applied sciences

MDPI

Article

Sales Forecasting with LSTM, Custom Loss Function, and Hyperparameter Optimization: A Case Study

Hyasseliny A. Hurtado-Mora, Alejandro H. García-Ruiz *, Roberto Pichardo-Ramírez, Luis J. González-del-Ángel and Luis A. Herrera-Barajas

Faculty of Engineering Tampico, Autonomous University of Tamaulipas, Centro Universitario Sur, Tampico 89336, Tamaulipas, Mexico; hhurtado@docentes.uat.edu.mx (H.A.H.-M.); rpichardo@docentes.uat.edu.mx (R.P.-R.); ljonas@docentes.uat.edu.mx (L.J.G.-d.-Á.); lubarajas@uat.edu.mx (L.A.H.-B.)

* Correspondence: ahgarcia@docentes.uat.edu.mx

Abstract: Forecasting sales trends is a valuable activity for companies of all types and sizes, as it enables more efficient decision making to avoid unnecessary expenses from excess inventory or, conversely, losses due to insufficient inventory to meet demand. In this paper, we designed a personalized cost function to reduce economic losses caused by the excessive acquisition of products or derived from their scarcity when needed. Moreover, we designed an LSTM network integrated with Glorot and Orthogonal initializers and dropout to forecast sales trends in a lumber mill in Tamaulipas, Mexico. To generalize and appropriately forecast the sales of the lumber mill products, we optimized the LSTM network's hyperparameters through a genetic algorithm, which was essential to explore the solution space. We evaluated our proposal in instances obtained from the historical sales of the five main products sold by the lumber mill. According to the results, we concluded that for our case study the proposed function cost and the hyperparameters optimization allowed the LSTM to forecast the direction and trend of the lumber mill's product sales despite the variability of the products.

Keywords: forecasting; custom loss function; LSTM; hyperparameter optimization; genetic algorithm

check for updates

Citation: Hurtado-Mora, H.A.; García-Ruiz, A.H.; Pichardo-Ramírez, R.; González-del-Ángel, L.J.; Herrera-Barajas, L.A. Sales Forecasting with LSTM, Custom Loss Function, and Hyperparameter Optimization: A Case Study. *Appl. Sci.* **2024**, *14*, 9957. https://doi.org/10.3390/app14219957

Academic Editor: Grigorios Beligiannis

Received: 23 September 2024
Revised: 25 October 2024
Accepted: 28 October 2024
Published: 31 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Currently, the field of forecasting has grown so much that companies have started to have a great interest in developing sales forecasting models that help them build business strategies in areas such as production, transportation, and the supply chain, among others [1,2]. These forecasting models facilitate identifying market opportunities and anticipating possible challenges [3], allowing business tactics to be adapted effectively [4]. Additionally, forecasting is essential for the efficient allocation of resources because it helps prioritize investments and optimize the use of assets [5]. Furthermore, sales forecasting provides valuable information for designing advertising campaigns and promotions that reach the right audience at the right time [6]. It also influences logistics by determining product flows and the storage capacity needed to keep the supply chain. Moreover, efficient inventory management and supply chain optimization are critical to any company's increasing operational and financial success rate; they are especially relevant in sectors such as goods, services, industry, manufacturing, logistics, transportation, the automotive sector, and e-commerce, where fluctuations in demand and seasonality can represent significant challenges [7].

In this sense, trends and forecasts are essential to addressing the inventory management problem. Gupta and Agarwal [8] emphasized the importance of sales forecasting accuracy for businesses, enabling proper resource allocation, efficient inventory management, and streamlined operations. Thus, analyzing historical sales trends will allow the identification of demand patterns, such as seasonality or sales cycles. The above will help

predict when and in what quantity products are needed, allowing industries to anticipate demand peaks and adjust their inventory levels accordingly, avoiding product shortages or excessive purchases at unnecessary times. Diverse papers can be found on the state-of-the-art related to forecasting, mainly using so-called Recurrent Neural Networks (RNNs). These networks have emerged as essential tools in machine learning, especially when processing sequential data. Their ability to maintain and use information about past events makes them particularly useful for diverse applications that depend on temporal patterns [9]. An example is [10], where Nagesh et al. addressed the prediction of hydrological time series, using a Recurrent Neural Network (RNN) to forecast monthly river flows. The authors conducted a comparative study of a multilayer perceptron network and an RNN. The study showed that the RNN outperformed the perceptron network by achieving greater accuracy in less training time. The paper concluded that the superior accuracy of the RNN was due to its ability to capture and utilize the temporal dependencies present in hydrological data.

In [11], Mandal and Prabaharan used an RNN to predict ocean waves. The RNN could predict waves at intervals of 3, 6, and 12 h. Thus, the results proved to be promising forecasts, showing correlation coefficients of 0.95 for 3 h, 0.90 for 6 h, and 0.87 for 12 h. Based on the forecasts obtained, offshore operations could be carried out more safely and efficiently, benefiting activities such as drilling, merchant ship navigation, and coastal construction. Similarly, in [12], Mandal et al. proposed an approach for predicting hourly prices in the electricity field based on similar days; the authors developed and trained an RNN model using backpropagation for their proposal. They also designed a method to select days with similar characteristics, to improve the price forecast. MAPE and FMSE were used to evaluate the performance of the RNN model. According to the results obtained, the authors identified that "demand" is the most crucial variable in the price of electricity and concluded that the RNN-based algorithm effectively solved the volatility problem in the price curves of the US-regulated electricity market.

However, despite their advantages, RNNs have some limitations that make them ineffective for specific problems. In this sense, LSTM networks have recently emerged as an advanced variant of RNNs for forecasting and time series analysis due to their ability to handle long-term dependencies and gradient stability. Thus, LSTMs are frequently found in the state-of-the-art instead of RNNs, due to their performance in complex time series forecasting, and they currently remain one of the most robust and versatile options. For example, Seng and Julio in [13] proposed an LSTM network for stock price prediction in the financial market; the authors used a three-layer network to predict the closing prices of stocks in the LQ45 financial sectors. The experimental results showed that the predictions were reasonable. Therefore, the authors recommended using LSTM networks in this context, based on the low MAPE and predictions obtained. In [14], Shen et al. developed an effective foreign trade forecasting method, using an LSTM network to analyze the trade data of 10 countries. The proposed method was developed based on the economic theory of two-way causal relationships in trade data. The results showed that the technique could adequately model the temporal information on uncertainty trends in foreign trade data. In almost all cases, the MAPE and RMSE errors calculated were lower than those of the other models analyzed. For their part, Francesco and Fabio in [15] developed three RNN models to predict short-term actual evapotranspiration. Additionally, they used an ensemble technique to improve the accuracy of the results. The models were trained and tested at two sites with different climates. The results indicated that LSTM models are more accurate with data from subtropical climates. In contrast, Non-linear AutoRegression with eXogenous inputs (NARX) models perform better when using data from semi-arid climates.

Furthermore, some authors have adapted LSTM networks to be bidirectional and have combined them with other techniques, as in the case of Wenjie et al. [16], where they proposed a method to predict the closing price of stocks for the next day, combining LSTM networks with Convolutional Neural Networks (CNNs) and an Attention Mechanism (AM). The authors compared the performance of the proposed model with seven other different methods. The authors calculated the following metrics for the comparison: Mean

Absolute Error (MAE), Root Mean Square Error (RMSE), and coefficient of determination (R^2). The experimentation showed that the CNN-BiLSTM model obtained the best results in predicting the closing price of stocks with an MAE of 21.952, an RMSE of 31.694, and an R^2 of 0.9804. Similarly, Kun et al. [17] developed a daily temperature forecasting model for the country of China, in which an approach combining seasonal and trend decomposition using the Seasonal–Trend decomposition employing the LOESS (STL) method and the Jumping Spectrum and Trend (JUST) algorithm, together with a bidirectional Long Short-Term Memory (Bi-LSTM) network, was proposed. The experimental results showed that the proposed model significantly outperformed other models, such as Bi-LSTM, END-LSTM, and STL-LSTM, in terms of accuracy and being the lowest in prediction errors, with an RMSE error of 0.2187 and an MAE of 0.1737.

In this paper, we propose the design of an LSTM network to forecast the sales of a lumber mill, considering two scenarios: (1) costs of product construction when demand exceeded the available stock in the inventory, and (2) operating costs generated by the products stored when demand was less than the inventory. This paper aimed to accurately forecast the sales demand trend for D'Gallery lumber mill products in Tamaulipas, Mexico. The proposed LSTM network focused on reducing costs associated with constructing products during high demand and minimizing storage costs by maintaining optimal inventory levels. In this way, we sought to provide D'Gallery with a tool to improve its inventory management, reduce operating costs, and increase its profitability.

