WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 3, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v3n2.33
Journal homepage: https://wepub.org/index.php/IJCSIT/index

A study of Machine Learning for Prediction in Panel Data

Shenglong Huang [1], Lingyun Hu [2, *]

[1] School of Statistics and Applied Mathematics, Anhui University of Finance and Economics, Bengbu, 233030, China

[2] School of Management Science and Engineering, Anhui University of Finance and Economics, Bengbu, 233030, China

*Corresponding Author: misshz@sina.com

ABSTRACT

With the development of the times, the prediction study of machine learning in panel data is becoming more and more extensive, this paper adopts the RNN algorithm and XGBoost algorithm to carry out the prediction study on the quarterly GDP panel data of 31 provinces and cities in China from the 1st quarter of 2005 to the 4th quarter of 2023, and compares and analyses the two methods. In the forecasting study, this paper considers the influence of geographic location in the panel data, and the results show that there are significant regional differences in the RNN algorithm in forecasting, and the eastern coastal and inland provinces with stronger economies perform better, for example, the training set correlation coefficient of Guangdong province is as high as 0.8052, followed by Anhui and Hubei. However, the Qinghai-Xizang region performs poorly and is at risk of overfitting. The XGBoost algorithm, on the other hand, shows high correlation coefficients on both the training and test sets in most provinces and cities, especially in Beijing, Tianjin and Hebei, where the correlation coefficients are above 0.9, showing good prediction results. In terms of mean square error, the MSE of the training set is generally smaller than that of the test set, indicating that the predicted values of some regions have a large deviation from the actual values. In terms of the mean absolute percentage error (MAPE), the MAPE of most provinces and cities is below 1%, which indicates that the relative error of prediction is small. Comprehensive analysis shows that XGBoost is good at dealing with nonlinear relationships and complex feature interactions, and is especially suitable for capturing nonlinear geolocation features, while RNN may be more effective in dealing with temporal geolocation features; XGBoost has a strong fitting ability and is suitable for sparse data, while RNN needs more data to learn effective representations; XGBoost has less need for hyperparameter tuning, while RNN is is more sensitive to hyperparameters.

KEYWORDS

Machine learning; RNN; XGBoost; Panel data; Prediction

1. INTRODUCTION

Panel data are repeated observations of the same individuals or units over time. It contains observations spanning multiple time periods and is therefore characterised by both time series data and cross-sectional data. Machine learning has significant potential for application in panel data forecasting research. Traditional panel data models, such as fixed-effects and random-effects models, are able to address panel data to some extent, but they tend to have limited ability to model non-linear relationships in the data. Machine learning research in panel data analysis mainly focuses on cluster analysis, model testing, and feature extraction. Therefore, the research direction of this paper not only has theoretical value, but also has practical application value. It contributes to a more in-depth

understanding of the economic development of China's regions, provides a scientific basis for policy formulation, and promotes the coordinated development of regional economy. The marginal contributions of this paper are mainly reflected in the following aspects: first, this paper further improves the application of machine learning in panel data forecasting. By introducing advanced machine learning algorithms, it successfully overcomes the limitations of traditional statistical methods in panel data prediction and improves the prediction accuracy and efficiency. Secondly, this paper delves into the influence of geographic location in panel data. By fully considering the geographic location factor, the patterns and trends behind the data can be captured more accurately, so that more accurate predictions can be made.

The literature on panel data forecasting, particularly where the cross-sectional dimension is large and the time-series dimension is short, is very sparse. They provide a theorem establishing a ratio-optimality property for a nonparametric kernel estimator of the Tweedie correction and consider implementations based on estimating mixtures of normals and nonparametric MLE in Monte Carlo simulations. Their practical application demonstrates that these forecasting techniques work well in practice and may be useful for executing bank stress tests [1, 2].

Qu, Ritong, Allan Timmermann, and Yinchu Zhu [3] have developed novel methods to assess the equality of predictive accuracy across panels of forecasts, utilizing both time series and cross-sectional data dimensions. Our study explores various testing approaches, including general tests that assess overall forecasting performance across all time periods and individual units, as well as subset (cluster) tests that concentrate on specific time periods or groups of units. These methodologies are applied in an empirical analysis comparing International Monetary Fund (IMF) forecasts of country-level real GDP growth and inflation to forecasts derived from private-sector surveys and a basic time-series model.

