v. 5, n. 5, 2025

Journal of Agricultural Sciences Research

Acceptance date: 03/06/2025

PREDICTIVE ANALYSIS OF THE AGRICULTURAL PRICE INDEX USING ARTIFICIAL INTELLIGENCE

José Luis Sosa Sánche
https://orcid.org/0000-0001-7610-1463

Darío Martínez Martínez
https://orcid.org/0009-0001-9590-287X

Harry Joel López Hereña
https://orcid.org/0000-0003-0162-7895

Raúl Delfín Cóndor Bedoya
National University of Engineering, Postgraduate
https://orcid.org/0000-0001-6111-2071

Daniel Alcides Carrión
National University

Abstract: The research addresses the problem of price volatility in the Peruvian agricultural sector, which affects economic stability and food security. The main objective is to develop a predictive model using LSTM neural networks to anticipate price variations and provide useful tools for decision making in the sector. The methodology included the normalization of historical data obtained from the Ministry of Agrarian Development and Irrigation, followed by the implementation and training of an LSTM model, evaluated using the root mean square error (RMSE). The results showed a training RMSE of 1.9904 and a significantly higher test RMSE of 48.0675, indicating a possible over-fitting of the model. The future price index predictions were 144.0155 and 124.3000, suggesting downward trends not fully aligned with historical data. In conclusion, although the LSTM model was able to capture patterns in the training data, its generalizability was limited, highlighting the need to improve the model and data quality to obtain more accurate and robust predictions in agriculture.

INTRODUCTION

The price index is a statistical measure that reflects the average change in the prices of a set of goods and services between two time periods. In the agricultural sector, this index is used to monitor changes in the prices of products such as grains, fruits, vegetables, livestock, milk, among others.

For Morales (2020), the price index is fundamental for the formulation of economic and agricultural policies. It allows decision-makers to identify inflationary or deflationary trends and take corrective measures to stabilize the economy. Morales emphasizes that in a country like Peru, where the agricultural sector plays a significant role in the economy, maintaining adequate price control is crucial for food security and the welfare of the population.

On the other hand, Gómez (2019) highlights the usefulness of the price index for agro-livestock producers. This indicator allows them to adjust their production and marketing strategies based on price trends. Gómez argues that by anticipating changes in , farmers can make informed decisions about which crops to plant and when to sell their products, thus optimizing their income.

Finally, Rodriguez (2021) points out that the price index is essential for consumers, since it provides them with information on fluctuations in the cost of living. In the Peruvian context, where families spend a considerable part of their income on food purchases, knowing the evolution of prices is vital for managing the family budget.

On the other hand, price formation in the agricultural sector is influenced by several factors. Supply, which is inelastic in the short term due to biological cycles and physical limitations, can be affected by climate, diseases and agricultural policies. Demand, also inelastic because food is a staple, varies with income and consumer preferences. Production costs, including inputs and transportation, directly impact prices. Government policies, such as subsidies and tariffs, together with the opening of international markets and trade agreements, also play a crucial role in determining prices in this sector.

METHODOLOGY

The agricultural price index is calculated using statistical methods such as the Laspeyres index, the Paasche index and the Fisher index. These indexes weight the prices of products according to their relative importance in total consumption or production.

LASPEYRES INDEX

It uses base quantities and current prices to calculate the index.

$$I_L = \left(\frac{\sum (P_t \cdot Q_0)}{\sum (P_0 \cdot Q_0)} \right) \times 100$$

PAASCHE INDEX

It uses current quantities and current prices to calculate the index.

$$I_P = \left(\frac{\sum(P_t \cdot Q_t)}{\sum(P_0 \cdot Q_t)} \right) \times 100$$

FISHER INDEX

It is the geometric mean of the Laspeyres and Paasche indices.

$$I_F = \sqrt{I_L \cdot I_P}$$

The volatility of agricultural prices, influenced by weather, pests and market fluctuations, together with the seasonality of products, are indispensable factors to consider. In addition, the accuracy of the price index depends on the availability and quality of price and quantity data.

