Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Research on the Construction and Application of Macroeconomic Forecasting Model Based on Time Series Cluster Analysis

Dezhi Yang[1,✉]

[1] *College of Science, Eastern Liaoning University, Dandong, Liaoning, 118003, China*

ABSTRACT

Aiming at the problem of large prediction error caused by the complex background of macroeconomic prediction, this paper proposes a macroeconomic prediction model based on time series clustering. The model adopts sparse self-encoder to deeply mine the features of the input vectors, constructs a bidirectional threshold cyclic unit network, and predicts the preliminary trend of the macroeconomy, and proposes a time series deep clustering algorithm that integrates the multi-scale feature extraction and clustering objectives of time series data into the same network. A sample generation strategy based on data augmentation and a multiclassification assistance module are used to extract the invariant patterns contained in the time series data to obtain a better representation for targeting time series clustering. Comparing this paper's model with different forecasting models, the RMSE metrics are 0.0038 and 0.003 for the two time horizons, which are better than the other two models. The prediction range of this paper's model for future GDP is 5.8%-5.9%, which is smaller than the GDP prediction range of the ARIMA model, indicating that this paper's model is suitable for the realistic application of macroeconomic forecasting.

Keywords: Time series clustering, Self-encoder, Bidirectional threshold recurrent unit network, Economic forecasting

1. Introduction

Macroeconomic forecasting is the process of predicting the future trends of macroeconomic variables (e.g., gross domestic product, inflation rate, unemployment rate, etc.), which is of great significance for the formulation of economic policies and decision-making. The accuracy and reliability of macroeconomic forecasts directly affect the decision-making effects of government policy makers and entrepreneurs [1-4]. In order to make better macroeconomic forecasts, economists have designed a

✉ Corresponding author.
E-mail address: 18704155188@163.com (D. Yang).
Received 25 February 2024; Revised 05 May 2024; Accepted 10 December 2024; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-441

variety of macroeconomic forecasting models, and constantly research and improve them. Macroeconomic forecasting model is a mathematical model that describes the relationship between macroeconomic variables. According to the form and theoretical basis of the model, macroeconomic forecasting models can be categorized into time series models, causal models and structural models [5-8].

Time series analysis method is a statistical analysis method based on historical data, through the modeling and forecasting of time series data, it can reveal the relationship and trend between macroeconomic variables and provide the basis for macroeconomic forecasting [9-11]. In macroeconomic forecasting, commonly used time series analysis methods include trend analysis, cycle analysis and seasonality analysis. Trend analysis refers to the long-term average trend of time series data by fitting and portraying, in order to predict the trend of the future period of time, often using simple moving average method, exponential smoothing method and trend line fitting and other methods for forecasting [12-15]. Cyclical analysis refers to the analysis of cyclical fluctuations in time series data to predict the fluctuations in the future period. Seasonal analysis, on the other hand, refers to the prediction of seasonal changes in a future period by analyzing the corresponding seasonal fluctuations in time series data [16-18].

Literature [19] presents new methods and case studies in multivariate time series forecasting using a class of BPS models. The case study verifies that BPS possesses the potential to improve multiple series forecasts over multiple forecast horizons and points out its application in learning forecasting models, among others. Literature [20] aims to assess the accuracy of structural change modeling for macroeconomic forecasting. A simulated real-time out-of-sample exercise with a stochastic volatility VAR to develop forecasts for inflation, unemployment and interest rates reveals that this model is more accurate than models such as time-varying autoregression. Literature [21] investigated the predictive ability of financial variables on macroeconomic variables in several countries. It was found that financial variables such as credit growth and stock prices have a very strong predictive ability for macroeconomic variables. Moreover, cross-country panel models produced more accurate out-of-sample forecasts than single country models. Literature [22] introduced the similarity-based approach as a way to deal with parameter instability and used it in macroeconomic forecasting. This is done by using it for forecasting simulated data from Monte Carlo experiments and a wide range of key macroeconomic indicators. The results indicate that the proposed method performs well compared to other methods, especially during the crisis. Literature [23] proposes a hierarchical shrinkage VAR model for multicountry economic forecasting. By analyzing the impact of model specification and a priori selection on GDP growth and inflation of G7 economies. It was found that hierarchical contraction performs well in forecasting inflation and also has the best density forecasting performance for output growth and interest rates. Literature [24] compared alternative models for time-varying volatility based on real-time point and density forecasting accuracy of macroeconomic time series. Bayesian autoregressive and vector autoregressive models were also considered. The results illustrate that AR and VAR metrics with regular stochastic volatility are relatively good overall.

In this chapter, a macroeconomic forecasting method based on SAE-BiGRU is proposed by studying the structure and related theory of sparse self-encoder and two-way threshold recurrent unit algorithm. The method first performs feature extraction on the screened macroeconomic feature set by the sparse self-encoder, and then trains the BiGRU network with the extracted macroeconomic features as inputs. In order to further optimize the performance of the forecasting model, a time series deep clustering model based on multi-scale feature extraction is proposed. The coding network of this model makes full use of the advantage of multi-scale features of extracted time series data, and extracts the invariant features in macroeconomic data through the sample generation strategy and multi-categorization auxiliary module, which obtains a better representation of macroeconomic features, and verifies that the model in this paper can effectively predict the macroeconomic trend with real datasets.

2. Macroeconomic forecasting model based on time series clustering

2.1. Sparse Self-Encoder

Self-encoder is a kind of unsupervised learning neural network, which can autonomously learn the intrinsic features of unlabeled data, not only to reduce the dimensionality of the original data, but also to find another effective expression of the original data, to achieve data compression and feature extraction, which is widely used in the fields of data noise reduction, image recognition and semantic segmentation.

Self-encoder consists of two parts: encoder and decoder. The encoding process is to map the input vector into a new feature vector, while the decoding process is to reconstruct the generated feature vector into an output vector similar to the input vector. The training process of the self-encoder is to minimize the reconstruction error between the output vector and the input vector by continuously encoding and decoding and adjusting the parameters of the network structure, and then to find the optimal mapping relationship between the input layer and the implicit layer to extract the intrinsic features of the input vectors, so that unsupervised macroeconomic feature learning can be realized.