They contribute to this literature by developing an empirical Bayes predictor that uses cross-sectional information in the panel to construct a prior distribution, forming a posterior mean predictor for each cross-sectional unit. The importance of this prior increases as the time-series dimension shortens and parameter heterogeneity decreases, leading to significant forecasting gains when using the posterior mean predictor instead of a plug-in predictor. They apply this idea to linear models with Gaussian innovations based on Tweedie’s posterior mean formula, implementing it by estimating the cross-sectional distribution of sufficient statistics for the heterogeneous coefficients in the forecast model [4-7].

2. INTRODUCTION TO THE MODEL

2.1. Neural Network Model

The neural network model is a mathematical model inspired by the structure of the biological nervous system, which is used to simulate the connection between neurons and the information transfer process in the human brain. Its basic principles involve information transfer between neurons, weight adjustment and model training. The basic building blocks of neural networks are neurons. The neuron receives inputs from other neurons, performs weighted summation, and outputs the result through an activation function. Commonly used activation functions are sigmoid, ReLU, etc. The connections between neurons are described by weights, the value of which determines how much the input signal affects the neuron. The weights indicate the strength and direction of information transfer between neurons. In neural networks, the flow of information from the input layer to the output layer is called forward propagation. During forward propagation, the input signals are passed to the hidden and output layers through connections and weights, which ultimately produce the output of the network. In order for the neural network to learn and adapt to the data, the connection weights need to be adjusted through the back propagation algorithm. Depending on the structure and connections of the network, several types of neural networks can be constructed, such as multilayer perceptron (MLP),

convolutional neural network (CNN), recurrent neural network (RNN), and so on. In a nutshell, neural network models simulate the process of information transfer through the connections and weight adjustments between neurons, and learn the complex relationship between inputs and outputs through training data. The basic principle is to continuously adjust the weights through the process of repeated iterations in order to make the model's prediction as close as possible to the actual value.

2.2. Recurrent Neural Networks

Recurrent Neural Network (RNN) is a neural network structure that is particularly suitable for processing sequence data, which includes time series, text, speech, etc. Unlike traditional feed-forward neural networks, RNNs have recurrent connections that allow information to be passed continuously within the network. The recurrent connections in RNNs allow information to be continuously passed inside the network. At each time step, the network receives the current input and the hidden state of the previous time step as input and produces the output and hidden state of the current time step. This structure allows the RNN to process sequence data of arbitrary length and to capture temporal correlations in the sequences.The hidden state of the RNN contains a summary of the network's information about the sequences observed in the past. The hidden state is updated at each time step and can be thought of as the network's "memory" at the current time step.The RNN uses the same weight parameters at each time step, which means that the network processes the sequences in the same way, and thus has the same pattern recognition ability when processing data at different time steps. Similar to other neural networks, RNNs are trained by a backpropagation algorithm. At each time step, the gradient of the loss function with respect to the network parameters is computed and the parameters are updated according to the gradient. However, since RNNs are prone to the problems of gradient vanishing or gradient explosion when dealing with long sequential data, the gradient trimming technique is usually used to control the size of the gradient during training to avoid these problems.The RNN algorithm process is as follows: denotes the input of the first step, and is the state of the first step of the hidden layer, which is a memory unit of the network. Based on the output of the current input layer and the state of the hidden layer at the previous moment, the calculation is carried out as shown in Equation (1). Where is the connection matrix of the input layer, is the weight matrix of the hidden layer from the previous moment to the next moment, and is generally a nonlinear activation function, such as tanh or ReLU.

$$S_t = f(UX_t + WS_{t-1}) \tag{1}$$

Is the output of the first step. The output layer is a fully connected layer, i.e., each node of it and each node of the implicit layer are connected to each other, is the connection matrix of the output layer, and is the activation function.

$$O_t = g(V * S_t) \tag{2}$$

If equation (1) is looped into equation (2) it can be obtained:

$$\begin{aligned} O_t &= g(VS_t) \\ &= Vf(UX_t + WS_{t-1}) \\ &= Vf(UX_t + Wf(UX_{t-1} + WS_{t-2})) \\ &= Vf(UX_t + Wf(UX_{t-1} + Wf(UX_{t-2} + WS_{t-3}))) \\ &= Vf(UX_t + Wf(UX_{t-1} + Wf(UX_{t-2} + Wf(UX_{t-3} + \ldots)))) \end{aligned} \tag{3}$$

As can be seen from equation (3), the output values of the recurrent neural network are associated with the historical input values of the previous moments, which allows it to look forward to any number of input values.