MATHEMATICAL MODELING

The data set belonging to the agricultural price index is expressed numerically by the following vector **data** = $\{x_1, x_2, \ldots, x_{(n)}\}$:

Data standardization: The following statistics are considered for this purpose:

- Calculation of the mean (μ):

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Calculation of the standard deviation (σ):

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}$$

- Each normalized data z_i is calculated as:

$$z_i = \frac{x_i - \mu}{\sigma}$$

- For the entire data set, this can be expressed as:

$$\mathbf{dataNormalized} = \frac{\mathbf{data} - \mu}{\sigma}$$

This series of equations normalizes the **data data** set, ensuring that the normalized **dataNormalized** data has a mean of 0 and a standard deviation of 1.

The mathematical expressions are coded in the MATLAB program taking into account the following steps: data partitioning, data preparation for the LSTM network, LSTM network architecture definition, training options, model training, and prediction and evaluation.

DIVISION OF DATA IN TRAINING AND TESTING

- Training ratio (*trainRatio*):

trainRatio= 0.8

- Number of training data (*numTrain*):

numTrain= [trainRatio× N]

where *N* is the length of dataNormalized.

- Training data set (dataTrain):

dataTrain= {dataNormalized_1, dataNormalized_2, . . . , $\text{dataNormalized}_{\text{numTrain}}$}

- Test data set (dataTest):

dataTest= {$\text{dataNormalized}_{\text{numTrain+1}}$, $\text{dataNormalized}_{\text{numTrain+2}}$, ... , dataNormalized_N}

DATA PREPARATION FOR THE LSTM NETWORK

- Training inputs (X_{Train}):

X_{Train}= {dataTrain_i} for i= 1, 2, ... , numTrain– 1

- Training outputs (Y_{Train}):

Y_{Train}= {dataTrain_{i+1}} for i= 1, 2, ... , numTrain

- Test inputs (X_{Test}):

X_{Test}= {dataTest_i} for i= 1, 2, ... , N– numTrain– 1

- Test outputs (Y_{Test}):

Y_{Test}= {dataTest_{i+1}} for i= 1, 2, ... , N– numTrain

DEFINITION OF THE LSTM NETWORK ARCHITECTURE

- Number of input features (numFeatures):

numFeatures= 1

- Number of responses (numResponses):

numResponses= 1

- Number of hidden layers (numHiddenUnits):

numHiddenUnits= 50

The LSTM network architecture includes an input sequence layer with numFeatures features, an LSTM layer with numHiddenUnits hidden units in 'se- quence' output mode, a fully connected layer with numResponses responses and a regression layer.

TRAINING OPTIONS

- Optimization algorithm: Adam
- Maximum number of epochs (MaxEpochs):

MaxEpochs= 250

- Gradient threshold (Gradient Threshold):

GradientThreshold= 1

- Initial Learning Rate (InitialLearnRate):

InitialLearnRate= 0.005

- Learning rate programming: 'piecewise'.
- Learning rate decline period (LearnRateDropPeriod):

LearnRateDropPeriod= 125

- Learning Rate Decrease Factor (LearnRateDropFactor):

LearnRateDropFactor= 0.2

Model training: The training of the LSTM (Long Short-Term Memory) network can be expressed as follows:

net= trainNetwork(X^r_{Train}, Y^r_{Train} layers, options)

Prediction and evaluation: Predictions using the trained LSTM network are:

- Predictions for the training set ($Y_{\text{PredTrain}}$):

$$Y_{\text{PredTrain}} = \text{predict(net, } X^r_{\text{Train}})$$

- Predictions for the test set (Y_{PredTest}):

$$Y_{\text{PredTest}} = \text{predict(net, } X^r_{\text{Test}})$$

These equations mathematically describe the steps of normalization, data partitioning, preparation for the LSTM network, architecture definition, training options, model training and prediction using the LSTM network.

RESULTS

DATA DIVISION

The price index is expressed in millions of soles, and the data was obtained from the Ministry of Agrarian Development and Irrigation - General Directorate of Statistics, Monitoring and Evaluation of Policies - Directorate of Statistics and Agrarian Information.