The network of the self-encoder consists of an input layer, a hidden layer and an output layer. From the input layer to the hidden layer, the self-encoder encodes the unlabeled samples $x=\{x_1,x_2,\cdots,x_n\}$ and encodes from equation (1) to compute the hidden layer value $h=\{h_1,h_2,\cdots;h_n\}$. From the hidden layer to the output layer, the self-encoder decodes $h=\{h_1,h_2,\cdots,h_n\}$ and decodes to compute the output layer value $z=\{z_1,z_2,\cdots,z_n\}$.

$$h=f(\omega_1 x+b_1) \tag{1}$$

$$z=g(\omega_2 x+b_2) \tag{2}$$

where f and g are the Sigmoid activation functions, ω_1 and ω_2 are the weights between the corresponding layers, and b_1 and b_2 are the biases between the corresponding layers.

Since the purpose of the autocoder is to make the output z as close as possible to x, the autocoder reduces the reconstruction error between z and x by continuously adjusting the network structure parameters during the training process, and the reconstruction error is measured by the cost function of the autocoder, which is as follows for the traditional autocoder:

$$J(W,b)=\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}\| h_{W,b}(x_i)-x_i \|^2) \tag{3}$$

where m denotes the number of input samples; W denotes the weight matrix; b denotes the bias vector; and $h_{w,b}(x_i)$ denotes the output of input x_i after the self-encoder, i.e. z_i.

Self-encoders are prone to overfitting problems during training, so a weight decay term is usually added. The weight decay term mainly plays the role of limiting the weight of the network, then the cost function of the self-encoder becomes:

$$J(W,b)=[\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}\| h_{W,b}(x_i)-x_i \|^2)]+\frac{\lambda}{2}\sum_{i}\sum_{i}\sum_{j}(W_{ij}^{(l)})^2 \tag{4}$$

where l denotes the number of layers of the network, $W_i^{(l)}$ denotes the value of the element of the weight matrix corner labeled i,j in layer l, and λ denotes the coefficient of the weight decay term.

The sparse self-encoder is based on the self-encoder and introduces a sparse penalty term to impose sparsity restriction on the self-encoder in order to enhance the learning ability and stability of the self-encoder. The sparsity restriction acts on the implicit layer of the self encoder.

The sparsity effect is achieved by suppressing the values of some nodes in the implicit layer. The specific implementation method is as follows:

1) Assuming that the average activity of neuron j in the hidden layer is $\hat{\rho}_j$, the $\hat{\rho}_j$ expression is as in equation (5):

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[h_j^{(2)}(x_i)\right] \tag{5}$$

where $h_j^{(2)}(x)$ denotes the activation of hidden layer neuron j given input x.

2) During the selfencoder training process, in order for the network to achieve a sparse effect, the average activity of certain neurons must be close to 0. If the average activity of the hidden layer neuron j is desired to be close to 0, the activation of the hidden layer neuron j for all the input samples must be close to 0. In order to satisfy this condition, it is necessary to add a penalty factor to the cost function to complete the sparsity restriction. The expression of the penalty factor is shown in equation (6):

$$\sum_{j=1}^{s_2} KL(\rho \| \hat{\rho}_j) = \sum_{j=1}^{s_2} \rho \log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{6}$$

where ρ denotes the sparsity parameter; s_2 denotes the number of neurons in the hidden layer.

3) The cost function $J_{sparse}(W,b)$ expression of SAE is shown in equation (7):

$$J_{sparse}(W,b) = J(W,b) + \beta\sum_{j=1}^{s_2} KL(\rho \| \hat{\rho}_j) \tag{7}$$

where β is the weight of the control sparsity penalty factor.

4) The gradient descent algorithm is used to iteratively update W,b to obtain the optimal weights and bias, and the update formula is shown in Eq. (8) and Eq. (9):

$$W_{ij}^{(l+1)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}} J_{sparse}(W,b) \tag{8}$$

$$b_i^{(l+1)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}} J_{sparse}(W,b) \tag{9}$$

where α is the learning rate.

2.2. Bidirectional Threshold Cyclic Cell Algorithm

Recurrent neural network (RNN) [25] is a kind of network for processing time series data. However, RNN has the problem of gradient vanishing or gradient explosion when dealing with data with too large a time-series span, and some scholars have improved it to address this problem. Long and short-term memory networks and threshold recurrent unit networks are both variants of RNN, which not only inherit the advantages of RNN, but also effectively avoid the problem of gradient disappearance or gradient explosion.

GRU is evolved on the basis of LSTM [26], which can track the data with large time-series span well. Compared to the LSTM network, there is no explicit unit state in the GRU network, which utilizes a reset gate to realize the roles of the forgetting gate and input gate in LSTM. By simplifying the structure of the gate, the GRU network structure is simplified, the model parameters are reduced, and the network's prediction performance on macroeconomic characteristics is improved.

x_i represents the input of the neuron at moment t, r_i represents the output of the reset gate at moment t, σ represents the sigmoid activation function, z_t represents the output of the update

gate at moment t, h represents the memory value of the neuron, h_t represents the output of the neuron at moment t; tanh represents the tanh activation function.

The neuron of GRU network [27] consists of update gate and reset gate. The mathematical model of GRU network is as follows:

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1}) \tag{10}$$

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1}) \tag{11}$$

$$h = \tanh(r_t * W_{sh}h_{t-1} + W_{sx}x_t) \tag{12}$$

$$h_t = (1-z_t) * h_{t-1} + z_t * h \tag{13}$$

where W denotes the corresponding weight matrix and " * " represents the Hadamard product operation.

Since both the reset and update gates use a Sigmoid activation function, the 1 value and the z_i value lie between 0 and r_i. The size of the r_i value determines how much the output h_{i-1} of the neuron of the previous moment contributes to the memory value h of the neuron of the current moment, and a larger value indicates that more information is retained h_{t-1} in h. When $r_i = 0$, it indicates that the current moment h completely discards the information in h_{t-1} and is only relevant to the input x_i of the current moment. It is this structure that makes the reset gate useful for capturing short-term dependencies.

The z_t value can affect how much the neuron's output h_t is updated. The larger the value of z_t, the smaller the value of $1-z_t$, the less information is retained by the output h_{t-1} of the neuron of the previous moment, and the greater the degree of h_t updating. When the value of z_t is equal to 1, the value of $1-z_t$ is equal to 0, and the output h_t of the neuron at the current moment completely discards the output h_{t-1} of the neuron at the previous moment, leaving the memory value h of the current moment intact.

The bi-directional threshold recurrent unit network is composed of two unidirectional GRU units with opposite directions, and the bi-directional structure can fully explore the deep time series features and improve the prediction effect.