The backpropagation algorithm for RNN is BPTT, and here the notation is changed to, and the specific algorithm is as follows:

$$S_t = \tanh\left(UX_t + WS_{t-1}\right) \tag{4}$$

$$\hat{y}_t = softmax\left(VS_t\right) \tag{5}$$

Again define the loss value as a cross-entropy loss as follows:

$$E_t\left(y_t, \hat{y}_t\right) = -y_t \log\left(\hat{y}_t\right) \tag{6}$$

$$E(y, \hat{y}) = \sum_t E_t\left(y_t, \hat{y}_t\right) = -\sum_t y_t \log \hat{y}_t \tag{7}$$

Here, denotes the moment The correct word, denotes the prediction. The following goal is to compute the gradient of the error values with respect to the parameters , as well as the parameters learnt well with stochastic gradient descent, summing up the gradient values for each moment for each training sample:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{8}$$

In order to calculate the gradient, you can use the chain derivation rule, which focuses on propagating the error backwards using a backpropagation algorithm. As an example, there are:

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} = \left(\hat{y}_3 - y_3\right) \otimes s_3 \tag{9}$$

2.3. XGBoost Model

XGBoost is an efficient and flexible machine learning algorithm for regression and classification problems. Developed by Tianqi Chen et al, it is an implementation of gradient boosting trees. It builds a powerful integrated model by combining multiple weak learners (decision trees) and adopts the idea of gradient boosting, which gradually improves the performance of the model by successively training and optimising a series of decision tree models and continuously correcting the residuals or errors of the previous models. In addition, XGBoost controls the complexity of the model by adding regularisation terms (e.g. L1 and L2 regularisation) to prevent overfitting. These regularisation terms help improve the generalisation ability of the model and reduce the model's error on unseen data. Meanwhile, XGBoost not only supports a variety of custom loss functions and evaluation metrics, which makes it adaptable to different types of problems and data, but also supports parallel computation, which enables it to efficiently deal with large-scale datasets.The regression algorithm process of XGBoost is as follows:

Input and output training set: $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, Final Return Tree $f(x)$ model initialisation Set and, assuming that the initialised weak learner $f_0(x_i) = c$, cost function $\cos t(c) = \sum_{i=1}^{n} L(y_i, c)$, so:

$$c = \arg_c \min \sum_{i=1}^{n} L(y_i, c) \tag{10}$$

When the loss function is a squared loss function, there are:

$$c = \frac{1}{n} \sum_{i=1}^{n} L(y_i, c) \tag{11}$$

Iteration, $t = 1, 2, ..., T$ iteration, For the first sample, its negative gradient $r_{t,i} = -\frac{\partial L(y_i, f(x))}{\partial f(x)}|_{f(x)=f_{t-1}(x_i)}$, In this paper, the squared loss function is chosen $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$, so:

$$r_{t,i} = y_i - f_{t-1}(x_i) \tag{12}$$

Then construct a new dataset using negative gradients $D_t = \{(x_1, r_{t,1}), (x_2, r_{t,2}), ..., (x_n, r_{t,n})\}$, count $g_{t,i}$ and $h_{t,i}$, $i = 1, 2, ..., n$, concurrent calculation G_j and H_j . Generate a new round of decision trees $w_t(x)$. The area of the leaf node is $R_{t,j}$, $j = 1, 2, ..., J_t$, take a value of $w_{t,j} = -\frac{G_j}{H_j + \lambda}$. The Renewal of Strong Learning period is:

$$f_t(x) = f_{t-1}(x) + w_t(x) = f_{t-1}(x) + \sum_{j=1}^{J_t} w_{t,j} I(x \in R_{t,j}) \tag{13}$$

Until the condition is satisfied, the iteration stops, at which point the final learner is:

$$f(x) = f_T(x) = f_0(x) + \sum_{t=1}^{T} \sum_{j=1}^{J_t} w_{t,j} I(x \in R_{t,j}) \tag{14}$$

3. PATH TO REALISATION

3.1. Code Design

Code design before proceeding with the algorithmic process allows for a clearer understanding of the modelling process and objectives. In the research of this paper, the geographic location processing and parameter selection of the panel data is very critical, and the following discussion is carried out to address this issue.