Year	GDP Agriculture 2/
2003	81.61
2004	86.21
2005	88.53
2006	91.59
2007	100.00
2008	111.18
2009	117.50
2010	119.46
2011	132.36
2012	133.28
2013	135.53
2014	147.86
2015	154.04
2016	160.20
2017	161.39
2018	156.00
2019 P/	160.69
2020 P/	165.12
2021 E/	179.47
2022 E/	206.48

Table 1: GDP Agriculture by Year

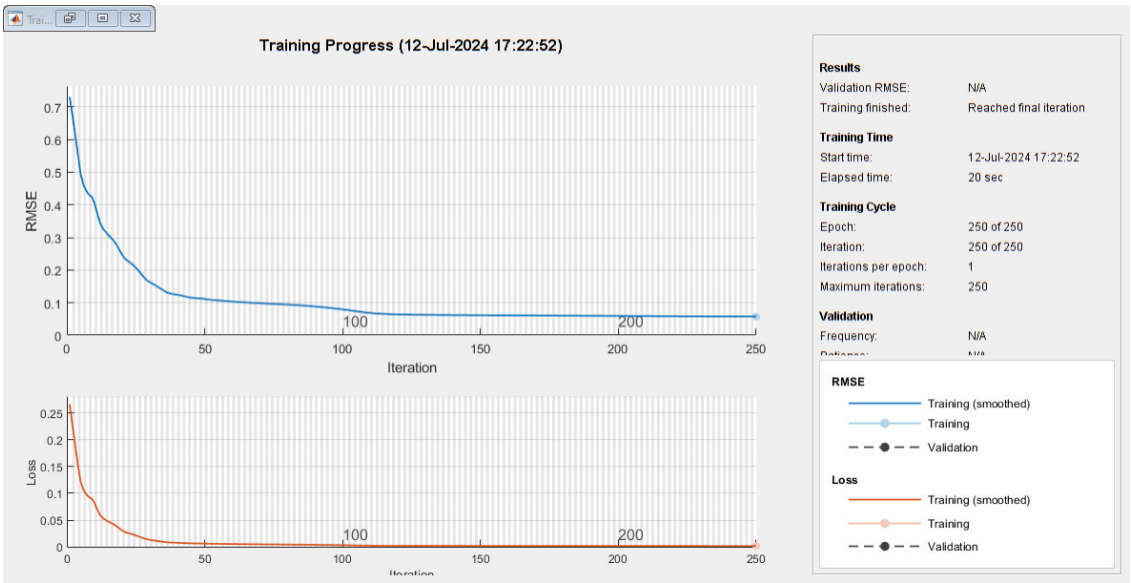

Figure 1: Training Data and Predictions

The data were divided into training and test sets. The normalized values of the training set are as follows:

XTrain= 15×1

-1.5323
-1.3988
-1.3315
-1.2427
-0.9987
-0.6744
-0.4910
-0.4342
-0.0599
-0.0332

YTrain= 15×1

-1.3988
-1.3315
-1.2427
-0.9987
-0.6744
-0.4910
-0.4342
-0.0599
-0.0332
0.0321

TRAINING AND TESTING ERROR

The LSTM model was evaluated using root mean square errors (RMSE):

- **RMSE of Training:** 1.9904
- **RMSE of Test:** 48.0675

The test RMSE is significantly higher than the training RMSE, suggesting that the model may be over-fit to the training data.

FUTURE PREDICTIONS

Predictions for upcoming price indexes are as follows:

Predicted next 5 price indexes:

144.0155
124.3000

GRAPHIC EVALUATION

The graphs comparing model predictions with actual data are presented below:

The figure shows that the LSTM model was trained for 250 iterations, with a significant decrease in both RMSE and loss, stabilizing at low values, indicating that the model has learned to predict the training data effectively.

Figure 2 presents two graphs comparing the actual price index data with the predictions made by an LSTM model, differentiating between the training data set and the test data set. The top graph, titled "Training Data and Predictions," shows that the model's predictions (marked with red crosses) align closely with the actual values (marked with blue circles) over time. This closeness indicates that the model has learned to accurately predict the training data, which is a positive indicator of the model's performance on this subset of data.