The BiGRU network consists of an input layer, an implicit layer and an output layer. The hidden layer consists of two GRU neurons in opposite directions and a connection layer, and the output of the hidden layer is obtained by weighted summing the output of the forward GRU neurons and the output of the backward GRU neurons, and the specific calculation formula is as follows:

$$\vec{h}_t = GRU(x_t, \vec{h}_{t-1}) \tag{14}$$

$$\overleftarrow{h}_t = GRU(x_t, \overleftarrow{h}_{t-1}) \tag{15}$$

$$h_t = \omega_{\vec{h}_t}\vec{h}_t + \omega_{\overleftarrow{h}_t}\overleftarrow{h}_t + b_t \tag{16}$$

where, $\vec{h}_i$ denotes the output of the forward GRU at moment t, $\overleftarrow{h}_i$ denotes the output of the backward GRU at moment t, h_i denotes the output of the implicit layer of the BiGRU network at moment t, $\omega_{\vec{h}_i}$ denotes the weight at moment t of $\vec{h}_i$, $\omega_{\overleftarrow{h}_i}$ denotes the weight at moment t of $\overleftarrow{h}_i$, and b_i denotes the bias at moment t.

The output layer formula is shown in equation (17):

$$y_t = g(\omega_y h_t) \tag{17}$$

where y_i denotes the output of the output layer of the BiGRU network at moment t, g denotes the activation function, and ω_y denotes the weight coefficients of h_t.

2.3. DTCMFE time series clustering model

Given a data set containing N time series noted as $D=\{x_1,x_2,...,x_N\}$, where a certain time series data sample $x_i, i\in(1,2,\ldots,N)$ contains T ordered numerical vectors denoted as $x_i=\{x_{i,1},x_{i,2},\ldots,x_{i,T}\}$. where $x_i, T\in R^m$ denotes the i th sample T moments m dimensional vectors, the DTCMFE clustering model framework is shown in Fig. 1.

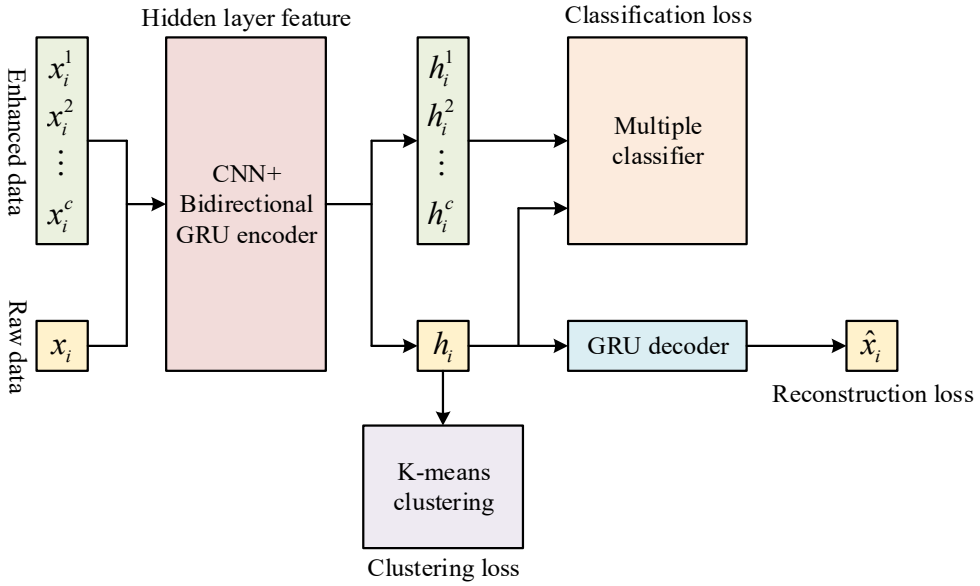

Fig. 1. DTCMFE clustering model framework

The model consists of three parts: the first part is the self-encoder module, which consists of an encoder consisting of a convolutional neural network with multiple convolutional kernels of different sizes and a network of bi-directionally gated recurrent units, as well as a GRU decoder, which is denoted by f_{enc}, f_{dec} to represent the encoding process of the self-encoder and the decoding process, respectively. The module obtains a representation h_i of the sample x_i by reconstructing the loss, which contains dynamic and multiscale features of the time series data. The second part is a multi-sample generation strategy based on data augmentation and a multi-classification assistance module, which augments the samples x_i by c data augmentation methods to obtain data samples $x_i^l, x_i^2, ..., x_i^e$, inputs the augmented samples $x_i^l, x_i^2, ..., x_i^e$ and samples x_i into an encoder f_{mc} to obtain feature representations $h_i^l, h_i^2, ..., h_i^f, h_i$, which are classified by a multi-classifier network based on the fully-connected layer as a means of extracting invariant patterns contained in the time-series data x_i; The third part is the K-means clustering module, which constrains the representation h_i to form a cluster structure by clustering losses.

2.3.1. Encoder structure design based on multi-scale feature extraction. Effective latent representation is the key to temporal clustering, and considering the multiscale nature of time series, a multiscale feature extraction based encoder network is designed in this paper to achieve this goal. The multiscale feature extraction based encoder is a coding network that combines a convolutional neural network layer with multiple convolutional kernels of different sizes and a network of bi-directionally gated recurrent units.

Where the first part is a multi-scale one-dimensional convolutional network layer consisting of m of the convolutional networks, where the convolutional kernel size of the k nd convolutional network

is s_k, where $k \in [1, m]$, let n be the number of channels of each convolutional network, and let the output of each channel be r_j, where $j \in [1, n]$, is defined as:

$$r_j = f(W_j x_i + b_j) \tag{18}$$

where x_i denotes the input data vector, W_j denotes the convolution kernel weight matrix, b_j is the bias vector, and $f(\cdot)$ denotes the activation function Leaky-RELU, after which the representation u_j is obtained by passing through a Maxpooling layer of size P.

Afterwards, the representation v_k is obtained by concatenation operation, where $k \in [1, m]$, is defined as:

$$v_k = concat[u_1, u_2, ..., u_n] \tag{19}$$

where $concat[\cdot]$ denotes the connectivity operation for one-dimensional vectors.