3.1.1. Geolocation Treatment of Panel Data

It is important to consider the impact of geographic location on panel data forecasts because geographic location itself contains a wealth of information that can reflect inter-regional variability. In panel data, geographic locations between provinces and cities are not just simple geographical coordinates, but also cover a complex set of geographical, climatic and human factors. Geographic location can reflect climatic differences. Different regions have different climatic conditions, such as temperature, precipitation, seasonality, etc. These factors have a significant impact on the development of certain industries, crop growth and energy consumption. Geographical location is also closely related to the level of economic development. Different regions have different levels of economic development, including differences in population density, industrial structure, and

infrastructure development. These factors affect the consumption level, industrial structure, employment, etc., which in turn have an impact on the characteristics of the panel data. In addition, geographic location reflects the cultural background and social environment among regions. Factors such as cultural traditions, folk customs and social habits in different regions will also have an impact on the panel data. Therefore, adding geographic location information as a feature to the panel data prediction model can more comprehensively take into account the impact of geographic factors on the data and improve the accuracy and interpretability of the prediction.

Using the geopy library to calculate distances between geographic locations as features to be added to the learning serves the following purposes: (1) A more comprehensive feature representation. Distance between geographic locations is a more intuitive and objective representation of features, which can help the model understand the relationships and differences between different regions more comprehensively. (2) Consider geographic correlation. The distance between geographic locations can reflect the geographic correlation between different regions. For example, closer regions may have more similar characteristics such as climate and level of economic development, while more distant regions may have greater differences. Therefore, adding the distance between geographic locations as a feature to learning can help the model better consider geographic correlations. (3) Enhance model generalisation. Distance between geographic locations as a feature can provide additional information that can help the model more accurately capture relationships between data. Doing so helps enhance the model's ability to generalise so that it can make accurate predictions even when faced with new data in the future. (4) Improve prediction accuracy. By considering the distance between geographic locations as a feature to be added to the learning, it allows the model to more accurately capture the impact of geographic factors on the data, thus improving the precision and accuracy of the predictions.

The inclusion of distances between geographic locations as features in the learning of panel data not only allows the model to consider the effects of geographic factors more comprehensively, but also captures geospatial correlations. This approach not only helps the model to understand the relationship between the data more accurately, but also enhances the model's understanding of the geographic context, which improves the model's ability to generalise. By introducing geographic distance features, the model is better able to predict changes and trends in geographic space, which in turn leads to more precise and accurate predictions. This integrated approach of considering geographic factors not only enriches the feature space of the model, but also improves the overall performance of the model, providing strong support for more effective prediction and decision-making.

3.1.2. Parameter Selection for RNN Models

Parameter selection is crucial in neural network models. It directly affects the performance, training efficiency and generalisation ability of the model. By choosing appropriate parameters such as learning rate, number of hidden layers, and number of neurons per layer, the complexity of the model can be controlled and the convergence speed and stability of the model during training can be improved. Different problems and datasets may require different parameter settings, so reasonable choices and adjustments need to be made according to the specific situation in order to obtain the best model performance. In this paper, we need to select the following parameters: (1) Learning rate (learning rate) is an important hyperparameter to control the step size of parameter update, which directly affects the convergence speed and stability of the model in the training process. Too large a learning rate may cause the model to fail to converge, while too small a learning rate may slow down the training speed, or even cause the model to fall into the local optimal solution. Therefore, it is necessary to choose an appropriate learning rate according to the specific situation and adjust it according to the performance of the model during the training process. (2) The number of RNN layers and the dimension of hidden states. These two parameters directly affect the model's expressive ability and learning ability. Increasing the number of RNN layers and the dimension of hidden states can increase the complexity of the model and improve its ability to fit the data, but it will also increase

