ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis

International Bachelor Econometrics and Operations Research

# Retail sales forecasting using LSTM and ARIMA-LSTM: A comparison with traditional econometric models and Artificial Neural Networks

Author: Cristian Cracan

Student ID: 474521cc

Supervisor: Utku Karaca

Second assessor: Kathrin Gruber

Date: 3 July 2020

**Abstract**

Aggregate retail sales is an important economic indicator that requires precise understanding and forecasting. This research extends the study of Alon, Qi, and Sadowski (2001), which makes use of three econometric models: Box-Jenkins ARIMA, Winters exponential smoothing, multivariate regression, and an Artificial Neural Network, to forecast retail sales in the US in two periods of time. The extension considers two additional methods: LSTM and ARIMA-LSTM and an additional extended period. The forecasting performances of the models are hence compared in three different scenarios. Overall, the results of the study are mixed with evidence that ARIMA and Winters models are the most consistent, while LSTM and ARIMA-LSTM have high accuracy in certain cases, such as in the extended period.

## Contents

**1 Introduction** **1**

**2 Literature** **2**

**3 Data** **4**

**4 Methodology** **5**

4.1 Forecasting . . . . . . . . . . 5

4.2 Econometric models . . . . . . . . . . 6

4.3 Artificial Neural Network . . . . . . . . . . 7

4.4 LSTM . . . . . . . . . . 10

4.5 ARIMA-LSTM . . . . . . . . . . 13

**5 Results** **13**

5.1 Comparison of errors . . . . . . . . . . 14

5.2 Test results . . . . . . . . . . 16

**6 Discussion and conclusion** **18**

**References** **21**

**A Algorithms for Machine Learning methods** **24**

**B Grid search results** **28**

## 1 Introduction

Private consumption is one of the main drivers of modern economies. Specifically retail trade provides an indicator of the domestic demand and it makes up a big part of national output, namely more than 5% of the American GDP for example (FRED, 2020). Therefore retail sales follow similar trends as the whole economy and they can be used to identify short-term trends in the final economic output. Forecasting aggregate retail sales is hence important from multiple economic and business standpoints. Especially during economic downturns, it is helpful for policymakers to have accurate forecasts of indicators such as retail sales to determine the level of the shock on the retail sector and on the economy. This can improve the quality of the response and of overall economic policies.

At business level, having accurate forecasts of retail sales can help businesses predict inventories and consumer spending, especially when facing less or more volatile demand than usual. Forecasting aggregate retail sales on market level can be crucial to planning and identifying potential growth channels (Fildes, Ma, & Kolassa, 2019). In particular forecasts that incorporate the strong seasonality of the retail sales can contribute to an efficient management of acquisitions and to a better prediction of revenues over the year. For example, forecasts for the holiday season around Christmas, marked by big increases, can help in correctly managing inventories. Hence, various functional areas of retail companies can benefit from accurate forecasting of sales especially as data and big data are increasingly utilized (Fisher & Raman, 2018). Furthermore, it could be possible to expand this field of study by predicting retail sales of particular companies in the US with the use of market shares and by including company specific characteristics.

At the same time, the study of time series forecasting has seen an increasing focus on new methods that are employed in various scenarios and fields of research. Given the monthly, quarterly or yearly frequencies of most economic time series, it is relevant to build robust and accurate models for variables with such characteristics. There is much evidence of Machine Learning techniques producing accurate forecasts such as Alon et al. (2001). Nevertheless when compared to standard econometric methods, model and architecture selection is often based on trial and error methods in the absence of sufficient theory. Moreover, overfitting the model on particular data sets can make the models unrobust. Hence, this research aims to contribute to the development of robust Machine Learning methods that can be used on different time series data with satisfactory results.

Recently, more advanced forms of Deep Learning methods such as Recurrent Neural Networks (RNN) or hybrid models have been investigated for forecasting time series. Thus, it is relevant to study the forecasting performance of these models in the context of retail sales relative to the econometric models and the ANN presented by Alon et al. (2001). Therefore this paper intends to study and implement two methods for forecasting aggregate retail sales in the US, namely LSTM and ARIMA-LSTM and compare their performance to the econometric methods implemented in Alon et al. (2001): Box-Jenking ARIMA, Winters smoothing model, multivariate regression and Artificial Neural Network. Hence the main research question is: *Do the LSTM and ARIMA-LSTM methods surpass the forecasting performance of traditional econometric models and of ANN for predicting aggregate retail sales in the US?* Also the performance of the new methods is compared with each other to identify the best one based on errors and tests. First sub-question is given by: *Which one of the two proposed methods exhibits the best forecasting performance?*