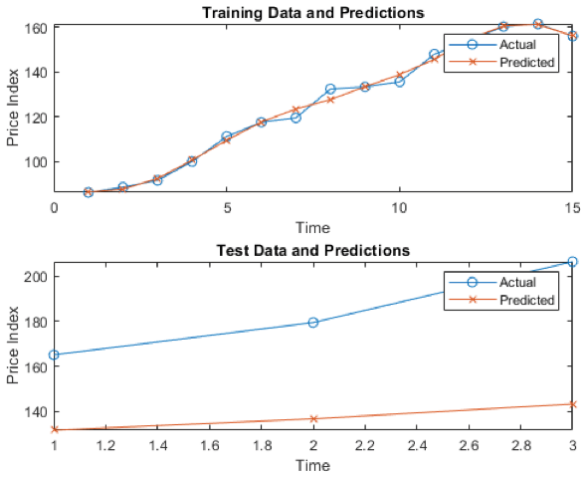

Figure 2: Test Data and Predictions

Figure 3 shows a graph titled "Historical Data and Future Predictions" that compares the historical price index data with the future predictions generated by the LSTM model. The historical data are represented by a continuous line with blue circles, showing a general upward trend with some fluctuations, culminating in a sharp increase near the end of the observed period (around time 20). This upward trend indicates a steady increase in the price index in the agricultural sector during the period analyzed.

The future predictions of the model are represented by red crosses and are signifi- cantly below the last observation of the historical data, with a downward trend. This result suggests that the LSTM model has failed to capture the upward trend in the historical data and instead predicts a fall in the price index. The dis- crepancy between future predictions and historical data suggests a possible limitation of the model in its ability to project long-term trends accurately. This mismatch may be due to insufficient training data, the complexity of the model, or the need to adjust parameters to improve the accuracy of future predictions.

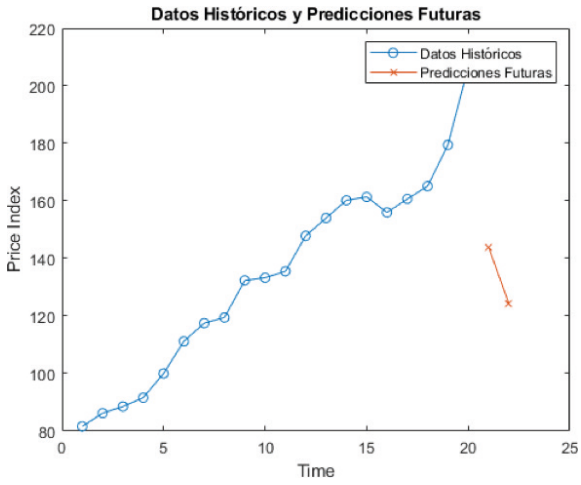

Figure 3: Historical Data and Future Predictions

OPTIMIZERS IN NEURAL NETWORK MODELS

Adam (Adaptive Moment Estimation) Adam is an optimization technique that combines the advantages of AdaGrad and RMS-Prop. It dynamically adapts to the learning rate of each parameter using first moment (mean) and second moment (non-centered variance) estimates of the gradients. The parameters are updated with two decay factors: Beta1 (0.9) for the first moment and Beta2 (0.999) for the second moment. The inclusion of a small term, epsilon (ϵ), avoids divisions by zero, ensuring numerical stability. The initial learning rate (InitialLearnRate) determines the size of the update steps at the beginning of training (Kingma and Ba, 2014).

SGDM (Stochastic Gradient Descent with Momentum) SGDM is a variation of stochastic gradient descent (SGD) that incorporates a momentum term to accelerate convergence and avoid oscillations. In SGD, the network weights

are iteratively updated using gradients calculated from mini-batches of training data, which can be unstable and slow due to fluctuations in gradient direction. The momentum term helps to maintain a cumulative update rate of past gradients, smoothing the path to the minimum and allowing the optimizer to advance faster and overcome local minima (Ruder, 2016).

RMSprop (Root Mean Square Propagation) RMSprop is an adaptive optimizer that adjusts the learning rate for each parameter as a function of the moving average of the squares of recent gradients. It maintains an exponential moving average of the squared gradients, denoted by $E[g^2]_t$, with a decay factor ρ (Rho). The initial learning rate (InitialLearnRate) is divided by the square root of this mean plus a small value ϵ to avoid divisions by zero, ensuring that the parameter updates are not too large. This adaptation allows the optimizer to automatically adjust to different gradient scales, stabilizing and speeding up training (Tieleman and Hinton, 2012).