the training time and resource consumption of the model. Therefore, the choice needs to be made based on the complexity of the task and the characteristics of the data, and experiments may be required to determine the optimal parameter settings. (3) Dimensional order of input data. The dimensional order of the input data determines how the data is represented in the model, and for RNNs, the sequence length is usually placed in the second dimension, i.e., (batch_size, seq_length, input_size), in order for the model to process the sequence data correctly. Therefore, it is necessary to ensure that the dimensional order of the input data is set correctly when constructing the model. (4) Number of training iterations (epoch). The number of training iterations determines the number of times the model is updated on the entire dataset, which directly affects the convergence speed and training time of the model. Typically, the more training iterations, the better the model may perform, but it also increases the risk of overfitting. Therefore, an appropriate choice needs to be made based on the performance of the model and the training time.

The learning rate was set to 0.0001, the number of layers of the RNN was set to 1, the dimension of the hidden state of the RNN layer was 32, and the number of iterations for training was 1000.

3.1.3. Parameter selection for the XGBoost model

The parameter selection of the XGBoost model requires more parameters to be set compared to the RNN model and is relatively more complex. They can be divided into the following categories: (1) Maximum depth. Define the maximum depth of each tree, a larger value will increase the complexity of the model, but it will also lead to overfitting, so it is usually adjusted by cross-validation. (2) Learning rate. Controls the weight adjustment step for each tree, smaller values make model training more cautious and require more trees to fit the data. (3) Number of iterations. Generally increasing the number of iterations, i.e. increasing the number of trees will improve the model performance, but it will also increase the training time. And it is adjusted at the same time as the learning rate, and there is generally an inverse relationship between them. (4) Gain value. The decreasing threshold of the loss function when the control node splits. As with maximum depth, larger values lead to more conservative algorithms that are more prone to overfitting. (5) Sample proportion and feature proportion. Smaller values of these two parameters lead to more randomness and help prevent overfitting. (6) Regularisation parameters. By adjusting the two parameters reg_alpha and reg_lambda, the weight of the regularisation can be controlled, thus affecting the complexity of the model. Increasing these values strengthens the effect of regularisation, which in turn reduces the complexity of the model and the risk of overfitting.

3.2. Algorithm Flow

3.2.1. RNN Algorithm Flow

An Excel file is first read, which contains the raw data, including time, longitude, latitude and corresponding GDP values, as well as province and time information. Next, the data was preprocessed, including feature extraction and distance calculation. Distances between geographic locations were calculated using the geodesic function and the distances were added to the features. Then, the features and labels were normalised to range from 0 to 1. Once the data preprocessing was completed, the dataset was divided into a training set and a test set for model training and evaluation. A neural network model containing fully connected and RNN layers was constructed and trained using the training set. During training, the mean square error was used as the loss function and the Adam optimiser was used for parameter optimisation. After the training was completed, the performance of the models on the training and test sets was evaluated, including the calculation of metrics such as correlation coefficient, mean square error and mean absolute percentage error, and the results were saved in an Excel table. At the same time, the comparison between the true and predicted values of the training and test sets for each province was shown visually, and the results were saved as image files. Finally, printouts of the prediction accuracy metrics for each province were made for further analysis and evaluation.

3.2.2. XGBoost Algorithm Flow

An Excel file containing the raw data was first read via the pd.read_excel() method. The characteristics of the data included time, longitude, latitude and corresponding GDP values, in addition to province and time information. Next, distances between geographic locations were calculated using the geodesic function, and the calculated distances were added as a new feature to the original features. The features and labels were then normalised to a range between 0 and 1 using MinMaxScaler. Once the data preprocessing was complete, the data set was divided into a training set and a test set using the train_test_split method for subsequent model training and evaluation. Next, an XGBoost regressor was initialised using the XGBRegressor class and trained using the training set. Then, the trained XGBoost model was used to predict the training and test sets and the prediction results were obtained. Subsequently, for each province, metrics such as correlation coefficient, mean square error, and mean absolute percentage error were calculated for the training and test sets, and the results were saved to a dictionary, which was converted to a DataFrame and saved to an Excel file. Finally, the real and predicted values of the training and test sets for each province were visually compared, and the results were saved as a picture file in order to visually observe the prediction effect of the model.