Additionally, given that Alon et al. (2001) explore two limited data sets from 1978 up to 1995, it would be reasonable to expand the data set. This would allow for a better comparison of the performance of the methods on a longer time span especially given that the Machine Learning techniques are usually trained on large amounts of data. Thus, the second sub-question becomes: *Which method performs the best on an expanded data set?*

Answers to these questions should present an overview of the best-performing methods that can be used to forecast retail sales in specific contexts. Future research can rely on these results to select the models appropriately or modify certain implementations to expand the research on this topic. As mentioned before, the Machine Learning research still has many aspects to cover until there can be a clear theoretical framework. On a practical side, both policymakers and businesses could use the findings and the methods of this paper to optimize their forecasting techniques. This way, it is hoped that the findings will bring positive practical changes beyond academic research.

The paper proceeds with a review of the Machine Learning methodology in the literature for forecasting time series. Next, the Data section presents general information on the used data series. The Methodology section is used to explain thoroughly the forecasting techniques, how forecasts are evaluated and the implementation of the models. Results are then displayed and discussed with advice for future research in the conclusion of the research.

## 2 Literature

Forecasting time series using Artificial Neural Networks (ANN) is a broad field of study with generally positive results. Applications cover: general time series (Ahmed, Atiya, Gayar, & El-Shishiny, 2010), environmental research (Feng et al., 2015), meteorology (More & Deo, 2003), hydrology (Daliakopoulos, Coulibaly, & Tsanis, 2005), finance (Guresen, Kayakutlu, & Daim, 2011), economics (Tkacz, 2001). For business tasks in various fields such as accounting, finance, marketing or sales, ANNs have been used widely with good accuracy overall, but there is still space left to improve their usability (Tkáč & Verner, 2016). In a wide research about the use of Artificial Neural Networks for forecasting, G. Zhang, Patuwo, and Hu (1998) find that ANNs are efficient for capturing non-linear patterns, but most implementations are inconsistent and lack systematic approaches, which leads to inconclusive results in the literature. There is also evidence that Neural Networks can effectively incorporate strong seasonal variations into the forecasts (Adhikari & Agrawal, 2012). On the other hand, G. P. Zhang and Qi (2005) find that neural networks fail to incorporate seasonal or trend variations effectively with data pre-processing being required. Alon et al. (2001) find that the ANN fares better than traditional econometric models in forecasting aggregate retail sales, while also being able to capture nonlinear trends and seasonality.

Recurrent Neural Networks (RNN) are derived from ANN, but the difference is that RNNs are interconnected along time sequences, which makes them suitable for identifying temporal patterns. RNNs are hence widely used for time series forecasting and show reasonable results in various fields of study (Kermanshahi, 1998; Mandal & Prabaharan, 2006; Hsieh, Hsiao, & Yeh, 2011).

A widely used RNN structure is the Long Short-Term Memory model (LSTM) (Hochreiter & Schmidhuber, 1997). LSTMs can capture long term temporal dependencies and do not suffer from the optimization problems of simple recurrent networks such as the problem of vanishing or exploding gradient (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2016). Hence, LSTMs have been used widely for forecasting economic data with high frequency like in finance, for stock markets movements (Chen, Zhou, & Dai, 2015; Nelson, Pereira, & de Oliveira, 2017; Tong, Shah, Cherukumalli, & Moulehiawy, 2018). Additionally, the method has been shown to be efficient for sales forecasting (Yu, Wang, Strandhagen, & Wang, 2017) and it has better results than ARIMA for sales of different products (Elmasdotter & Nyströmer, 2018). In general it has better forecasting performance for economic time series than ARIMA (Siami-Namini & Namin, 2018).

In addition to pure Neural Networks applications, some researchers also consider hybrids of econometric models and Neural Networks. This is done in order to capture the linear component using the ARIMA model and the nonlinear component using the Neural Network, which shows satisfactory improvements over basic models (G. P. Zhang, 2003). This model is subsequently improved in more recent papers with better forecasting results (Khashei & Bijari, 2010, 2011).

Additionally, Choi (2018) successfully implements an ARIMA-LSTM model for forecastig price correlation. Models combining traditional and Machine Learning techniques are applied in certain fields of study with success, meaning that these can be part of a potentially robust modelling framework for time series data (Jain & Kumar, 2007; Pai & Lin, 2005).