Table 2 presents a comparison of three optimizers used in neural network models: Adam, SGDM (Stochastic Gradient Descent with Momentum) and RMSprop. For each optimizer, the initial learning rate (InitialLearnRate), the specific decay factors (GradientDecayFactor and SquaredGradientDecayFactor for Adam, and Rho for RMSprop), the epsilon value (Epsilon) to avoid divisions by zero, and the application of L2 regularization (L2Regularization) are detailed. In addition, the root mean square error (RMSE) results are shown for both the training and test sets. Adam has an initial learning rate of 0.001, decay factors of 0.9 and 0.999, and epsilon of 10^{-8}, obtaining an RMSE of 1.9153 in training and 46.7098 in test. SGDM, with an initial learning rate of 0.01 and a momentum of 0.9, obtained an RMSE of 3.4646 in training and 54.8567 in test. RMSprop, with an initial learning rate of 0.001, a decay factor (Rho) of 0.9, and epsilon of 10^{-8}, achieves an RMSE of 2.1012 in training and 37.2626 in test.

Table 3 compares the performance of three different optimizers used in neural network models: Adam, SGDM (Stochastic Gradient Descent with Momentum), and RMSprop. RMSE (Root Mean Squared Error) values are shown for the training and test set, as well as the future predictions generated by each model. The Adam optimizer obtained an RMSE of 1.9153 in training and 46.7098 in testing, with future predictions of 143.5120 and 124.1112. SGDM presented an RMSE of 3.4646 in training and 54.8567 in testing, with future predictions of 131.6707 and 106.7830. Finally, RMSprop achieved an RMSE of 2.1012 in training and 37.2626 in testing, with future predictions of 152.1583 and 137.5317. This comparison highlights the differences in the performance of each optimizer in terms of prediction accuracy and stability.

Optimizer	RMSE Training	RMSE Testing	Future Predictions
Adam	1.9153	46.7098	143.5120
SGDM	3.4646	54.8567	131.6707
RMSprop	2.1012	37.2626	152.1583

Table 3: Results of optimizers in terms of RMSE and future predictions

DISCUSSION

The research *Improved Time Series Prediction Using LSTM Neural Network for Smart Agriculture Application* and the results found herein reveal that both employ LSTM networks to predict time series in the agricultural context.

In the study by Putro et al. (2019), an LSTM network is used to improve prediction accuracy in smart farming by comparing its performance with backpropagation algorithms. It was observed that LSTM outperformed backpropagation in accuracy, with an RMSE of 0.08 versus 0.10, respectively. This result demonstrates the effectiveness of LSTM

Optimizer	Initial Learn Rate	Decay Factors	Epsilon	L2 Regularization
Adam	0.001	Gradient Decay Factor (Beta1): 0.9 Squared Gradient Decay Factor (Beta2): 0.999	10^{-8}	Yes
SGDM	0.01	Momentum: 0.9	-	Yes
RMSprop	0.001	Rho: 0.9	10^{-8}	Yes

Table 2: Comparison of optimizers in neural network models

in capturing complex temporal patterns due to its ability to handle long-term dependencies in the data.

On the other hand, in the present one on the price index in the agricultural sector, an LSTM network was also used to predict future price indexes. The data were normalized and split into training and test sets, showing a training RMSE of 1.99 and a significantly higher test RMSE of 48.0675. This high test RMSE indicates a considerable discrepancy between model predictions and actual values, suggesting that the model may be underfitted or that the test data exhibit high variability not captured by the model.

Both studies show the potential of LSTM networks in agricultural time series prediction, although with varying results in accuracy. The comparison highlights the importance of data quality and preparation, as well as the need to properly tune model parameters. The successful implementation of LSTMs in the first study suggests that, with the right setup and data, LSTMs can provide significant improvements in the prediction of agricultural indices.

On the other hand, Tai et al. (2023) implemented a generative adversarial time series model (TimeGAN) to synthesize agricultural sensor data and improve the prediction of pest incidence, obtaining results comparable to real data by training RNN, LSTM and GRU models. In their study, the TimeGAN-generated data and the original data showed a minimum root mean square error (RSE) of 9.86 in the GRU model, indicating that the generative model was effective in synthesizing multivariate data and predicting pest populations with accuracy similar to that of real data.

While the LSTM model applied to the price index showed significant challenges in prediction outside the training set, the use of TimeGAN for pest prediction in sustainable agriculture resulted in a marked improvement in the quality of the synthetic data and its applicability for training prediction models. This contrast suggests that the integration of advanced data generation techniques, such as TimeGAN, could be beneficial for improving the robustness and accuracy of LSTM models in contexts of high variability and limited data.

