root mean squared error (RMSE), and correlation coefficient (R^2), this study assesses the effectiveness of three models in forecasting the volatility. The average percent discrepancies between predicted and actual values are represented by MAE, the average squared errors between observed and predicted values are measured by RMSE, and the alignment between expected and actual numbers is determined by R^2,

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i|, \tag{2}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - \hat{x}_i}{x_i}\right|, \tag{3}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}, \tag{4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{N}(x_i - \hat{x})^2}, \tag{5}$$

where N is the total number of run-up data that have been seen, x_i and $(\hat{x}_i)$ are the i-th observed and forecasted run-up values, respectively, and $\hat{x}$ is the mean observed run-up value.

10. Experimental results

Table 4. Statistics of data.

Number of data	3075
Data missing	960
Mean	0.0217
Std	0.0047
Max	0.04
Min	0.013
Quartile: 25%	0.017
Quartile: 50%	0.0225
Quartile: 75%	0.024

We will use the Moroccan treasury bill reference rate download from the website of Bank Al-Maghrib as our time series data set and make predictions from July 1, 2015 to November 31, 2023.

The 3075 datasets are divided into training and testing data for implementing LSTM, GRU, and XGBoost algorithms. The ideal split is 80:20 or 70:30, with training taking up 80% or 70% and testing 20% or 30%. The 70:30 ratio was used in this study but can vary depending on dataset size.

Table 4 summarizes the data statistics.

The outcomes of three models are depicted in Figure 5. We use the Python programming language to make the graphs.

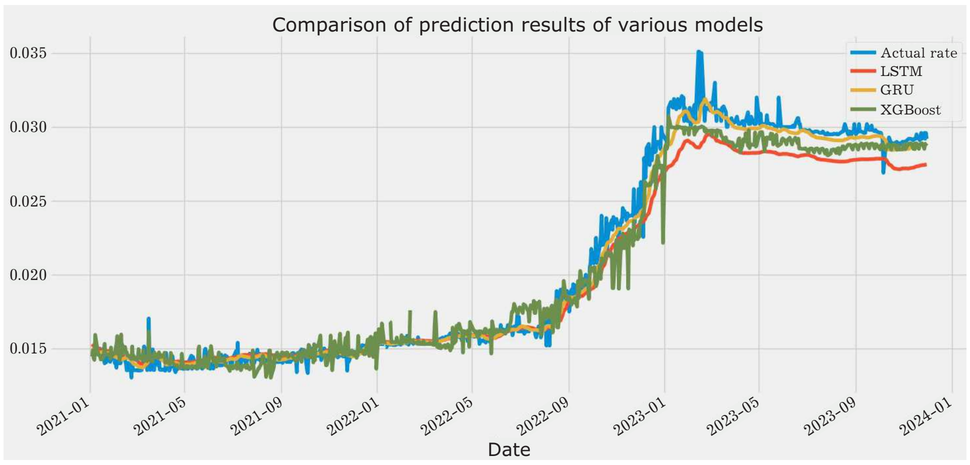

Fig. 5. Comparison of predictions of the three models.

Table 5. MAE, MAPE, RMSE, and R^2 values of the three models.

ML Method	MAE	MAPE	RMSE	R^2
LSTM	0.001	0.043	0.00145	0.9571
GRU	0.00086	0.04	0.0011	0.972
XGBoost	0.00047	0.022	0.0007	0.9891

Table 5 summarizes the measure metric computations for LSTM, GRU, and XGBoost models. The MAE, MAPE, and R^2 of GRU and XGBoost are roughly equal. XGBoost has slightly lower values than LSTM, with values of 0.00047, 0.022, and 0.9891. The correlation coefficient of 0.971 for XGBoost shows that it performs better at forecasting the yield curve than other models.