Thus, the remainder of the paper is as follows: Section 2 presents the materials and methods; Sections 3 and 4 describe the experimentation results and the discussion, respectively; and, finally, Section 5 presents the conclusions and future work.

2. Materials and Methods

For this study, we built the dataset by collaborating with a lumber mill company in Tamaulipas, Mexico. We analyzed the company's history for five products: dining room, interior door mod1, interior door mod2, entrance door mod1, and entrance door mod2. We selected these products because these were the most significant ones, in terms of sales, for the lumber mill. The data provided by the company covered five years, from 2019 to 2023. In addition, the company handles sales reports for products every week. Thus, the data instance had 261 records for each product considered in this study.

The following variables constituted the instance:

- Year: Year extracted from the Date column.
- Sine_Week: Value of the sine of the week of the year, representing the weekly periodicity.
- Cosine_Week: Cosine value of the week of the year, complementing the cyclical representation.
- Event: Indicated whether there were any events during the week.
- Sales: Number of units sold in the week.

As the weeks of the years were cyclic, we characterized them as the sine and cosine functions. Figure 1 shows a characterization for some weeks for product 1.

Furthermore, since we observed that sales of the different products considered had different behaviors among themselves, we decided to have one instance for each product. Table 1 presents a partial part of the data instance obtained for product 1: dining room. We used this structure for each of the instances created.

Table 1. Data instance of product 1: dining room.

Record	Year	Sine_Week	Cosine_Week	Event	Sales
1	2019	0.120537	0.992709	0	32
2	2019	0.239316	0.970942	0	12

Table 1. *Cont.*

Record	Year	Sine_Week	Cosine_Week	Event	Sales
3	2019	0.354605	0.935016	0	32
4	2019	0.464723	0.885456	0	37
5	2019	0.568065	0.822984	1	74
6	2019	0.663123	0.748511	1	65
7	2019	0.748511	0.663123	1	68
8	2019	0.822984	0.568065	0	24
9	2019	0.885456	0.464723	0	46
10	2019	0.935016	0.354605	0	32
11	2019	0.970942	0.239316	0	31
12	2019	0.992709	0.120537	0	38
13	2019	1	0	0	37
14	2019	0.992709	−0.120537	0	9
15	2019	0.970942	−0.239316	0	39

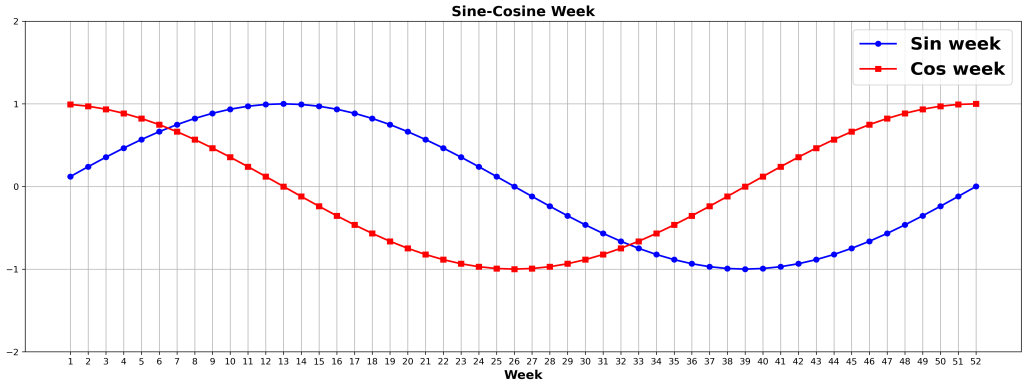


Figure 1. Cyclical representation of the weeks of the year.

2.1. Preprocessing

Once we had defined the instance variables, we established the normalization method for the instance. This process was essential to ensuring that all the variables had the same scale and, thus, to facilitating their manipulation by the LSTM network. The normalization method considered was standardization, ensuring that all the variables had a mean of 0 and a standard deviation of 1: we chose this method to evaluate their efficiency in simplifying the training and improving the convergence of the LSTM network. Table 2 shows a fragment of the data instance for product 1, dining room, as an example of the result of the normalization process.

Table 2. Standardization of the instance of product 1: dining room.

Record	Year	Sine_Week	Cosine_Week	Event	Sales
1	−1.412863	0.170128	1.395949	−0.318223	0.437206
2	−1.412863	0.33842	1.365223	−0.318223	−0.632249
3	−1.412863	0.501768	1.31451	−0.318223	0.437206
4	−1.412863	0.657789	1.24455	−0.318223	0.70457
5	−1.412863	0.804209	1.156364	3.142451	2.683063

Moreover, we divided each instance into training, validation, and testing sets. The percentages for producing each set in the experimentation were 60% training, 20% validation, and 20% testing.

Subsequently, we treated each instance set to generate datasets—an essential step, as this paper considered a single-step multivariate approach (Multivariate-Unistep). Thus, we used these datasets to train, validate, and test the proposed LSTM network.

Based on preliminary tests performed to create the datasets, we chose 26 weeks for the input datasets and 1 week for the output datasets. In this sense, we created the datasets for each instance and variable, as shown in Figure 2, for the variable sales.

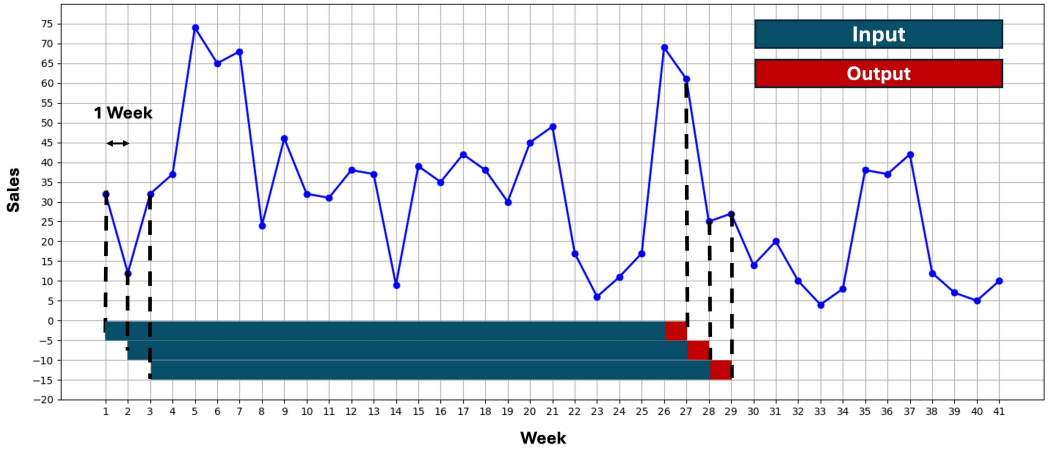


Figure 2. Datasets generation.

2.2. LSTM Network Structure

Figure 3 shows the structure of the proposed LSTM network and considers the following characteristics:

- Input. Datasets generated for the product are to be forecasted by the LSTM network.
- LSTM layers. A set of three LSTM layers, each of them with a dropout rate, to handle data sequences, capture long-term dependencies, and avoid overfitting or underfitting.
- Dense layer. Layer with a linear activation function used to generate the forecasts by the LSTM network.
- Output. Network's forecast for sales of a product for the next period.