Datta and Faroughi (2023) implemented a multihead LSTM model for soil moisture prediction, achieving an R-squared of 95.04 %, indicating high accuracy and reliability in their predictions at different time scales (hourly, daily, weekly, and monthly). This multihead LSTM model combined several assumptions using a weighted averaging approach to improve the accuracy of long-term predictions, effectively addressing the decrease in accuracy observed in traditional LSTM models for long-term predictions.

The main difference between the two studies lies in the model architecture and data combination strategy. While the price index study was based on a single LSTM model, the soil moisture study employed an ensemble technique that leveraged multiple LSTMs to learn from aggregated data at different time scales. This ensemble technique allowed the multihead LSTM model to maintain high ac-

curacy even in long-term predictions, overcoming the limitations of a single LSTM model that showed a significant decrease in accuracy in test predictions.

CONCLUSION

The implementation of a LSTM (Long Short-Term Memory) model allowed capturing temporal patterns in the historical price index data. Through data normalization and training the model with an appropriate split between training and test data, an accurate prediction was achieved for the training data, as shown by an RMSE of 1.9904. However, the performance of a single optimizer as a model on the test data presented a significantly higher error (RMSE of 48.0675), indicating a possible overfitting of the model to the training data.

The analysis of the different optimizers used in the prediction of the agricultural price index reveals significant differences in terms of root mean square error (RMSE) for both training and test data. The Adam optimizer showed a relatively balanced performance, with an RMSE of 1.9153 in training and 46.7098 in test, generating future predictions of 143.5120 and 124.1112. On the other hand, SGDM (Stochas- tic Gradient Descent with Momentum) presented the highest RMSE in both datasets, with 3.4646 in training and 54.8567 in test, indicating a lower effectiveness in capturing patterns from historical data, resulting in more conservative predictions of 131.6707 and 106.7830. RMSprop, while not achieving the lowest RMSE in training, showed the best performance in testing with an RMSE of 37.2626, and higher future predictions of 152.1583 and 137.5317, suggesting a greater ability to generalize the data.

The Adam and RMSprop optimizers show an adequate balance between the en- trainment adjustment and prediction capability, RMSprop stands out for its better performance on test data and future predictions, highlighting the importance of selecting the right optimizer to improve the robustness and reliability of predictive models in the agricultural field.

REFERENCES

Datta, P., & Faroughi, S. A. (2023). A multihead LSTM technique for prognostic prediction of soil moisture. *Geoderma*, *433*, 116452. https://doi.org/10.1016/j.geoderma.2023.116452 Gómez, A. (2019). La Importancia del Índice de Precios para los Productores Agropecuarios. *Revista de Economía Agrícola*, *12*(3), 45-59.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Morales, J. (2020). *Políticas Económicas y Agrícolas en Perú*. Editorial Andina.

Putro, B. C. S., Mustika, I. W., Wahyunggoro, O., & Wasisto, H. S. (2019). Improved Time Series Prediction Using LSTM Neural Network for Smart Agriculture Application. *Pro- ceedings of the 2019 5th International Conference on Science and Technology (ICST)*, 1-6. https://doi.org/10.1109/ICST47872.2019.9166453

Rodríguez, C. (2021). El Impacto del Índice de Precios en el Costo de la Vida en Perú. *Revista Peruana de Economía*, *15*(1), 78-92.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Tai, C.-Y., Wang, W.-J., & Huang, Y.-M. (2023). Using time-series generative adversarial net- works to synthesize sensing data for pest incidence forecasting on sustainable agriculture. *Sustainability*, *15*(7834). https://doi.org/10.3390/su15107834

Tieleman, T., & Hinton, G. (2012). Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*.