## 3 Data

The data for aggregate retail sales can be obtained in the Historical Data section of Monthly Retail Trade data offered by the United States Census Bureau (US Census Bureau, 2020). Figure 1 shows that retail sales have been increasing constantly over the last decades according to a positive trend with deviations occurring during economic downturns like the one in 2007-2009. Additionally, there is a clear seasonality in the data with increasing patterns over time. Nearly every year, there is a clear spike in sales in December during the holiday season and during the summer months.

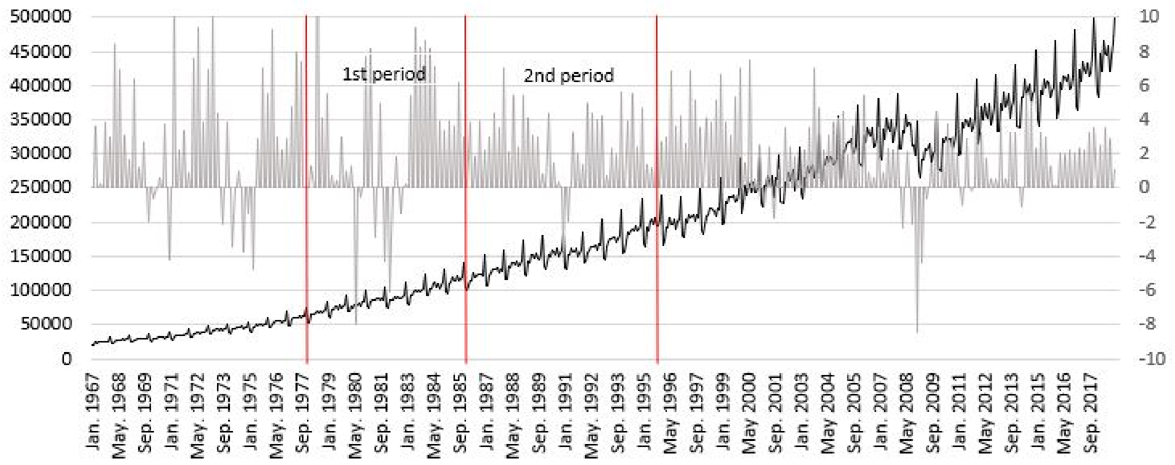

Figure 1: Monthly Aggregate Retail Sales (millions of US dollars) and quarterly GDP growth (%) (Bureau of Economic Analysis) in the US in the period 1967-2018

The two periods analyzed by Alon et al. (2001) span between 1978 to 1985 and from 1986 to May of 1995. The first period is more volatile with several recessions followed by high growth in the period from 1978 to 1983. On the other hand, the second period is characterised by stable growth with only one downturn in years 1990-1991 as can be seen in Figure 1. The extended period covers all the data from 1967 to 2018.

## 4 Methodology

This section first explains the forecasting and evaluation techniques used for all the models. This is followed by detailed explanations of the implemented models: the four from Alon et al. (2001) and the LSTM and ARIMA-LSTM models.

### 4.1 Forecasting

Two techniques are used for out of sample predictions. One step ahead forecasts are based on predicting only one observation into the future, using all the available information beforehand. This requires an estimation of the model each time using one more additional data point before performing a one step ahead forecast. Hence this technique is expected to incorporate the structural shocks better. On the other hand, multi-step ahead forecasts are more useful for practical applications. These are predictions done using the model fitted only on data before the prediction window.

The Mean Absolute Percentage Error (MAPE) is calculated for all forecasts to get a measure of accuracy, because it offers a good relative picture of the difference between the forecasts and the actual values. It is calculated as follows:

$$MAPE = N^{-1} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i} \tag{1}$$

where $\hat{y}_i$ represents the predicted value. Averages of one-step and multi-step ahead forecasts for the methods are also considered for each period. Moreover, total averages of all errors for each model are considered to create an overview of general performance.

For direct comparisons of forecasting performance between two methods, two tests are implemented. The Diebold Mariano (DM) test can be used to test whether two predictions are equally accurate. The test is based on comparing the mean of differences between squared errors of two predicted series, also called loss differential:

$$d_t = \epsilon_{1,t}^2 - \epsilon_{2,t}^2 \text{ and } \bar{d} = N^{-1} \sum_{t=1}^{N} d_t \tag{2}$$

Compared to the implementation of the test by Alon et al. (2001), the test is based on the method from Franses et al. (1998). The difference between the two implementations is the way the sample standard deviation is computed. Given $P$ realizations $d_t$ for $t = T+h, ..., T+h+P-1$, the the sample standard deviation of the loss differential can be computed as $\hat{\sigma} = \hat{\gamma_0} + 2 \cdot \sum_{j=0}^{h-1} \hat{\gamma_j}$, where is $\hat{\gamma_j}$ is the autocovariance of order $j$. In this case $h = 0$, hence $\hat{\sigma} = \hat{\gamma_0}$. The Diebold Mariano test statistic with the null hypothesis of identical forecasts, which asymptotically follows the standard normal distribution is given by:

$$DM = \frac{\bar{d}}{\sqrt{2\hat{\sigma}N^{-1}}} \tag{3}$$

The results of the test are interpreted by checking p-values for significance at 10% and the sign of the test statistic for an indication about which method of the two compared is better.