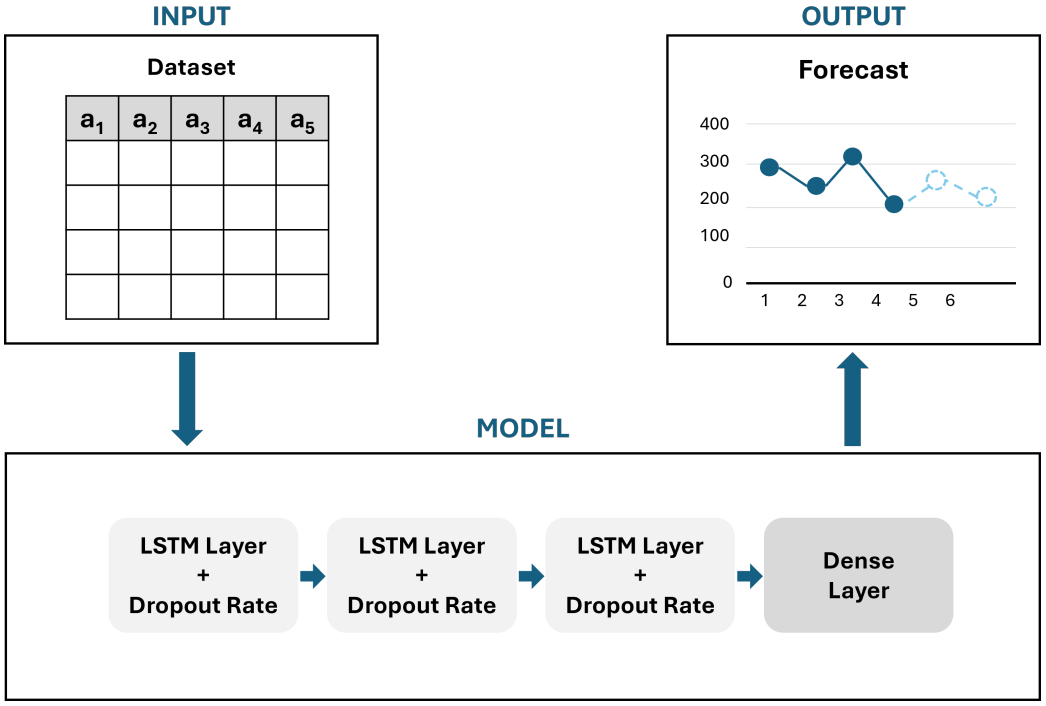


Figure 3. LSTM network structure.

The LSTM network structure included nine hyperparameters that modified its behavior, which we explored through an evolutionary algorithm to find the best configuration of hyperparameters. Based on the preliminary experiment results, the value ranges shown in Table 3 were established for the hyperparameters.

Table 3. Value range for LSTM network hyperparameters.

Hyperparameter	Value Range
LSTM units	100–400
Epochs	10–50
Batch size	1–Total blocks
Learning Rate	0.0001–0.001
Dropout	0.2–0.5

As for the rest of the LSTM network configurations, we considered the following:

- We used the Adam optimizer because, in the literature, this optimizer has been demonstrated to be computationally efficient, uses little memory, is invariant to the diagonal rescaling of gradients, and is suitable for significant problems in terms of data or parameters [18].
- For the loss function for the LSTM network, we used the Root Mean Square Error (RMSE). The RMSE has been used as a standard to measure the performance of diverse models in multiple studies. This metric expresses the error in the same magnitude or unit as the variable measured for a model, facilitating a more straightforward interpretation of the errors [19].
- We initialized with zeros on the biases of the neurons in all layers of the LSTM network [20,21], and we proved the generation of the initial values of the neuron weights with the following kernels: uniform random, Glorot, and Orthogonal [22–25].

2.3. Custom Loss Function Cost

We designed a custom loss cost function focused on satisfying the demand of the lumber mill, considering the storage cost and the cost of processing the products. In this way, the function cost consisted of two relevant variables that allowed the calculation of the total cost associated with the model's forecasts.

- Cost Per Excess Inventory (CPEI), which represented the storage cost per unit of excess inventory.
- Cost Per Inventory Shortage (CPIS), which was associated with the production cost per missing product unit.

For the selection of the values for both variables assumed in the custom loss function cost proposed, we considered that the loss of having shortage inventory should be higher than the loss of having excess inventory. We established the above because an excess of inventory represents a cost for the company for the storage of the products, operating, and logistics, while having an insufficient inventory when customers want to purchase products requires that the company produces more products in a short time, which, after multiple occasions, can signify an additional cost to satisfy the demand; however, it is necessary because missing products represents an absence of sales, which could cause an inadequate reputation for the company that could result in a decrease in its sales in the future.

In preliminary tests, we set the value of the cost function variables to 2.0 for the cost per unit due to excess inventory and 6.0 for the cost per unit due to a shortage of inventory. This allowed us to clearly differentiate between the impact of inventory shortages and excess inventory when calculating the company's losses.

Algorithm 1 shows the designed cost function. The function iterated over the differences between the predictions made by the model and the actual values for a data instance. After that, the function calculated the total cost considering those differences,

the inventory stocks, and the defined multipliers. Finally, the function returned the total accumulated cost.

Additionally, the function tracked and updated the inventory as forecasts were processed—an inventory update occurred when there was an excess or shortage of products to satisfy the client's demand. Keeping the current inventory level up to date was essential for the custom function to adequately perform the total cost calculation.

Through the custom cost function, it is possible to have a more suitable mechanism to evaluate the performance of an LSTM network based on the costs obtained from its forecasts. The function is particularly useful in practical applications where forecasts can have direct economic impacts, such as in the case of the lumber mill considered in this paper.

Algorithm 1 Custom loss function cost

Require: $y_{forecast}$: Sales forecasts, y_{actual}: Actual sales
Ensure: $total_loss$: Accumulated loss
CPEI $\leftarrow$ 2.0 ▷ Cost Per Excess Inventory
CPIS $\leftarrow$ 6.0 ▷ Cost Per Inventory Shortage
inventory $\leftarrow$ 0 ▷ Initial inventory
total_loss $\leftarrow$ 0 ▷ Initial loss
diff $\leftarrow y_{forecast} - y_{actual}$ ▷ Differences among forecasts and actual values
for each value **in** diff **do**
    value $\leftarrow$ inventory + round(value)
    if value $>$ 0 **then**
        loss $\leftarrow$ value $\times$ CPEI
        total_loss $\leftarrow$ total_loss + loss
        inventory $\leftarrow$ value
    else if value $<$ 0 **then**
        value $\leftarrow$ value $\times -1$
        inventory $\leftarrow$ 0
        loss $\leftarrow$ value $\times$ CPIS
        total_loss $\leftarrow$ total_loss + loss
    end if
end for
return total_loss

3. Results

For the development of this paper and the experimentation realized, we used a MacBook Pro M1 with 512GB SSD, 16GB RAM, and 10 CPUs. For the programming of the proposal, we used Python 3.11 and the following modules: Tensorflow 2.15.0, Keras 2.15.0, Numpy 2.1.1, Pandas 2.2.0, and Matplotlib 3.8.2.

We performed the hyperparameter optimization of the LSTM network using a genetic algorithm based on [26]. Genetic algorithms were proposed by Holland (1998) [27] and are one of the most popular evolutionary algorithms. Diverse authors have used the genetic algorithms [28–30]. These algorithms have several advantages, one of the most important being that genetic algorithms start with randomly selected initial values, making their implementation easier as they do not require much effort to choose good initial values [31]. According to Haupt and Haupt (2004) [32], a genetic algorithm is an optimization technique based on genetics and natural selection principles. A genetic algorithm allows a population of many individuals to evolve under specific selection rules toward a state that maximizes their fitness value.

In our genetic algorithm implementation, the population consisted of a matrix of n solutions, each representing a configuration of the nine hyperparameters for the LSTM network. The first five chromosome values corresponded to discrete values and the rest to continuous values. Table 4 shows the structure of a chromosome.

Table 4. Structure of a chromosome.

Hyperparameter	Description	Type
0	LSTM 1 Units	Discrete
1	LSTM 2 Units	Discrete
2	LSTM 3 Units	Discrete
3	Epochs	Discrete
4	Batch size	Discrete
5	Learning Rate	Continuous
6	Dropout 1	Continuous
7	Dropout 2	Continuous
8	Dropout 3	Continuous

Since a genetic algorithm measures population fitness through the objective function [33], to measure the performance of a given hyperparameter set, we decided to use our custom loss function cost in the results obtained from the trained LSTM network.

For the implementation of the genetic algorithm, we considered the following operators found in [26,34]:

- Selection: Binary Tournament.
- Crossover:
  - Uniform. In discrete parameters;
  - SBX. In continuous parameters.
- Mutation:
  - Uniform. In discrete parameters;
  - Polynomial. In continuous parameters.

Algorithm 2 presents the genetic algorithm designed and implemented in this paper.

Algorithm 2 Pseudocode for the Genetic Algorithm

Require: $Population_size$, $Problem_size$, $P_{crossover}$, $P_{mutation}$
Ensure: S_{best}
$Population \leftarrow$ InitializePopulation($Population_size$, $Problem_size$)
EvaluatePopulation($Population$)
$S_{best} \leftarrow$ GetBestSolution($Population$)
while not StopCondition() **do**
    $Parents \leftarrow$ SelectParents($Population$, $Population_size$)
    $Children \leftarrow \emptyset$
    for all $Parent_1, Parent_2 \in Parents$ **do**
        $Child_1, Child_2 \leftarrow$ Crossover($Parent_1$, $Parent_2$, $P_{crossover}$)
        $Children \leftarrow$ Mutate($Child_1$, $P_{mutation}$)
        $Children \leftarrow$ Mutate($Child_2$, $P_{mutation}$)
    end for
    EvaluatePopulation($Children$)
    $S_{best} \leftarrow$ GetBestSolution($Children$)
    $Population \leftarrow$ Replace($Population$, $Children$)
end while
return S_{best}

In order to perform an adequate exploration of the solution space with the genetic algorithm, we made a sensitivity analysis of the algorithm's parameters to determine the configuration that provided the best results for optimizing the LSTM hyperparameters with the algorithm. Table 5 shows the range of values that we considered for the sensitivity analysis.