4. EMPIRICAL RSEARCH

In this paper, the quarterly GDP of 31 provinces and cities in China is selected as the experimental data from the 1st quarter of 2005 to the 4th quarter of 2023, and the RNN algorithm and the XGBoost algorithm are run separately using Spyder software.

4.1. Analysis of the results of the RNN algorithm

According to Table 1, it is obvious to see that there are significant regional differences in the prediction results, and the models of the eastern coastal cities (e.g., Shanghai, Jiangsu, Zhejiang, Fujian, and Guangdong) and some inland provinces with stronger economies (e.g., Hubei, Hunan, Anhui, and Henan) perform better, which may be related to the frequent economic activities in these regions and the strong regularity of the data. The training set correlation coefficient of Guangdong province reaches 0.8052, followed by Anhui and Hubei provinces. In contrast, regions such as Xizang, Qinghai, and Ningxia perform poorly, which may be due to the fact that their economies are small and affected by special geographic environments and industrial structures, and the GDP fluctuation patterns are more different from those of other regions, which makes it difficult for the model to capture the patterns. In terms of the model's generalisation ability, the high correlation coefficient on the training set indicates that the model is able to learn the patterns in the historical data better. However, the relatively low correlation coefficients and high MSE and MAPE values on the test set indicate that the model is at risk of overfitting when faced with new data. In particular, the negative correlation coefficients for Inner Mongolia and Xizang imply that the model may not properly understand the economic characteristics of these regions or fail to effectively predict economic trends. Observing the relationship between the correlation coefficient and MSE and MAPE, it can be found that not in all cases the higher the correlation coefficient the smaller the error. For example, despite the high correlation coefficients, the MSE and MAPE of regions such as Shanghai and Anhui are also relatively high, which suggests that the correlation coefficients alone may not be able to fully evaluate the model performance.

Table 1. RNN model training results.

Province	Training set correlation coefficient	Test set correlation coefficient	Training set MSE	Test set MSE	Training set MAPE	Test set MAPE
Beijing	0.6818	0.4999	82900000	118730000	0.5266	0.4711
Tianjin	0.6144	0.5574	66646000	88747000	1.3387	2.2934
Hebei	0.6938	0.3573	55485000	123360000	0.5367	1.0294
Shanxi	0.5441	0.7019	58160000	39345000	1.6358	1.1448
Inner Mongolia	0.4332	-0.1769	37335000	37509000	0.8768	0.6358
Liaoning	0.6089	0.0860	44555000	64897000	0.6164	0.6724
Jilin	0.4148	0.4182	19072000	20396000	1.1732	1.1360
Heilongjiang	0.3577	0.2754	16508000	19761000	0.6120	1.0403
Shanghai	0.7037	0.7731	128070000	85429000	0.9573	0.9738
Jiangsu	0.7429	0.7013	659820000	685830000	0.4604	0.5037
Zhejiang	0.7305	0.7390	163750000	251760000	0.5491	0.3511
Anhui	0.7498	0.7075	139300000	173830000	1.4294	1.5568
Fujian	0.7175	0.7452	95824000	45224000	0.6918	0.8968
Jiangxi	0.6862	0.3965	81393000	96347000	1.3703	1.5414
Shandong	0.7442	0.5036	263130000	247210000	0.5412	0.5662
Henan	0.7234	0.0552	134120000	361530000	0.5658	0.9520
Hubei	0.7475	0.5326	82742000	165260000	0.6934	1.2369
Hunan	0.7240	0.7351	93401000	26679000	0.6080	0.3710
Guangdong	0.8052	0.5472	459450000	263720000	0.5539	0.6754
Guangxi	0.5333	0.1778	45363000	51250000	0.9572	0.8294
Hainan	0.4307	0.3735	30293000	55963000	3.6152	4.5491
Chongqing	0.7088	0.6102	50516000	96306000	1.2596	1.3439
Sichuan	0.6977	0.8098	122390000	117210000	0.5206	0.4646
Guizhou	0.5943	0.6918	54043000	103270000	3.0729	4.1696
Yunnan	0.6140	0.6379	51074000	68635000	1.2013	2.4394
Xizang	-0.0970	-0.2047	10689000	20053000	10.4510	5.5943
Shaanxi	0.6905	0.6578	35933000	40694000	0.7954	1.3776
Gansu	0.5596	0.6742	7017200	7625300	0.9364	0.9671
Qinghai	0.3366	0.4196	4980400	5240500	2.7855	2.9246
Ningxia	0.4280	0.3076	3757300	3985800	2.3680	2.1190
Xinjiang	0.5859	0.7544	12021000	16291000	1.0922	0.8067