The second test used for comparing the predicted values is Wilcoxon's signed-ranks test (SR), a non-parametric test, which is implemented using SciPy package in Python programming language (Virtanen et al., 2020) and is given by the statistic:

$$SR = \sum_{t=1}^{N} I_{+}(d_t) rank(|d_t|) \tag{4}$$

where $I_{+}(d_t)$ is a function which takes a value of 1 if $d_t$ is positive and 0 otherwise. The test gives more importance through a larger weight to observations with bigger errors. The test can be used to study whether two samples come from the same distribution and its results are interpreted similarly to the DM test. The test statistic follows a standard normal distribution after scaling using:

$$\frac{SR - N(N+1)/4}{\sqrt{N(N+1)(2N+1)/24}} \tag{5}$$

Both tests are applied to compare the performance of the ANN, LSTM and ARIMA-LSTM with the other models. Additionally, comparisons between ANN and the two new methods LSTM and ARIMA-LSTM are made this way.

## 4.2 Econometric models

**Box-Jenkins ARIMA:** Autoregressive integrated moving average (ARIMA) models are widely used for time series analysis. ARIMA models are based on a regression of time series data on autoregressive (AR) component and moving average (MA) component with a possible initial differencing of $d$ times for establishing stationarity. The AR component refers to the lags with number $p$ of the time series, while the MA component refers to the lags with number $q$ of regression errors. Henceforth an ARIMA(p,d,q) model with the time series X has the general form:

$$diff(X, d) = AR(p) + MA(q) + \epsilon \tag{6}$$

The Box-Jenkins ARIMA method refers to the approach of finding the best ARIMA model for fitting the time series (Box, Jenkins, Reinsel, & Ljung, 2015). This approach consists of model selection, that involves making the time series stationary and identifying the required lags for AR and MA components, parameter estimation, evaluation and diagnostic checking such as checking residual autocorrelation. The implementation of Box-Jenkins ARIMA is done using Eviews programming language. The best models for all periods are selected as follows: for 1978-1985 period the ARIMA(12,2,11) model, for 1986-1995 and 1967-2018 periods the ARIMA(11,1,11) model.

**Winters' exponential smoothing:** The second econometric model used for forecasting sales is based on the method from Winters (1960). Exponential smoothing models are based on adapting the coefficients on recent observations by giving exponentially decreasing weights to older observations. It is used primarily for time series analysis and can be designed to take into account both the linear trend and the seasonality of the data. The component of seasonality can be either additive or multiplicative, the latter meaning that the pattern of seasonality is increasing over time. As can be seen in Figure 1 on the long term, seasonality can be considered multiplicative as the differences between months are more obvious in the last decades. Nonetheless in the two shorter periods, this process is not so evident. Hence, the model with additive seasonality is chosen for these two periods. The model with additive seasonality is given by:

$$sales_{t+k} = a + b \cdot k + c_{t+k} \tag{7}$$

where $a$, $b$ and $c$ are the permanent, trend and seasonality components respectively, that are all calculated recursively using damping factors. This implementation is given by the Eviews tool for exponential smoothing forecasting.

**Multiple regression:** The final econometric model used in the research is a linear regression of sales data on monthly dummies and a trend variable. This can be done as following:

$$sales_t = \beta_o + \sum_{i=1}^{11} \beta_i \cdot d_{i,t} + \beta_{12} \cdot t + \epsilon_t \tag{8}$$

where one monthly dummy variable is excluded. The point of this model is to see if a model that only captures linearly the seasonality and the trend is capable of producing accurate forecasts in different scenarios. It is implemented in Python using scikit-learn package(Pedregosa et al., 2011).