Table 5. Range of parameters for the sensitivity of the genetic algorithm.

Parameter	Value
Population size	[1–50]
Number of generations	[10–1000]
Crossover probability	[0.1–1.0]
Mutation probability	[0.1–1.0]

Table 6 shows the parameters chosen to execute the genetic algorithm.

Table 6. Selected genetic algorithm parameters.

Parameter	Value
Population size	10
Number of generations	100
Crossover probability	0.9
Mutation probability	0.2

3.1. Weight Initializers

We tested three configurations with the initialization methods selected in this paper, to establish the weight initialization method that we would ultimately use to train the LSTM network. Table 7 shows the configurations tested. We used the proposed configurations to perform hyperparameter optimization with the data instance of each lumber mill product considered in this paper.

Table 7. Input and recurrent kernel weight initializers.

Configuration	Input Kernel	Recurrent Kernel
A	Uniform	Uniform
B	Uniform	Orthogonal
C	Glorot	Orthogonal

We performed the tests of the configurations with the data instances in their original form and their standardized version, to determine whether standardization is suitable for improving product sales forecasts.

Below, we present the results of each configuration for the dining room product. However, we replicated this process for the five lumber mill products considered in this paper.

3.1.1. Configuration A (Uniform, Uniform)

The models obtained with configuration A allowed us to observe the forecasts made by the LSTM network. Figure 4 shows better results for when the data instance was standardized. The above is especially noticeable in the peak of higher sales in the actual data, where the standardized model allowed a better sales forecast than those made by the model with the instance in its original form, demonstrating, in the first instance, that standardization is a promising feature for allowing the model to generalize the information in a better way.

With our proposed cost function, we obtained a cost of 1430 when testing configuration A with the original instance and 1806 when using the standardized version of the instance. In this way, configuration A presented better results (less cost) when training the network with data instances in their original form.

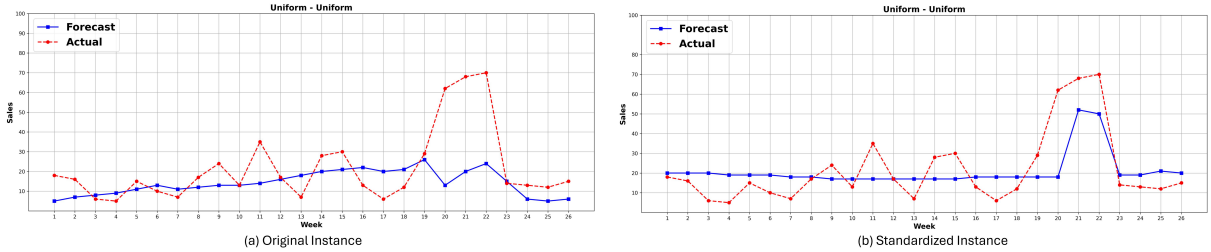


Figure 4. Results of weight initializer configuration A.

3.1.2. Configuration B (Uniform, Orthogonal)

A slight advantage could be seen in this second configuration when using the non-standardized data instance. Figure 5 shows that the model tried to adjust to increases in sales. However, the difference was stretched compared to the results obtained with the standardized instance, so it is difficult to conclude which was better.

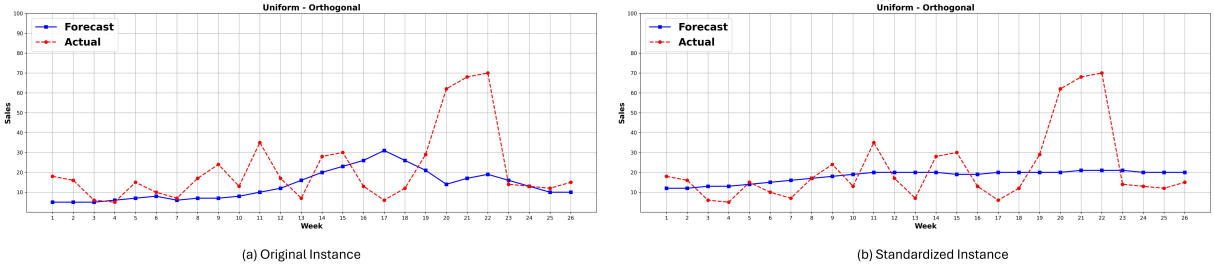


Figure 5. Results of weight initializer configuration B.

The cost function obtained a total of 1596 for the test with the instance in its original form, while a cost of 1580 was counted for the standardized version, in terms of sales forecasts with these initializers, demonstrating that the LSTM network obtained more significant savings with the standardized instance. Despite that, based on the narrow difference between costs, we observed that whether the instance was standardized was indifferent to the LSTM network initialized with this weight configuration.

3.1.3. Configuration C (Glorot, Orthogonal)

For this configuration, although the forecast slightly exceeded the actual sales peak, it is observed that, as with configuration A, there was a clear improvement in the results when using the standardized data instance (Figure 6).

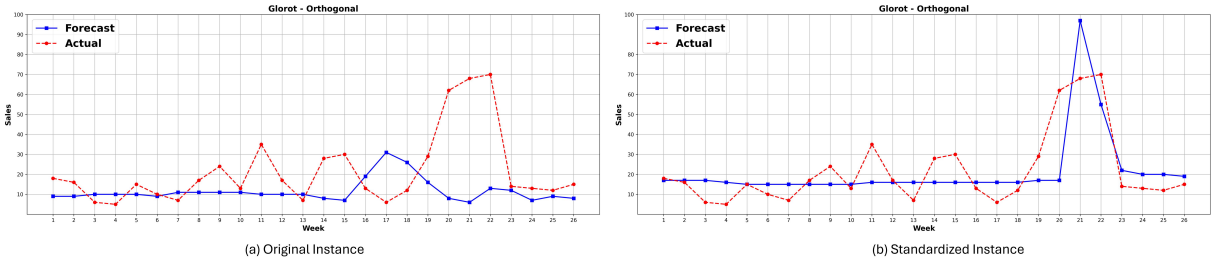


Figure 6. Results of weight initializer configuration C.

Similarly, when comparing the total cost of the test performed with the original instance (1856) with the results of the standardized instance (1214), a notable difference was observed between the results, thus positioning the instance standardized as the best with a more significant decrease in cost.

The results obtained in the tests demonstrated that, depending on the weight initializers used, the standardization of the instances can positively influence the results.

However, due to the need to establish a configuration for training the LSTM network, we decided to standardize the instances and use configuration C to initialize the weights. The above was because we observed that under these conditions the forecasts of the trained model allowed for more significant cost savings compared to the other configurations tested.

3.2. Instance Balancing

During our experimentation, we observed that for the training set, only a tiny portion of the resulting datasets contained events. For this reason, we decided to use an instance-balancing process for the training set. Instance balancing is a typical process used in machine learning to level the occurrence of classes in the instances so that all classes have the same impact on the algorithms, thus allowing them to perform better [35,36].

In the literature, diverse authors have used instance balancing to improve the results of their proposals, as in the case of Wongvorachan et al. [37] and Yang et al. [38]. In this sense, we considered an exciting alternative to adopting this concept in time series, to corroborate if this could help the LSTM network accurately learn how to identify its correct behavior in the presence of events.

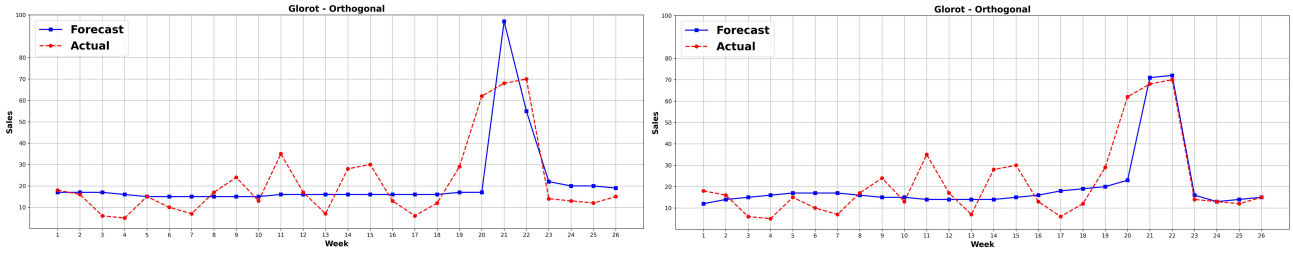
The balancing process consisted of duplicating the datasets containing events in the instances until we balanced them with those without events in the training set.