11. Conclusion

Machine learning researchers have been using financial time series forecasting for more than 40 years. With the advent of deep learning implementations for financial forecasting research, the financial community has recently experienced a surge in activity, leading to the appearance of numerous new articles. We provide a deep learning method in our work that can analyze and forecast time series. The three suggested methods are eXtreme Gradient Boosting (XGBoost), managed recurrent units (GRU), and long-term memory (LSTM). To provide a summary of the state-of-the-art in the field of deep learning applications to financial time series forecasting. For univariate and multivariate time series forecasting, this work used the scikit-learn, Pandas, NumPy, and Matplotlib machine learning libraries in a PythonSciPy environment together with the Keras, Tensor, and Xgboost Flow deep learning libraries. The decision was driven mostly by deep learning models' increased capacity to represent time-dependent data and their adaptability in capturing process non-linearity. Each model's performance was confirmed in terms of R^2, RMSE, MAE, and MAPE. The xgboost model outperformed all the other models in terms of forecasting performance, according to the results.

[1] Abbasimehr H., Paki R. Improving time series forecasting using LSTM and attention models. Journal of Ambient Intelligence and Humanized Computing. **13**, 673–691 (2022).

[2] Sagheer A., Kotb M. Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing. **323**, 203–213 (2019).

[3] Hwang J. Modeling Financial Time Series using LSTM with Trainable Initial Hidden States. Preprint arXiv:2007.0684 (2020).

[4] Guan Y. J. Financial time series forecasting model based on CEEMDAN-LSTM. 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC). 1–5 (2022).

[5] Yamak P. T., Yujian L., Gadosey P. K. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. ACAI'19: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. 49–55 (2020).

[6] S. Alhirmizy and B. Qader Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution. 2019 International Conference on Computing and Information Science and Technology and Their Applications (ICCISTA). 1–5 (2019).

[7] Chniti G., Bakir H., Zaher H. E-commerce Time Series Forecasting using LSTM Neural Network and Support Vector Regression. BDIOT'17: Proceedings of the International Conference on Big Data and Internet of Thing. 80–84 (2017).

[8] Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. **9** (8), 1735–1780 (1997).

[9] Zhai N., Yao P., Zhou X. Multivariate Time Series Forecast in Industrial Process Based on XGBoost and GRU. 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). 1397–1400 (2020).

Порівняння LSTM, GRU та XGBoost для прогнозування кривої дохідності Марокко

Жаб К.[1], Сауді Ю.[2], Фаллул М. Е. М.[1]

[1] *Університет Султана Мулая Сліман, Лабораторія економіки та управління, Хурібга, Марокко*
[2] *Університет Ібн Тофаїл, Лабораторія передових систем та інженерії, Кенітра, Марокко*

Сфера прогнозування часових рядів значно зросла за останні декілька років і зараз дуже активна. У багатьох областях застосування глибоких нейронних мереж є точними та потужними. Через ці фактори вони є одними з найпопулярніших методів машинного навчання для вирішення проблем із великими даними. Історично існувало багато методів для точного прогнозування наступних змін даних часових рядів. Проблема прогнозування часових рядів та її математичні основи вперше сформульовані в цьому дослідженні. Після цього надається опис найпопулярніших архітектур глибокого навчання, які на сьогодні успішно використовуються для прогнозування часових рядів, підкреслюючи як їхні переваги, так і недоліки. Особлива увага приділяється мережам прямого зв'язку, рекурентним нейронним мережам (таким як мережі Елмана), довго- та короткочасній пам'яті (LSTM) і вентильним рекурентним блокам (GRU). Крім того, переваги методу розширеного дерева XGBoost показали його перевагу в численних змаганнях з аналізу даних за останні роки. Згідно з результатами, високі коефіцієнти метричних показників вказують на те, що запропонована модель XGBoost забезпечує хорошу прогнозну продуктивність.

Ключові слова: *прогнозування; крива дохідності, глибоке навчання; довгострокова пам'ять; вентильний рекурентний блок; екстремальне посилення градієнта.*