The v_k obtained in the first part is used as the input for the second part, which is used to learn long-distance dependencies on different time scales by means of bidirectional GRU and fully connected network layers to further reduce the data dimensions, making the representation more suitable for clustering tasks. In this case, the hidden layer of the two-way GRU network implements two processes, forward computation and backward computation, which can provide data context information for the network, and the computation process of the two-way GRU network is shown in Eq. (20) to Eq. (22):

$$z_k^+ = GRU^+(v_k) \tag{20}$$

$$z_k^- = GRU^-(v_k) \tag{21}$$

$$y_k = concat[z_k^+, z_k^-] \tag{22}$$

where $GRU^-(\cdot), GRU^+(\cdot)$ denotes the forward and reverse GRU network operations, respectively, and z_k^+, z_k denotes the final state of v_k through the forward and reverse GRUs, respectively, which is obtained by passing the obtained state through the connectivity operation to obtain γ_k, where $k \in [1, m]$, after passing the connectivity operation, is fed into the fully-connected network to obtain the latent representation h_i of the time series x_i, defined as:

$$h_i = f_R(W_i concat[y_1, ..., y_m] + b_l) \tag{23}$$

where W_l and b_l are the weight values and bias vectors of the fully connected network, respectively. $f_R(\cdot)$ denotes the activation function RELU, after which h_i is fed into the GRU decoder to obtain the output $\hat{x}_i$, where $\hat{x}_i \in R^T$, is defined as:

$$\hat{x}_i = GRU(h_i) \tag{24}$$

Use the mean square error (MSE) as the reconstruction loss, defined as:

$$L_{reconstruction} = \frac{1}{n}\sum_{i=1}^{n} \left\| x_i - \hat{x}_i \right\|_2^2 \tag{25}$$

2.3.2. Design of task modules based on data enhancement. Since the capability of the encoder determines how well the features are extracted, this chapter designs a multi-sample generation strategy based on data enhancement as well as a multi-categorization assistance module to enhance the encoder's functionality. The data enhancement methods used in this chapter are:

1) Time interrupt: given a time series $x_i \in R^T$, the random contiguous portion of the time series signal is replaced by zeros to produce augmented data. The selected time step is $[\alpha \times T]$, where $\alpha \in (0, 1]$.

2) Time delay: given a time series $x_i \in R^T$, the time series data is shifted randomly to the right by $[\beta \times T]$ time steps at a random time step where $\beta \in (0, 0.5]$.

3) Random Displacement: given a time series $x_i \in R^T$, randomly reorganize a certain percentage of time steps, with a selected time step length of $[\mu \times T]$, where $\mu \in (0,1]$.

Let c, $c \in [1,3]$ of the above data augmentation methods be used to augment x_i to obtain c augmentation data samples $x_i^1, x_i^2, ..., x_i^c$, different labels $y_i^l, y_i^2, ..., y_i^c$ are set for the augmentation samples of different augmentation method types, and the augmentation data sample $x_i^{'}, x_i^2, ..., x_i^c$ and the original data sample x_i are input to the encoder together to obtain the corresponding feature representation $h_i^1, h_i^2, ..., h_i^c, h_i$, and finally the prediction junction $\hat{y}_i^l, \hat{y}_i^2, ..., \hat{y}_i^c, \hat{y}_i$ is obtained through a multi-classifier network based on the fully connected layer, and the representation $h_i^l, h_i^2, ..., h_i^c, h_i$ is constrained by using the classification loss $L_{classification}$.

The classification loss function is defined as:

$$\hat{y}_i^j = W_{fc2}(W_{fc1}h_i^j)L_{classification} = -\frac{1}{cN}\sum_{i=1}^{cN}\sum_{j=1}^{c} 1\left\{y_i^{\ j} = 1\right\} \log \frac{\exp \hat{y}_i^{\ j}}{\sum_{j=1}^{c} \exp(\hat{y}_i^{\ j})} \tag{26}$$

where c is the number of classifications of the multiclassification assistance strategy, N is the number of samples, y_i^l denotes the c-dimensional one-hot vector labels of the augmented data of class j of the original data x_i, where $i \in (1,2,\ldots,c)$. $\hat{y}_i$ is the prediction results of x_i^l. $W_{f\in 1} \in \square^{m\times d}$, $W_{f\in 2} \in \square^{d\times c}$ are the parameters of the fully connected layer of the multiclassifier network.

2.3.3. K-means clustering module. In order to make the representation h_i learned by the selfencoder form a cluster structure, this method further guides the network learning through K-means clustering method.

The K-means algorithm (K-means algorithm) serves as a classical algorithm for large-scale data classification. It calculates the similarity between the data elements of macroeconomic features so as to group the similar data into one class, thus achieving the purpose of concentrating each class of macroeconomic features after classification. It works in the following steps: 1. First randomly select k data from the original macroeconomic data as the initial clustering centers. 2. Divide the remaining data to the nearest clustering centers. 3. Redetermine the clustering centers for each macroeconomic class after classification is completed. 4. Repeat steps 2 and 3 until convergence. 5.

Since there are scalar data and vector data, the K-means algorithm calculates the similarity of these two types of data differently, the specific differences are as follows.

The similarity of scalar data can be directly calculated by Euclidean distance, for example, two scalar variables $W = \{3,2,2\}, Z = \{1,4,3\}$.

The Euclidean distance of the two is calculated as follows:

$$d(W,Z) = \sqrt{(3-1)^2 + (2-4)^2 + (2-3)^2} = 3 \tag{27}$$

And vector data because of its own both size and direction, you can use the cosine of two vectors to measure the similarity of the two, such as the two vectors for W_1, Z_1, the cosine of the two similarity $s(W_1, Z_1)$ calculations are shown in Equation (28).

$$s(W_1, Z_1) = \frac{Z_1 W_1}{\| Z_1 \| \| W_1 \|} \tag{28}$$

where $\| Z \|$ denotes the L2 paradigm of Z and $\| W \|$ denotes the L2 paradigm of W.

Since clustering using the K-means algorithm directly divides the data to the nearest clustering center in each step of the classification process, but in reality many of the classes to which the data belongs are not in an either/or relationship, and it is unscientific to just mechanically divide the data into a certain class. Fuzzy C-mean clustering takes into account the shortcomings of K-means clustering, and does not directly classify the data into a certain class, but assigns multiple weights to the data, and the sum of the weights is one, and the number of weights is the number of categories of macroeconomic data classification, and each weight marks the similarity between the current data and the corresponding class.