Note: Training set MSE and test set MSE are scientific notation

4.2. Analysis of XGBoost Algorithm Results

In terms of correlation coefficients, the correlation coefficients of the training and test sets in most of the provinces and municipalities are very high, close to 1, which indicates that the model fit and prediction effect are generally good. In particular, the correlation coefficients of Beijing, Tianjin, Hebei, Inner Mongolia, Shanghai, Jiangsu, Zhejiang, Fujian, Henan, Hubei, Hunan, Guangdong, Guangxi, Chongqing, Sichuan, Guizhou, Yunnan, Shaanxi, Gansu, Qinghai, Ningxia, and Xinjiang

are above 0.9, which show very good prediction effects. In terms of mean square error (MSE), the MSE of the training set is generally smaller than that of the test set because the model is fitted on the training set. Among them, the MSEs of the test sets in Beijing, Tianjin, Hebei, Shanxi, Inner Mongolia, Liaoning, Jilin, Heilongjiang, Shanghai, Jiangsu, Zhejiang, Anhui, Fujian, Jiangxi, Shandong, Henan, Hubei, Hunan, Guangdong, Guangxi, Hainan, Chongqing, Sichuan, Guizhou, Yunnan, Xizang, Shaanxi, Gansu, Qinghai, Ningxia, and Xinjiang are all over the 6th power of 10, indicating that the predicted values of these regions have a large deviation. In particular, the test set MSE of Xizang reaches 2.62 times 10 to the seventh power, which is much higher than that of other regions, indicating that the prediction effect in Xizang is poorer. In terms of the mean absolute percentage error (MAPE), the MAPEs of most provinces and cities are below 1 per cent, which indicates that the relative error of prediction is small. However, the MAPE of the test set in Xizang reaches 12.4034%, which is much higher than that in other regions, which further confirms the poor prediction effect in Xizang.

Table 2. XGBoost model training results.

Province	Training set correlation coefficient	Test set correlation coefficient	Training set MSE	Test set MSE	Training set MAPE	Test set MAPE
Beijing	0.9984	0.9204	345330	19526000	0.0484	0.4869
Tianjin	0.9889	0.9333	1223400	38445000	0.1708	2.4744
Hebei	0.9974	0.9526	579750	16655000	0.0499	0.9289
Shanxi	0.9923	0.9630	382550	4657300	0.0946	0.6695
Inner Mongolia	0.9944	0.9463	450760	2994300	0.0877	0.6672
Liaoning	0.9974	0.9590	399790	7533000	0.0599	0.5516
Jilin	0.9874	0.9751	892410	5151700	0.2095	1.8199
Heilongjiang	0.9941	0.9736	400470	1421400	0.1079	1.5714
Shanghai	0.9965	0.9955	1296800	10858000	0.0655	0.5899
Jiangsu	0.9931	0.8929	17165000	249090000	0.1314	0.5651
Zhejiang	0.9960	0.9849	4069000	32165000	0.1175	0.3966
Anhui	0.9932	0.9163	3032200	40200000	0.1980	0.9735
Fujian	0.9980	0.9924	908600	4835700	0.0965	1.1447
Jiangxi	0.9822	0.8493	5362100	25253000	0.3307	1.9087
Shandong	0.9948	0.8949	7742400	74183000	0.1247	0.5254
Henan	0.9978	0.9697	1252200	8406300	0.0604	0.6562
Hubei	0.9984	0.9881	588980	6517400	0.0583	1.0160
Hunan	0.9994	0.9058	214390	14381000	0.0364	0.7062
Guangdong	0.9943	0.8044	18246000	119990000	0.1320	0.5141
Guangxi	0.9972	0.9769	289570	1822200	0.0661	0.5368
Hainan	0.9176	0.8380	1352800	77712000	0.6803	2.8784
Chongqing	0.9981	0.9927	245920	366730	0.0668	1.0532
Sichuan	0.9906	0.8905	4680600	64245000	0.1774	0.4428
Guizhou	0.9925	0.9191	733890	3807300	0.1808	2.3250
Yunnan	0.9987	0.9064	145250	6881800	0.0595	0.7099
Xizang	0.7704	0.8227	880720	26206000	2.3154	12.4034
Shaanxi	0.9977	0.9740	342140	3487700	0.0618	1.4461
Gansu	0.9828	0.9642	416300	2818500	0.1840	1.6695
Qinghai	0.8436	0.8655	1053400	7000800	0.9932	8.4754
Ningxia	0.8933	0.8860	1137100	10872000	0.9895	7.1679
Xinjiang	0.9957	0.7917	165220	11292000	0.0864	1.0719