We designed Formula (1) to balance the instances. In this formula, R represents the number of times the balancing process will duplicate the event-containing datasets, S_{wo} represents the number of datasets that end in a week without an event, and S_w represents the number of datasets that end in a week with an event:

$$R = \frac{S_{\mathrm{wo}} - S_{\mathrm{w}}}{S_{\mathrm{w}}} \tag{1}$$

After standardizing the instances, we tested the best configuration found, Configuration C, with the balanced and standardized instances. The objective was to evaluate the potential of this balancing approach and determine its relevance for training the LSTM network to improve forecast accuracy.

Figure 7 presents the experimental results. For clarity, we compared the outcomes of the balanced instances with those of the unbalanced:



(a) Original Instance (b) Balanced Instance

Figure 7. Comparison of original instance and balanced instance.

The C configuration test results, combined with instance balancing, showed a considerable improvement compared to those obtained without balancing. This approach enabled the LSTM network to generalize more effectively for sales forecasting of lumberyard products.

3.3. Results by Product

Based on the genetic algorithm designed in this paper to optimize the LSTM network, we obtained the hyperparameter configurations shown in Table 8. Subsequently, we trained the network with these configurations, to obtain a model for each lumber mill product considered.

Table 8. Hyperparameter configuration for each product.

Hyperparameter	Product 1	Product 2	Product 3	Product 4	Product 5
LSTM 1 Units	309	336	354	157	157
LSTM 2 Units	225	193	380	384	290
LSTM 3 Units	279	399	311	311	340
Epochs	41	29	20	19	35
Batch size	57	10	121	217	98
Learning Rate	0.0098	0.0043	0.0071	0.0098	0.0054
Dropout 1	0.2367	0.4785	0.2399	0.3447	0.3721
Dropout 2	0.2668	0.3693	0.2459	0.2973	0.2039
Dropout 3	0.2040	0.2762	0.2395	0.3496	0.2650

We used the models obtained from the trained LSTM networks to prove the test data for each product. The goal was to evaluate the performance of the models in forecasting product sales, considering the customized cost function proposed in this paper. To illustrate the behavior of the trained models, Figures 8–12 provide a visual comparison between the actual and predicted values for each product.

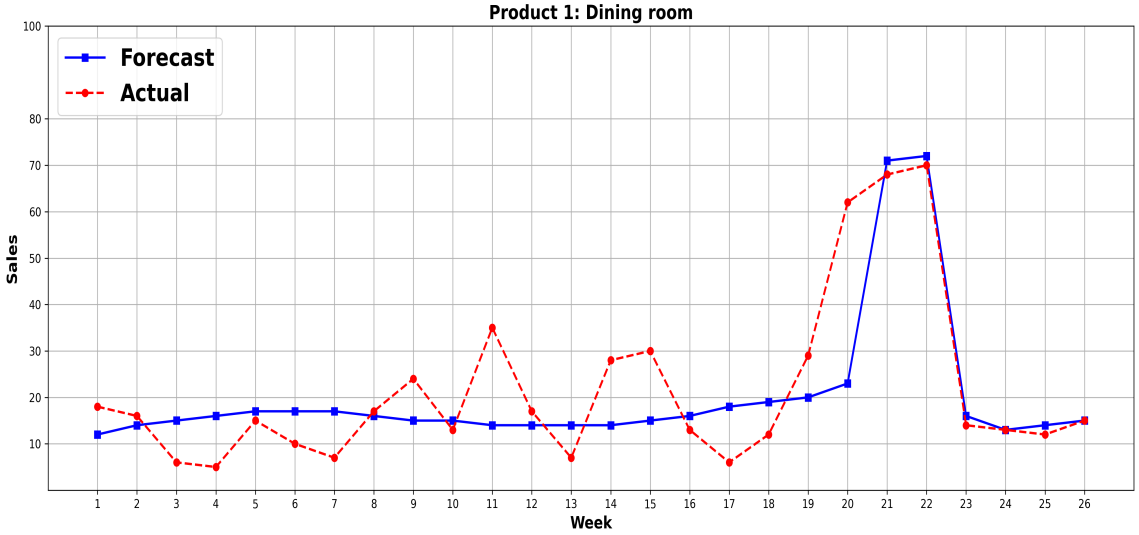


Figure 8. Comparison of actual and forecasted sales of product 1.

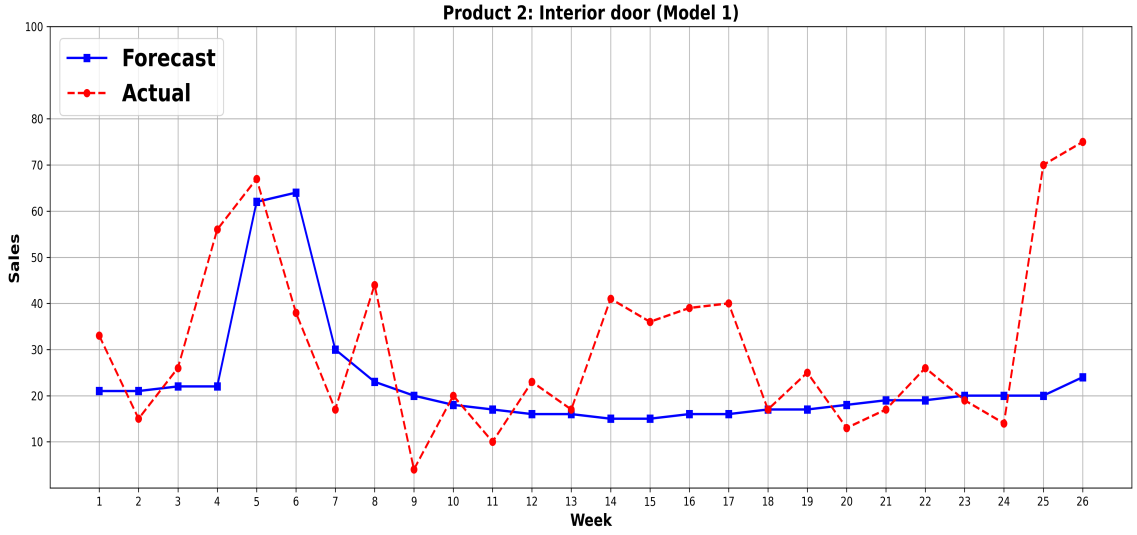


Figure 9. Comparison of actual and forecasted sales of product 2.

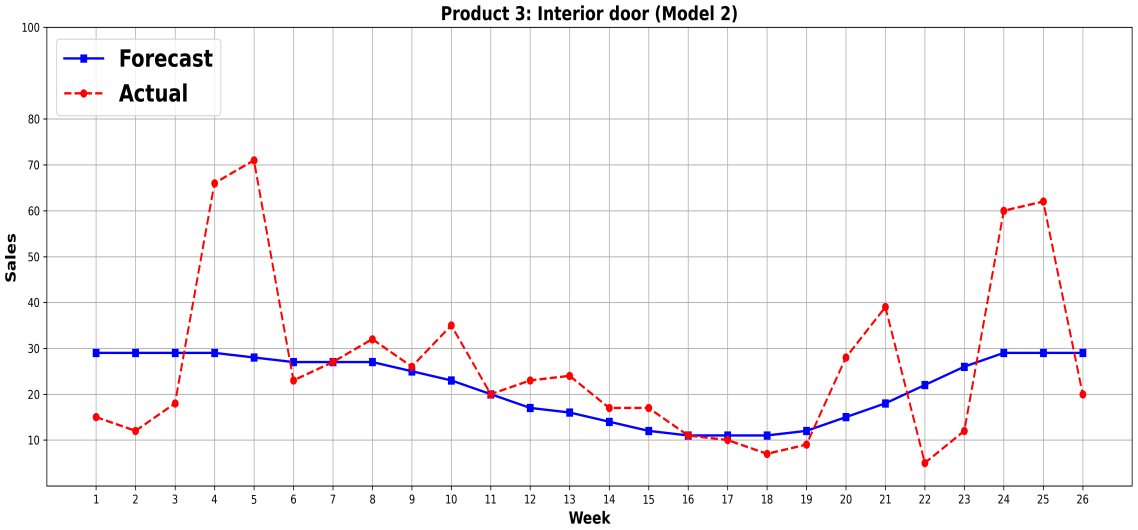


Figure 10. Comparison of actual and forecasted sales of product 3.