The objective function of the fuzzy C-mean clustering algorithm is shown in equation (29):

$$A = \sum_{i=1}^{c} A_i = \sum_{i=1}^{c} \left(\sum_{x_k \in G_i}^{n} \left\| b_k - c_i \right\|^2 \right) \tag{29}$$

where the objective function is A, c represents the number of subgroups, and c_i is the clustering center of the macroeconomic characteristics of the group.

3. Analysis of the results of model application

This experiment uses Tu Share's stock data interface to crawl the stock information related to the example stock A. The data in this experiment are detected using the matrix function. Missing value detection is done using the matrix function, and the data in this experiment is detected as complete data with no missing values. Data normalization is performed using the MinMaxScaler function.

In order to improve the degree of linearity of the experimental data, the difference process of this paper's model is introduced to differentiate the data, and the results of 2019-2023 and 2020-2023 are shown in Fig. 2 and Fig. 3, respectively. When the time span gradually decreases, the corresponding data are more volatile and the differential processing effect is worse.

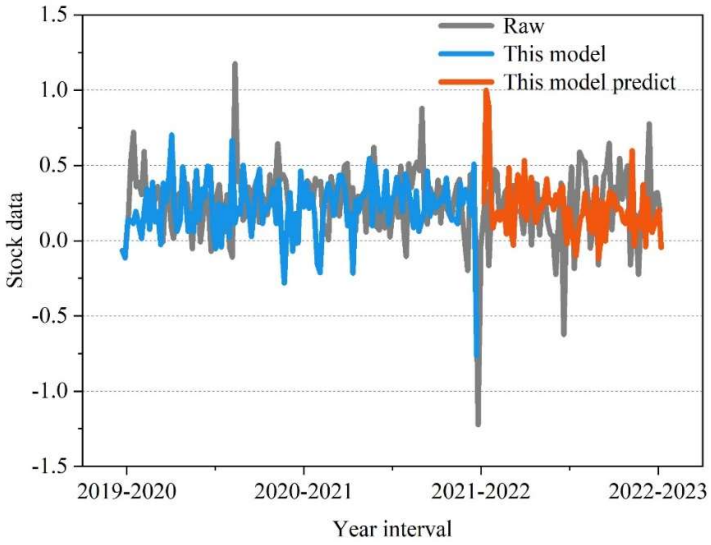

Fig. 2. 2019-2023 votes according to difference processing results

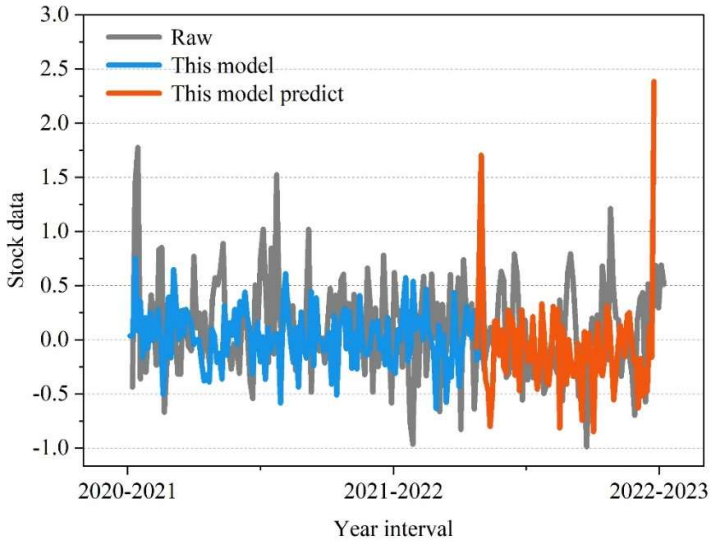

Fig. 3 2020-2023 votes according to difference processing results

In this paper, in order to verify the impact of data with different time spans on the experimental results, two sets of A-share data with different time spans are selected for modeling and analysis, namely a) 2019 to 2023, a total of four years, and b) 2020 to 2023, a total of three years. The first 80% of the data is taken as the training set to train the model, and the second 20% of the data is taken as the test set to test the predictive ability of the model, setting the values of batch_size and epochs to 20 and 10, respectively, i.e., the size of the sample for each input is 20, and repeating the training of the neural network for 10 times.The prediction results for the years 2019-2023 and the prediction results for the years 2020-2023 are shown in Figures 4 and 5, respectively. When the time span is long and the training set data is sufficient, the prediction ability of the BIGRU model is significantly improved, and the error between the prediction data and the real data of the model in this paper is reduced from 10% to 3%. With the increase of time horizon, the model is able to capture more market trends and price patterns to predict the movement of the target stock more accurately, which means that the model shows better adaptability and generalization ability when dealing with nonlinear data and large amount of data. Therefore, by increasing the time horizon and the amount of data in the training set, the model's economic forecasting accuracy can be further improved, leading to more confident investment decisions.