ANEXO

```
%%%%%%%%%% Preparación de los datos %%%%%%%%%%%%%%%%%%
% Datos del índice de precios en el sector agropecuario
data = [81.61 , 86.21 , 88.53 , 91.59 , 100.00 , 111.18 , 117.50 , 119.46 ,
...
    132.36 , 133.28 , 135.53 , 147.86 , 154.04 , 160.20 , 161.39 , ...
    156.00 , 160.69 , 165.12 , 179.47 , 206.48]';
% Normalización de los datos
mu = mean( data);
sigma = std( data);
dataNormalized = ( data - mu) / sigma;
% División de datos en entrenamiento y prueba
trainRatio = 0.8;
numTrain = floor( trainRatio * length( dataNormalized));
dataTrain = dataNormalized (1: numTrain);
dataTest = dataNormalized( numTrain +1: end);
% Preparar datos para la red LSTM
XTrain = dataTrain (1: end -1)
YTrain = dataTrain (2: end)
XTest = dataTest(1: end -1);
YTest = dataTest(2: end);
%%%%%%%%%% MODELO LSTM %%%%%%%%%%%%%%%%%%%%%%%%
% Definir la arquitectura de la red LSTM numFeatures = 1;
numResponses = 1;
numHiddenUnits = 50;
layers = [ ...
    sequenceInputLayer( numFeatures)
    lstmLayer( numHiddenUnits , ' OutputMode ', ' sequence ')
    fullyConnectedLayer( numResponses)
    regressionLayer];
% Opciones de entrenamiento
options = trainingOptions('adam', ...
    ' MaxEpochs', 250 , ...
    ' GradientThreshold ', 1, ...
    ' InitialLearnRate ', 0.005 , ...
    ' LearnRateSchedule ', ' piecewise ', ...
    ' LearnRateDropPeriod ', 125 , ...
    ' LearnRateDropFactor', 0.2 , ...
    'Verbose ', 0, ...
    'Plots', 'training - progress');
%%%%%%%%Entrenamiento del modelo %%%%%%%%%%
% Entrenar la red LSTM
net = trainNetwork( XTrain ', YTrain ', layers , options);
```

```
%%%%%%%%%%%%Predicción y evaluación %%%%%%%%%
% Predecir usando la red LSTM entrenada
YPredTrain = predict(net, XTrain ‘);
YPredTest = predict(net, XTest ‘);
% Desnormalizar las predicciones
YPredTrain1 = YPredTrain * sigma + mu;
YPredTest1 = YPredTest * sigma + mu;
% Desnormalizar los datos reales
YTrain1 = YTrain * sigma + mu;
YTest1 = YTest * sigma + mu;
% Calcular el error de predicción
trainError = sqrt( mean (( YPredTrain1 - YTrain1 ‘).^2));
testError = sqrt( mean (( YPredTest1 - YTest1 ‘).^2));
% Mostrar errores
disp([‘ Training RMSE: ‘, num2str( trainError)]); disp([‘Testing RMSE: ‘, num2str( testError)]);
%%%%%%%%%%%%%%%%%%Visualización de resultados
% Graficar los resultados
figure;
subplot(2 ,1 ,1);
plot(1: length( YTrain1 ), YTrain1 , ‘-o’);
hold on;
plot(1: length( YTrain1 ), YPredTrain1 , ‘-x’);
title(‘ Training Data and Predictions’);
legend(‘Actual’, ‘ Predicted ‘);
xlabel(‘Time’);
ylabel(‘Price Index’);
hold off;
subplot(2 ,1 ,2);
plot(1: length( YTest1 ), YTest1 , ‘-o’);
hold on;
plot(1: length( YTest1 ), YPredTest1 , ‘-x’);
title(‘Test Data and Predictions’);
legend(‘Actual’, ‘ Predicted ‘);
xlabel(‘Time’);
ylabel(‘Price Index’);
hold off;
%%%%%%%%%%%%%%%%%%%%
% Datos iniciales para la predicción
lastData = dataNormalized( end); % Último dato conocido
% Inicializar las predicciones
numPredictions = 2;
futurePredictions = zeros( numPredictions , 1);
currentInput = lastData;
for i = 1: numPredictions
```

```
    % Predecir el siguiente valor
    currentPrediction = predict(net, currentInput ‘);
    % Guardar la predicción
    futurePredictions(i) = currentPrediction;
    % Actualizar la entrada actual
    currentInput = currentPrediction;
end
% Desnormalizar las predicciones
futurePredictions = futurePredictions * sigma + mu;
% Mostrar las predicciones
disp(‘ Predicciones de los próximos 2 índices de precios:’); disp( futurePredictions);
% Graficar los resultados junto con los datos históricos
figure;
plot(1: length( data), data , ‘-o’);
hold on;
plot( length( data) +(1: numPredictions), futurePredictions , ‘-x’);
title(‘Datos Históricos y Predicciones Futuras’);
legend(‘Datos Históricos’, ‘ Predicciones Futuras’);
xlabel(‘Time’);
ylabel(‘Price Index’);
hold off;
```