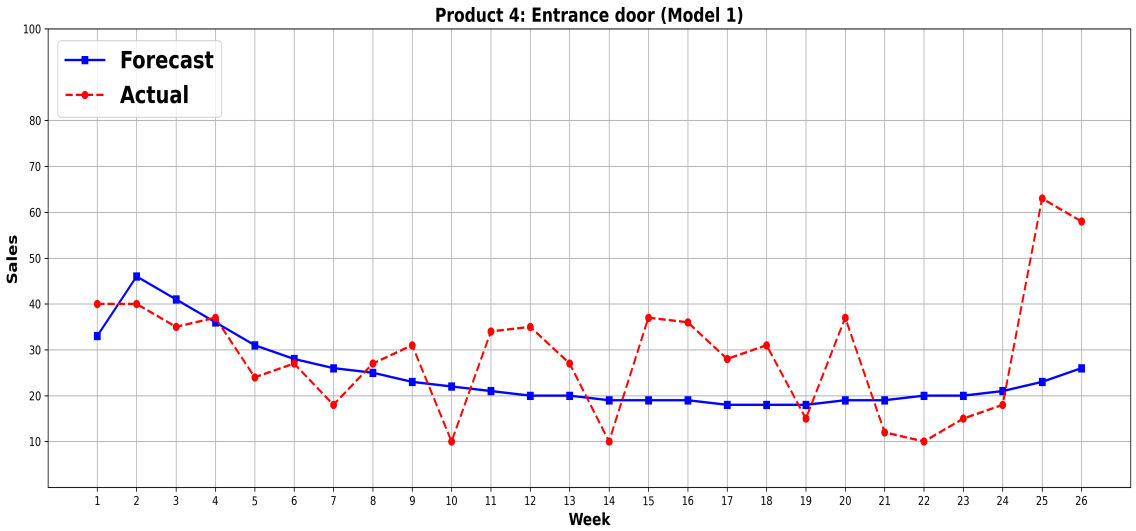


Figure 11. Comparison of actual and forecasted sales of product 4.

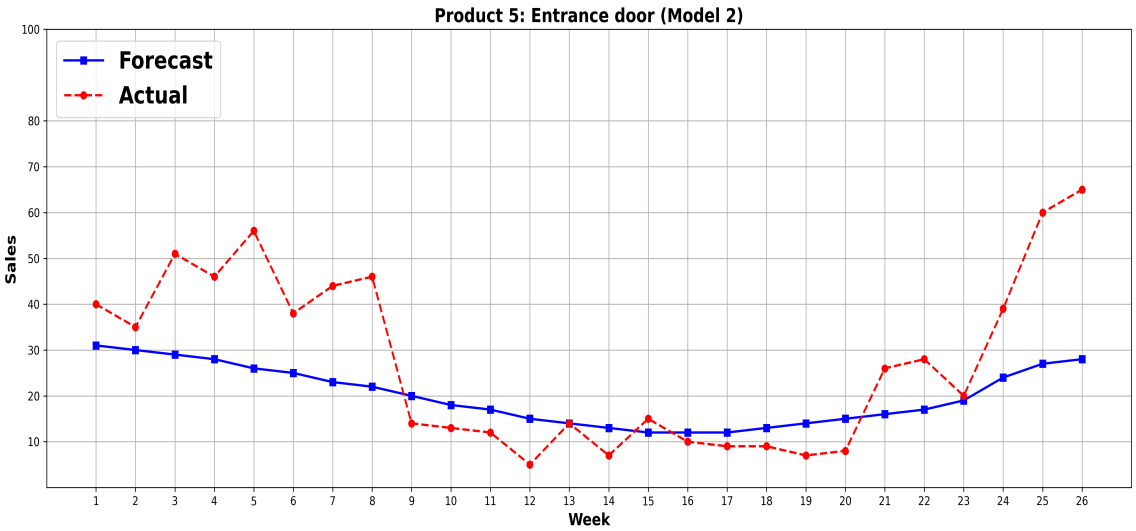


Figure 12. Comparison of actual and forecasted sales of product 5.

The results demonstrate that the LSTM network aimed to balance the total cost of the forecasts. Forecasts that exceeded actual values incurred expenses due to the operating cost of storing products in the lumber mill. Conversely, a forecast below actual sales led to expenses from producing additional products to meet demand. Table 9 presents each model's total cost calculated per product, the Root Mean Square Error (RMSE), and the Mean Absolute Percentage Error (MAPE) obtained for the LSTM network. The calculations were made with the forecast and actual values in the original scale of the time series.

Table 9. Total cost, RMSE, and MAPE obtained for each product.

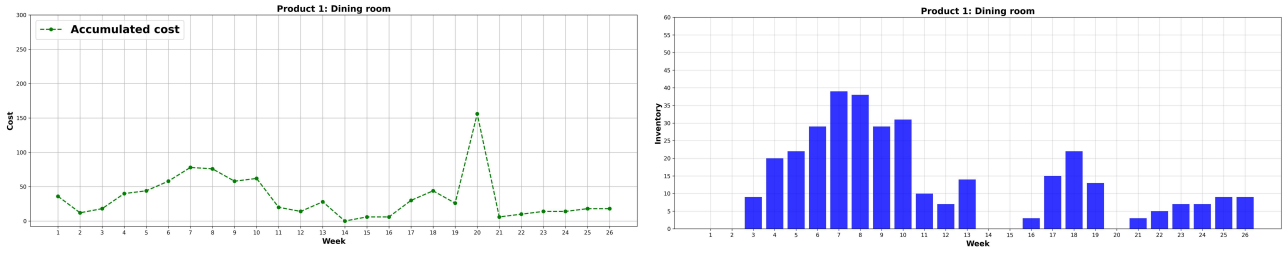
Product	Cost	RMSE	MAPE
1	892	11.139	53.539
2	1838	20.119	54.088
3	954	16.866	53.757
4	1272	13.859	40.348
5	2110	15.520	48.225

Although Table 9 shows an average MAPE of 49.99%, which is moderately high, we consider that the MAPE obtained was produced by the volatility of the sales for the products that the LSTM network could not adequately generalize. However, the LSTM network achieved an acceptable forecast of sales trends for all products. Based on MAPE, products 1 to 3 had similar behavior and were more complicated for the network to predict than products 4 and 5.

Moreover, we can observe that product 1 produced the lowest total cost and RMSE despite the MAPE obtained. In contrast, Product 2 had the second-highest total cost and the highest RMSE, suggesting that the network struggled to accurately forecast sales for this product, resulting in higher storage and production costs.

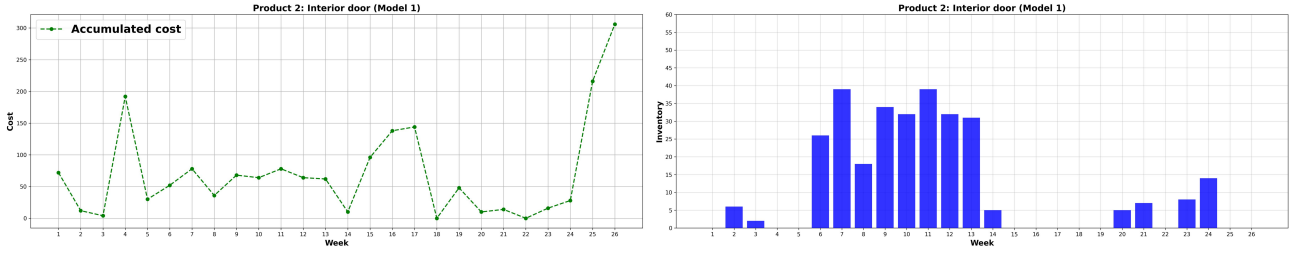
Products 3 and 4 produced moderately similar and intermediate results on cost and RMSE, indicating that the LSTM network reasonably well-identified sales trends for these products despite the considerable differentiation on the MAPE obtained for both. Product 5 incurred the highest cost, due to the network's difficulty accurately forecasting high sales values, though it still maintained an acceptable RMSE and the second-lowest MAPE.

We observed that the smoothing forecasts achieved by the LSTM network were able to produce a reasonable cost balance, which is essential for minimizing both storage costs and the cost of producing unplanned products. Figures 13–17 illustrate the behavior observed for the five products analyzed. We note higher forecasts higher than actual sales can lead to high storage costs. Maintaining an adequate inventory level can help address cases where forecasts fall short of actual sales.