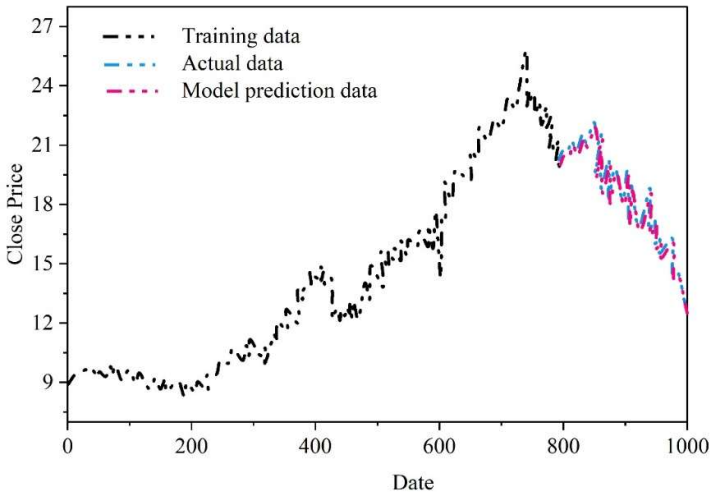

Fig. 4. 2019-2023 prediction results

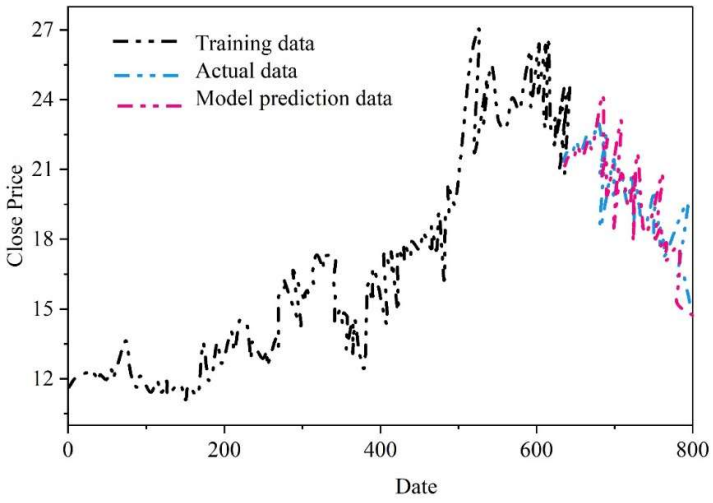

Fig. 5. 2020-2023 prediction results

The partial results of the cluster analysis of selected stocks based on K-Means algorithm for 2019~2023 and 2020~2023 are shown in Table 1 and Table 2, respectively. The degree of recommendation for stock A under the two different time spans is 2 and 1, respectively, both of which are at a high level.

The prediction results for different time horizons are shown, where the root mean square error is 0.0031 and 0.0048 for the 2019-2023 and 2020-2023 time horizons, respectively.

The model in this paper shows better prediction performance in long time horizon and is able to predict the movement of the target stock more accurately than the short time horizon. After several iterations, the loss function of the model under long time horizon decreases faster and converges more rapidly, while the model fits better and predicts more accurately. Combined with the differential processing, it can be clearly observed that the volatility of the corresponding data increases as the time span gradually decreases, which also leads to a decrease in the prediction effect.

With the average return, volatility and the degree of recommendation of A-share under different time horizons, we are able to understand more quickly that A-share is a stock with gentle ups and downs and stable returns, which has a better investment protection. So based on the observation of the prediction performance of this paper's model under different time horizons and the analysis of A stock, it can be concluded that this paper's model under long time horizons can provide more accurate stock prediction results, and A stock, as a stock with stable returns, has a high investment value.

Table 1. Partial Results of Stock Cluster Analysis from 2019 to 2023

Stock number	Average yield on target time	Target time volatility	Degree of recommendation (1 is the highest level)
A	-0.14459	0.33561	2
B	-0.47023	0.48478	3
C	-0.65412	0.33778	4
D	-0.66954	0.41445	1
E	-0.33784	0.29544	2
F	-0.33745	0.36878	3
G	-0.54711	0.38048	4
H	-1.01212	0.58742	5
I	-0.77445	0.35645	5
J	-0.48545	0.382152	3

Table 2. Partial Results of Stock Cluster Analysis from 2020 to 2023

Stock number	Average yield on target time	Target time volatility	Degree of recommendation (1 is the highest level)
A	0.56107	0.31635	1
B	0.19479	0.38867	3
C	0.19898	0.34253	3
D	0.51874	0.48759	1
E	0.25083	0.52989	2
F	-0.07252	0.31744	4
G	0.18309	0.40162	3
H	0.02942	0.2931	4
I	0.23061	0.29452	3
J	0.45446	0.47749	1

In order to have a more intuitive representation of the prediction effect, this experiment uses the ARIMA model as well as the BP neural network to predict the same experimental data, and the results for different time spans of 2019-2023 and 2020-2023 are shown in the model prediction part of the graph as shown in Fig. 6 and Fig. 7.In the year of 2019-2023, the RMSE values of this paper's model, ARIMA, and BP are, respectively, 0.003, 0.025, 0.0045. 0.003, 0.02, and 0.0045, and in 2020-2023, the RMSE values of this paper's model, ARIMA, and BP are 0.0038, 0.025, and 0.0049, respectively.The RMSE values of this paper's model are the smallest regardless of time span, which shows that this paper's model has more predictive ability and better accuracy performance. When using the same time span of experimental data for prediction, this paper's model shows the best prediction effect with its RMSE values of 0.003 and 0.0038 for different time spans, respectively. And its prediction curve is smooth, compared to the BP neural network whose prediction results are similar to the model in this paper, but with a slightly larger error. On the other hand, the ARIMA model is a time series model, but it is less capable of dealing with experimental data over long time spans, and its forecasting results tend to converge to a horizontal straight line, which fails to capture the complex movements of stock prices. Therefore, considering the forecasting effects of each model, the model in this paper performs well in stock price forecasting with high accuracy and stability, and can be regarded as a more reliable tool, and the selection of a suitable forecasting model is crucial for making investment decisions and managing risks in practical applications.