4.3. Comprehensive Analysis

The performance of RNN and XGBoost models in different regions varies significantly, with better prediction results in the eastern coastal cities (e.g., Shanghai, Jiangsu, Zhejiang, Fujian, and Guangdong) and some inland provinces with stronger economies (e.g., Hubei, Hunan, Anhui, and Henan), which is related to the frequent economic activities in these regions and the strong regularity of the data.The RNN model performs excellently in capturing the time-dependence of time series data. dependence, and the high correlation coefficients on the training set indicate that it is able to learn the patterns in the historical data better, but there are high MSEs and MAPEs on the test set, especially in Inner Mongolia and Xizang, showing the risk of overfitting and inaccurate prediction. In contrast, the XGBoost model shows better fit and prediction with correlation coefficients close to 1 for both the training and test sets in most regions, but still suffers from poor prediction in the test set with high MSE and MAPE in some regions such as Xizang. Overall, XGBoost outperforms RNN in most regions, but both models have some prediction errors in economically underdeveloped regions, and further data preprocessing and model optimisation are needed to improve the prediction accuracy and generalisation ability.

5. CONCLUSION

Through the above empirical study, the following conclusions are obtained:(1) XGBoost is a tree-based model suitable for dealing with nonlinear relationships and complex feature interactions. When the effects of geolocation features are nonlinear, XGBoost is more suitable for capturing such relationships. Whereas the RNN model is suitable for sequential data, it performs well on continuous data such as time series or text. In the case where the effect of geolocation features is non-linear, the effect is weaker than XGBoost, and if the geolocation features are time-series, RNN may perform better; (2) RNN is particularly suitable for processing sequential or time-series data because it can capture the time dependence of the data through the hidden layer state. The core of RNN lies in its "memory capability", which can update the hidden state according to the information of the input sequence and transfer the information between time steps; (3) XGBoost usually needs less hyper-parameter tuning to achieve better performance, while RNN may be more sensitive to hyper-parameters, such as the network structure, learning rate, etc., and needs to be tuned continuously. In order to further improve the prediction accuracy of the model, we will continue to explore the objective selection method of the model parameters in depth in the later study and form a new model by combining the advantages of the RNN model and the XGBoost model.

ACKNOWLEGEMENTS

This work is supported by Anhui Provincial Philosophy and Social Science Planning (Project number: AHSKY2020D42); Anhui Province Key Project in Natural Science for Universities (Project number:2022AH050602); Anhui University of Finance and Economics Research Fund Key Project (Project number: ACKYB23011).

REFERENCES

[1] Baltagi, Badi H. "Forecasting with panel data." Journal of forecasting 27.2 (2008): 153-173.

[2] Liu, Laura, Hyungsik Roger Moon, and Frank Schorfheide. "Forecasting with dynamic panel data models." Econometrica 88.1 (2020): 171-201.

[3] Qu, Ritong, Allan Timmermann, and Yinchu Zhu. "Comparing forecasting performance with panel data." International journal of forecasting 40.3 (2024): 918-941.

[4] Baltagi, Badi H. "Panel data forecasting." Handbook of economic forecasting 2 (2013): 995-1024.

[5] Timmermann, Allan, and Yinchu Zhu. "Comparing forecasting performance with panel data." (2019).

[6] Baltagi, Badi H., Bernard Fingleton, and Alain Pirotte. "Estimating and forecasting with a dynamic spatial panel data model." Oxford Bulletin of Economics and Statistics 76.1 (2014): 112-138.

[7] Baltagi, Badi H. "Forecasting with panel data." Center for Policy Research Working Paper 91 (2007).