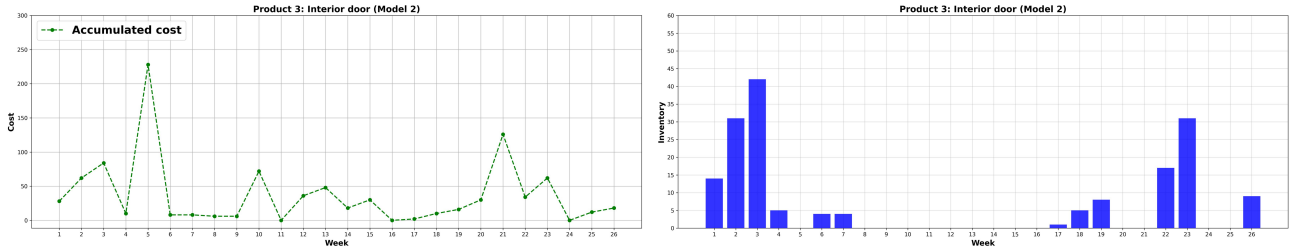
(a) Acumulated Cost (b) Storage

Figure 13. Cost of forecasts and inventory of product 1.



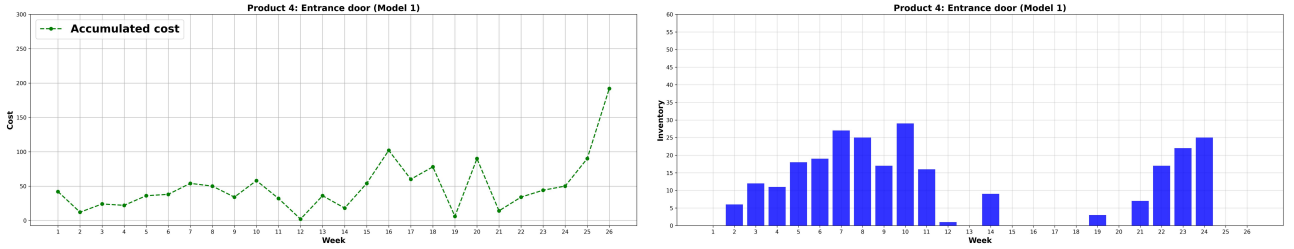
(a) Acumulated Cost (b) Storage

Figure 14. Cost of forecasts and inventory of product 2.



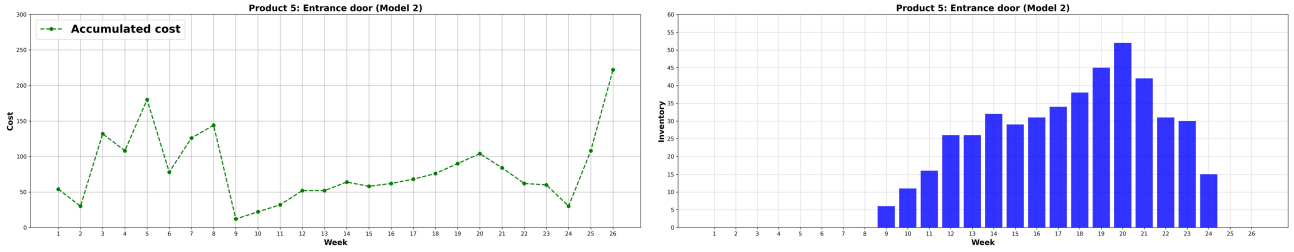
(a) Acumulated Cost (b) Storage

Figure 15. Cost of forecasts and inventory of product 3.



(a) Acumulated Cost (b) Storage

Figure 16. Cost of forecasts and inventory of product 4.



(a) Acumulated Cost (b) Storage

Figure 17. Cost of forecasts and inventory of product 5.

Finally, to assess the efficiency of the LSTM network results obtained with the best hyperparameters found with the genetic algorithm, we decided to compare the loss costs calculated for each product with the costs obtained by the LSTM network with the hyperparameters of the first execution of the genetic algorithm for each product. Table 10 presents the comparison and the relative variation of each product concerning the Initial Solution Found Cost (ISFC) obtained and the Best Solution Found Cost (BSFC) obtained by the LSTM network.

Table 10. Genetic algorithm found cost.

Product	ISFC	BSFC	Improvement
1	947	892	5.81%
2	2918	1838	37.01%
3	4102	954	76.74%
4	1910	1272	33.40%
5	3556	2110	40.66%

The results presented in Table 10 demonstrate that the genetic algorithm hyperparameter optimization improved the LSTM network performance in forecasting the sales trends of the D'Gallery lumber mill in Tamaulipas, Mexico, obtaining an average general improvement of 38.73%.

4. Discussion

The experimentation results show us that for the lumber mill case study, the conditions of the products observed made it possible for the custom loss function, the network structure, and the genetic algorithm proposed to address the forecasting sales trends in the lumber mill. In this sense, with the initialization of weights with the proposed configuration C, it was possible to improve the convergence and final performance of the LSTM network, which improved the cost of the production and storage of the lumber products.

Similarly, we observed that standardizing and balancing product instances is appropriate to improving the network's forecast results. Moreover, we noticed that the personalized loss cost function proposed in this paper for considering the cost of manufacturing and storage products was essential to adapting the LSTM network's learning to the specific needs of the lumber mill.

Although we observed higher volatility in the sales behavior of some products, despite working with the lumber mill's five most sold products, our preliminary tests showed good forecasting performance.

We established the range of the LSTM network hyperparameters and the parameters of the genetic algorithm implemented to explore the hyperparameter solution space; it was possible to train and test the LSTM network with each of the lumber products considered for this paper, to obtain models that allowed an adequate forecast of lumber sales.

The results obtained in the experimentation show that the proposed LSTM network and the genetic algorithm for optimizing its hyperparameters improve the forecasts for all products. Here, product 1 was the most complicated for the network, with an improvement of 5.81% compared to other products. We believe that this behavior came from the higher volatility of this particular product during all test time series in contrast with other products that, at specific periods, did have a more stable sales ratio. This behavior suggests that the sales forecasting between products is not the same and emphasizes the implication of producing a specific network for each product.

5. Conclusions

Loss management is a crucial aspect that directly impacts the profitability and stability of any company. The consequences of losses go beyond the financial aspect; they also affect the company's reputation, customer confidence, and employee morale. Thus, although losses are inevitable in virtually any context, their impact and frequency can be significantly reduced by implementing accurate and efficient forecasting strategies. Consequently, researchers have developed artificial intelligence techniques to perform forecasting tasks more efficiently and in less time than it would take to perform the process using traditional methods.

In this paper, we designed an LSTM network to forecast sales of a lumber mill in Tamaulipas, Mexico. The LSTM network was integrated with three LSTM layers with a dropout rate to prevent overfitting of the network and improve its ability to capture temporal dependencies and patterns in the sales data of the lumber mill products, thus optimizing its performance. Furthermore, we designed a custom loss cost function, which allowed a more precise evaluation of the performance of the LSTM network, including the particular needs of the data instances and the problem addressed in this paper. Similarly, we designed and implemented a genetic algorithm for the optimization of the hyperparameters of the LSTM network.

The experimental results show that the proposal of the personalized cost function, the structure of the LSTM network, and the genetic algorithm used for optimizing the LSTM hyperparameters contributed actively and crucially to improving the accuracy and efficiency of sales forecasts for the lumber mill products. Thus, the results obtained show an improvement for all the products considered in this paper compared with the results of the first execution of the genetic algorithm for optimizing the LSTM network, which reached an average improvement of approximately 38.73%. The results show that the lowest relative variation was found in product 1, with a value of 5.81%, and the most significant variation was found in product 3, with approximately 76.74%, which indicates that the genetic algorithm is suitable for optimizing the hyperparameters of the LSTM network.

Finally, as future work, we will consider testing other genetic operators and evolutionary algorithms for optimizing the hyperparameters of the LSTM network and, additionally, testing other variants of the LSTM network structure and proposing new customized functions that allow companies to consider other aspects of importance in addition to the storage and production costs.