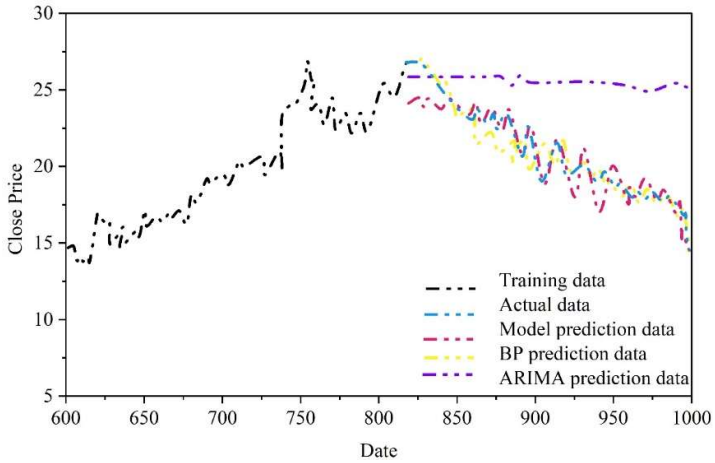

Fig. 6. 2019-2023 results in the prediction section of the model

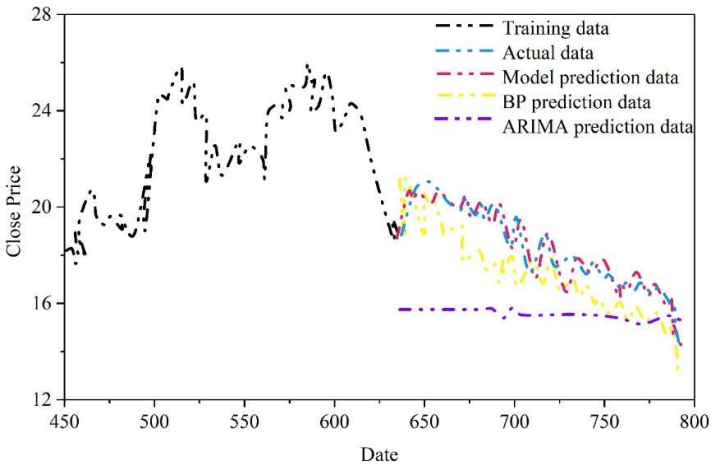

Fig. 7. 2020-2023 results in the prediction section of the model

Then the residual data of the model is tested, Figure 8 and Figure 9 show the residual series of the ARIMA model and the Ljung-Box test, respectively, and it is found that there is no autocorrelation in the residual series, and the fluctuation of the residual series is very small, and the prediction error is equal to about 0 after 2015, which indicates that the ARIMA model is more reasonable. The predicted GDP growth rate for the next four quarters using the ARIMA model ranges from 5.8% to 6.4%.

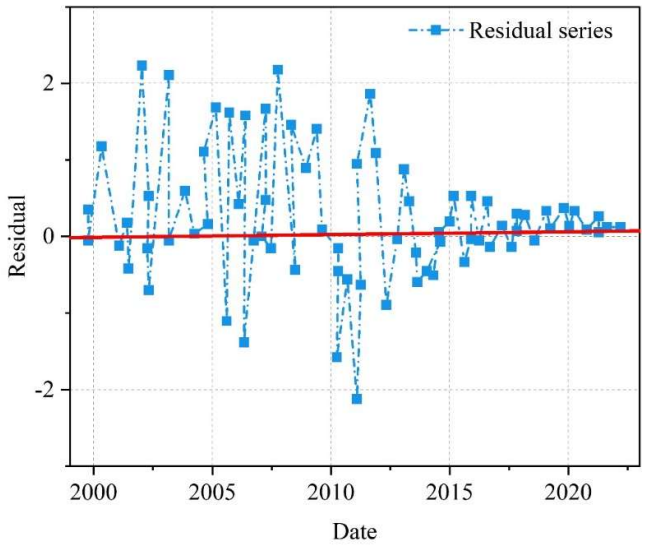

Fig. 8. The ARIMA model predicts the loss of GDP growth

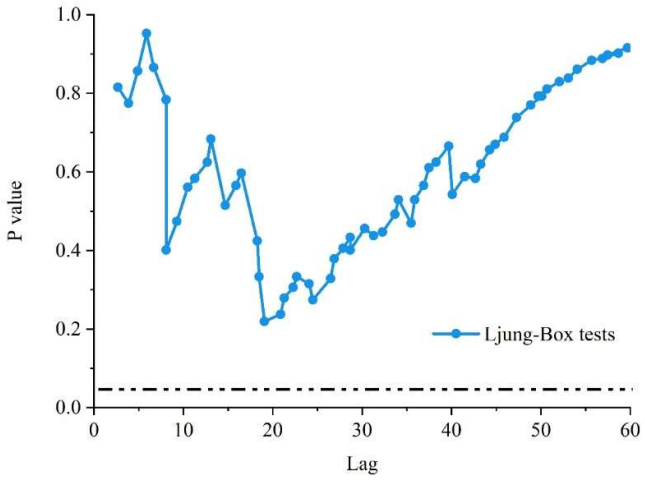

Fig. 9. The ARIMA model predicts the Ljung-box test for GDP growth rate

Figures 10 and 11 show the residual series of this model and the Ljung-Box test, respectively. It is found that there is no autocorrelation in the residual series of this paper's model, and the fluctuation of the residual series is very small, and the prediction error of the model is about equal to 0 after 2013, which indicates that the established neural network model is reasonable. The neural network model in this paper predicts the GDP of the next four quarters in the range of 5.8%-5.9%, and in terms of volatility, the prediction range of the model in this paper is more stable than that of the ARIMA model, and the range of prediction error is smaller than that of the ARIMA model.

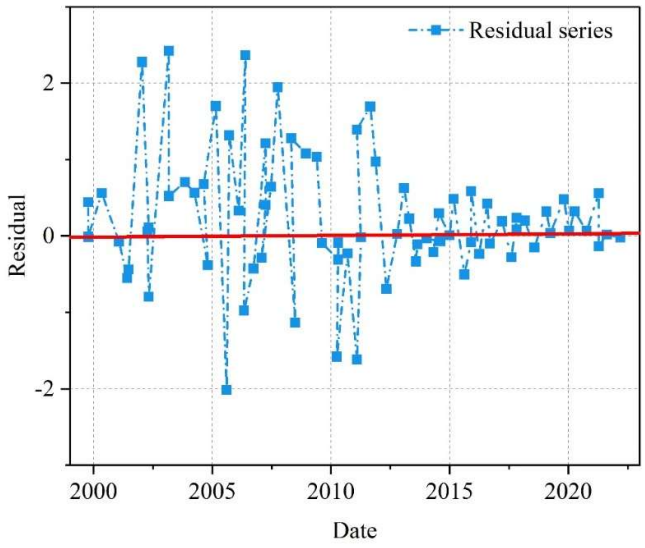

Fig. 10. This model predicts the loss of GDP growth

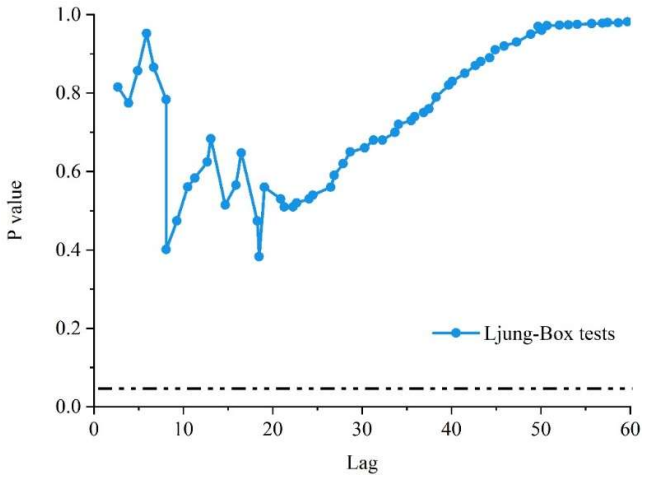

Fig. 11. This model predicts the Ljung-box test for GDP growth rate

4. Conclusion

This paper is dedicated to applying artificial intelligence technology to the field of macroeconomic forecasting and constructing a macroeconomic quantization method with high accuracy.

In terms of the impact of time span on the prediction performance of the model in this paper, the longer the time span and the more sufficient data in the training set, the more significant the improvement of the model's prediction ability, and the prediction error of the model is reduced from 10% to 3% when the time span is increased from 2020-2023 to 2019-2023. Therefore, the macroeconomic forecasting accuracy of the model can be improved by increasing the time span.

In terms of the performance comparison experiments of different forecasting models, the RMSE index of this paper's model performs the best in both the 2020-2023 and 2019-2023 time spans, with RMSE values of 0.0038 and 0.003, respectively, which indicates that this paper's model has a better macroeconomic forecasting accuracy, and it can be regarded as a reliable method for macroeconomic forecasting.

In terms of the autocorrelation of the residual series and its volatility of the ARIMA model and this paper's model, both models have been free of autocorrelation, and the fluctuation of the residual series is small, this paper's model is approximately equal to 0 in the model prediction error after 2013, whereas the prediction error of the ARIMA model tends to be close to 0 only after 2015, and the forecast range of this paper's model for the future period of the GDP is between 5.8%- 5.9%, which is smaller than the GDP prediction range of 5.8%-6.4% of the ARIMA model, showing again the excellent performance of this paper's model in economic prediction.

References

[1] Panagiotelis, A., Athanasopoulos, G., Hyndman, R. J., Jiang, B., & Vahid, F. (2019). Macroeconomic forecasting for Australia using a large number of predictors. International Journal of Forecasting, 35(2), 616-633.

[2] Knotek II, E. S., & Zaman, S. (2019). Financial nowcasts and their usefulness in macroeconomic forecasting. International Journal of Forecasting, 35(4), 1708-1724.

[3] Goulet Coulombe, P., Leroux, M., Stevanovic, D., & Surprenant, S. (2022). How is machine learning useful for macroeconomic forecasting?. Journal of Applied Econometrics, 37(5), 920-964.

[4] Ellingsen, J., Larsen, V. H., & Thorsrud, L. A. (2022). News media versus FRED-MD for macroeconomic forecasting. Journal of Applied Econometrics, 37(1), 63-81.

[5] Bok, B., Caratelli, D., Giannone, D., Sbordone, A. M., & Tambalotti, A. (2018). Macroeconomic nowcasting and forecasting with big data. Annual Review of Economics, 10(1), 615-643.

[6] Smalter Hall, A., & Cook, T. R. (2017). Macroeconomic indicator forecasting with deep neural networks. Federal Reserve Bank of Kansas City Working Paper, (17-11).

[7] Kurpayanidi, K. I. (2020). ON THE PROBLEM OF MACROECONOMIC ANALYSIS AND FORECASTING OF THE ECONOMY. Theoretical & Applied Science, (3), 1-6.

[8] Pettenuzzo, D., & Timmermann, A. (2017). Forecasting macroeconomic variables under model instability. Journal of Business & Economic Statistics, 35(2), 183-201.

[9] Hamilton, J. D. (2020). Time series analysis. Princeton university press.

[10] Ghysels, E., & Marcellino, M. (2018). Applied economic forecasting using time series methods. Oxford University Press.

[11] Fulcher, B. D. (2018). Feature-based time-series analysis. In Feature engineering for machine learning and data analytics (pp. 87-116). CRC press.

[12] Aminikhanghahi, S., & Cook, D. J. (2017). A survey of methods for time series change point detection. Knowledge and information systems, 51(2), 339-367.

[13] Wei, W. W. (2019). Multivariate time series analysis and applications. John Wiley & Sons.

[14] Petrusevich, D. A. (2019). Analysis of mathematical models used for econometrical time series forecasting. Russian Technological Journal, 7(2), 61-73.

[15] Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017). Convolutional neural networks for time series classification. Journal of systems engineering and electronics, 28(1), 162-169.

[16] Cai, Z., Huang, J., & Huang, L. (2018). Periodic orbit analysis for the delayed Filippov system. Proceedings of the American Mathematical Society, 146(11), 4667-4682.

[17] Ji, X., Martin, J. S., & Yao, Y. (2017). Macroeconomic risk and seasonality in momentum profits. Journal of Financial Markets, 36, 76-90.

[18] Kajol, K., Nath, M., Singh, R., Singh, H. R., & Das, A. K. (2020). Factors affecting seasonality in the stock market: A social network analysis approach. International Journal of Accounting & Finance Review, 5(4), 39-59.

[19] McAlinn, K., Aastveit, K. A., Nakajima, J., & West, M. (2020). Multivariate Bayesian predictive synthesis in macroeconomic forecasting. Journal of the American Statistical Association, 115(531), 1092-1110.

[20] D'Agostino, A., Gambetti, L., & Giannone, D. (2013). Macroeconomic forecasting and structural change. Journal of applied econometrics, 28(1), 82-101.

[21] Chen, S., & Ranciere, R. (2019). Financial information and macroeconomic forecasts. International Journal of Forecasting, 35(3), 1160-1174.

[22] Dendramis, Y., Kapetanios, G., & Marcellino, M. (2020). A similarity-based approach for macroeconomic forecasting. Journal of the Royal Statistical Society Series A: Statistics in Society, 183(3), 801-827.

[23] Bai, Y., Carriero, A., Clark, T. E., & Marcellino, M. (2022). Macroeconomic forecasting in a multi-country context. Journal of Applied Econometrics, 37(6), 1230-1255.

[24] Clark, T. E., & Ravazzolo, F. (2015). Macroeconomic forecasting performance under alternative specifications of time-varying volatility. Journal of Applied Econometrics, 30(4), 551-575.

[25] Chundi Jiang. (2024). African vulture optimized RNN algorithm maximum power point tracking (MPPT) controller for photovoltaic (PV) system. Measurement: Sensors101392-101392.

[26] Van Dai Vuong,Luong Ha Nguyen & James A. Goulet. (2025). Coupling LSTM neural networks and state-space models through analytically tractable inference. International Journal of Forecasting(1), 128-140.

[27] Pascal Riedel,Kaouther Belkilani,Manfred Reichert,Gerd Heilscher & Reinhold von Schwerin. (2024). Enhancing PV feed-in power forecasting through federated learning with differential privacy using LSTM and GRU. Energy and AI100452-100452.