Author Contributions: Conceptualization, A.H.G.-R. and L.J.G.-d.-Á.; methodology, L.A.H.-B.; software, A.H.G.-R. and H.A.H.-M.; validation, A.H.G.-R., H.A.H.-M. and L.A.H.-B.; formal analysis, A.H.G.-R. and R.P.-R.; investigation, L.J.G.-d.-Á.; resources, R.P.-R.; data curation, A.H.G.-R.; writing—original draft preparation, A.H.G.-R. and R.P.-R.; writing—review and editing, H.A.H.-M. and L.A.H.-B.; visualization, L.A.H.-B.; supervision, A.H.G.-R. and L.J.G.-d.-Á.; project administration, A.H.G.-R., L.J.G.-d.-Á. and R.P.-R.; funding acquisition, R.P.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The APC was funded by the Autonomous University of Tamaulipas.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The benchmark instances for this study are available at https://github.com/ahgarciar/Lumber_Mill_Instances, accessed on 20 September 2024).

Acknowledgments: A.H.G.-R. would like to acknowledge CONAHCyT Mexico for the SNII salary award.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Verstraete, G.; Aghezzaf, E.H.; Desmet, B. A leading macroeconomic indicators' based framework to automatically generate tactical sales forecasts. *Comput. Ind. Eng.* **2020**, *139*, 106169. [CrossRef]
2. Li, B.; Yang, Q.; Chen, J.; Yu, D.; Wang, D.; Wan, F. A Dynamic Spatio-Temporal Deep Learning Model for Lane-Level Traffic Prediction. *J. Adv. Transp.* **2023**, *2023*, 3208535. [CrossRef]
3. Tanaka, K.; Akimoto, H.; Inoue, M. Production risk management system with demand probability distribution. *Adv. Eng. Inform.* **2012**, *26*, 46–54. [CrossRef]
4. Gooijer, J.G.D.; Hyndman, R.J. 25 years of time series forecasting. *Int. J. Forecast.* **2006**, *22*, 443–473. [CrossRef]
5. Hyndman, R.; Athanasopoulos, G. *Forecasting: Principles and Practice*; OTexts: Melbourne, Australia, 2018.

6. Santos, I.S.L.; Oyuelos, E.G.D. *Previsión de Ventas y Fijación de Objetivos*; ESIC Editorial: Madrid, Spain, 2013.
7. Sohrabpour, V.; Oghazi, P.; Toorajipour, R.; Nazarpour, A. Export sales forecasting using artificial intelligence. *Technol. Forecast. Soc. Change* **2021**, *163*, 120480. [CrossRef]
8. Gupta, A.; Agarwal, P. Enhancing Sales Forecasting Accuracy through Integrated Enterprise Resource Planning and Customer Relationship Management using Artificial Intelligence. In Proceedings of the 3rd International Conference on Artificial Intelligence for Internet of Things (AIIoT), Vellore, India, 3–4 May 2024; pp. 1–6. [CrossRef]
9. Pietila, G.; Lim, T.C. Intelligent systems approaches to product sound quality evaluations—A review. *Appl. Acoust.* **2012**, *73*, 987–1002. [CrossRef]
10. Kumar, D.N.; Raju, K.S.; Sathish, T. River flow forecasting using recurrent neural networks. *Water Resour. Manag.* **2004**, *18*, 143–161. [CrossRef]
11. Mandal, S.; Prabaharan, N. Ocean wave forecasting using recurrent neural networks. *Ocean Eng.* **2006**, *33*, 1401–1410. [CrossRef]
12. Mandal, P.; Senjyu, T.; Urasaki, N.; Yona, A.; Funabashi, T.; Srivastava, A.K. Price Forecasting for Day-Ahead Electricity Market Using Recursive Neural Network. In Proceedings of the 2007 IEEE Power Engineering Society General Meeting, Tampa, FL, USA, 24–28 June 2007; pp. 1–8. [CrossRef]
13. Hansun, S.; Young, J.C. Predicting LQ45 financial sector indices using RNN-LSTM. *J. Big Data* **2021**, *8*, 104. [CrossRef]
14. Shen, M.L.; Lee, C.F.; Liu, H.H.; Chang, P.Y.; Yang, C.H. Effective multinational trade forecasting using LSTM recurrent neural network. *Expert Syst. Appl.* **2021**, *182*, 115199. [CrossRef]
15. Granata, F.; Nunno, F.D. Forecasting evapotranspiration in different climates using ensembles of recurrent neural networks. *Agric. Water Manag.* **2021**, *255*, 107040. [CrossRef]
16. Lu, W.; Li, J.; Wang, J.; Qin, L. A CNN-BiLSTM-AM method for stock price prediction. *Neural Comput. Appl.* **2021**, *33*, 4741–4753. [CrossRef]
17. Zhang, K.; Huo, X.; Shao, K. Temperature time series prediction model based on time series decomposition and bi-lstm network. *Mathematics* **2023**, *11*, 2060. [CrossRef]
18. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
19. Hodson, T.O. Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geosci. Model Dev. Discuss.* **2022**, *15*, 5481–5487. [CrossRef]
20. Krähenbühl, P.; Doersch, C.; Donahue, J.; Darrell, T. Data-dependent initializations of convolutional neural networks. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
21. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
22. Keras. Adam. Available online: https://keras.io/api/optimizers/adam (accessed on 21 October 2024).
23. Saxe, A.M.; McClelland, J.L.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Proceedings of the International Conference on Learning Represenatations, Banff, AB, Canada, 14–16 April 2014.
24. Padierna, L.C.; Carpio, M.; Rojas-Domínguez, A.; Puga, H.; Fraire, H. A novel formulation of orthogonal polynomial kernel functions for SVM classifiers: The Gegenbauer family. *Pattern Recognit.* **2018**, *84*, 211–225. [CrossRef]
25. de Pater, I.; Mitici, M. A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. *Neural Netw.* **2023**, *166*, 579–594. [CrossRef]
26. García Ruiz, A.H.; Ibarra Martínez, S.; Castán Rocha, J.A.; Terán Villanueva, J.D.; Laria Menchaca, J.; Treviño Berrones, M.G.; Ponce Flores, M.P.; Santiago Pineda, A.A. Assessing a Multi-Objective Genetic Algorithm with a Simulated Environment for Energy-Saving of Air Conditioning Systems with User Preferences. *Symmetry* **2021**, *13*, 344. [CrossRef]
27. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1998.
28. Sheng, Y.; Yang, Z.; Meng, Y.; Jia, B. A Study on the Design of a New Three-Dimensional Seismic Isolation Bearing Based on an Improved Genetic Algorithm for Bridge Engineering. *Appl. Sci.* **2024**, *14*, 9453. [CrossRef]
29. Lim, H.; Chung, K.; Lee, S. Multi-Objective Optimization of Pick-Up and Delivery Operations in Bike-Sharing Systems Using a Hybrid Genetic Algorithm. *Appl. Sci.* **2024**, *14*, 6703. [CrossRef]
30. Zhao, W.; Bian, X.; Mei, X. An Adaptive Multi-Objective Genetic Algorithm for Solving Heterogeneous Green City Vehicle Routing Problem. *Appl. Sci.* **2024**, *14*, 6594. [CrossRef]
31. Lobo, F.G.; Goldberg, D.E.; Pelikan, M. Time complexity of genetic algorithms on exponentially scaled problems. In Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO 2000), Las Vegas, NV, USA, 8–12 July 2000; pp. 151–158.
32. Haupt, R.L.; Haupt, S.E. *Practical Genetic Algorithms*; John Wiley: Hoboken, NJ, USA, 2004; p. 22.
33. Ruge, I.A.; Alvis, M.A. Aplicación de los algoritmos genéticos para el diseño de un controlador PID adaptativo Application of genetic algorithms for the design of an adaptive PID controller. *Tecnura* **2010**, *13*, 8.
34. Lim, S.M.; Sultan, A.B.M.; Sulaiman, M.N.; Mustapha, A.; Leong, K.Y. Crossover and mutation operators of genetic algorithms. *Int. J. Mach. Learn. Comput.* **2017**, *7*, 9–12. [CrossRef]
35. Mujahid, M.; Kına, E.; Rustam, F.; Villar, M.G.; Alvarado, E.S.; De La Torre Diez, I.; Ashraf, I. Data oversampling and imbalanced datasets: An investigation of performance for machine learning and feature engineering. *J. Big Data* **2024**, *11*, 87. [CrossRef]

36. Ghosh, K.; Banerjee, A.; Chatterjee, S.; Sen, S. Imbalanced Twitter Sentiment Analysis using Minority Oversampling. In Proceedings of the 2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan, 23–25 October 2019; pp. 1–5. [CrossRef]
37. Wongvorachan, T.; He, S.; Bulut, O. A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. *Information* **2023**, *14*, 54. [CrossRef]
38. Yang, Y.; Khorshidi, H.A.; Aickelin, U. A review on over-sampling techniques in classification of multi-class imbalanced datasets: Insights for medical problems. *Front. Digit. Health* **2024**, *6*, 1430245. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.