## 4.3 Artificial Neural Network

The Artificial Neural Network model is a widely used efficient method for forecasting. Its main idea is about passing the input information through multiple transformations using weights that are updated to minimize the error of the output. The transformations happen in multiple nodes, also called neurons, that are organized in one or multiple hidden layers. The input data is transformed using the weights, summed and passed through a function if selected in every neuron, whose value can be passed to neurons in another layer. Given the complexity of the functions and transformations taking place in a Neural Network, it is popular for finding non-linear and hidden patterns in data, but this comes at a cost of more computational complexity relative to linear models. A Neural Network with 10 input layer nodes, one hidden layer with 10 nodes and one output has 110 weights to optimize. Identity transfers functions between layers can also make the relations more complex. Hence, the most frequent criticism of ANNs is its limited capacity of explaining the effect of the inputs and its reduced interpretability. But when the model is used solely for forecasting it has good results thanks to its ability of capturing patterns and non-linearities.

The used Artificial Neural Network is a feedforward network with a backpropagation algorithm for optimizing the weights of the network. For the three layer network with one hidden layer and one output in Figure (2), the number of nodes in the input layer is equal to 12, representing the number of features that are 11 monthly dummies and the trend, and the hidden layer has 8 hidden nodes as selected. Both layers have one additional bias node.

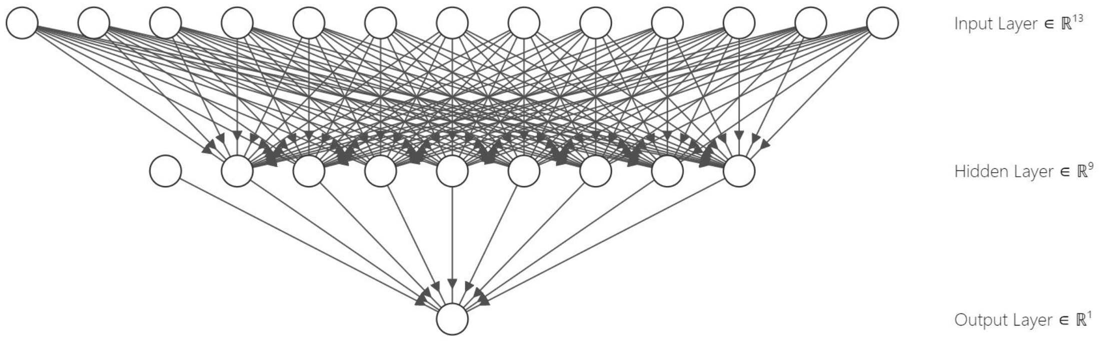
Input Layer ∈ $\mathbb{R}^{13}$
Hidden Layer ∈ $\mathbb{R}^{9}$
Output Layer ∈ $\mathbb{R}^{1}$

Figure 2: ANN architecture

The 113 weights connecting the layers are initialised according to the Nguyen and Widrow (1990) algorithm, which assigns to each hidden node its own interval. $H$ hidden nodes means that weights will be in the interval of length $2/H$ or $2/(0.7 \cdot H)$ to create overlapping intervals. Given that the sigmoid transfer function $sigmoid(w_i x + w_{bi})$ is linear over $(-1, 1)$, it can be obtained that $-1/w_i - w_{bi}/w_i < x < 1/w_i - w_{bi}/w_i$. This interval has length $2/|w_i| = 2/H$. Moreover, bias weights are initialised such that $x$ is in the region $(-1, 1)$. Hence bias weights are uniform values between $-|w_i|$ and $|w_i|$. The algorithm guarantees that the weights are initially in the input space, which makes the training process faster. The algorithm is implemented using Matlab function *initnw*.

The feedforward step means that the input data for the whole sample is passed through the network. All inputs are multiplied with the corresponding weights and summed in each hidden node. Alon et al. (2001) use a sigmoid transfer function in the hidden layer, meaning that the values in the hidden nodes are transformed before being passed to the output layer, as follows:

$$output_t = \alpha_0 + \sum_{i=1}^{8} \cdot \alpha_i \cdot sigmoid(\beta_{i,0} + \sum_{j=1}^{12} \beta_{ji} \cdot x_{j,t}) + \epsilon_t \tag{9}$$

The backpropagation algorithm is based on computing the gradient of the loss function, mean squared error $\frac{1}{n}\sum_i^n (y_i - \hat{y_i})^2$, with respect to the weights. The weights are updated towards achieving the minimum of the error function. This process is repeated several times, called epochs in Machine Learning terminology. The feedforward-backpropagation mechanism makes Neural Networks learn, as information is passed through the network to get the predictions and the network is improved based on errors. This is done until the stopping rule is attained or the error is no longer changing.

The training algorithm used in the network is the Levenberg-Marquadt (LM) algorithm combined with Bayesian Regularization (BR). The LM algorithm solves $argmin_\beta \sum_{i=1}^m [y_i - f(x_i, \beta)]^2$. and is a combination of Gradient Descent and Gauss-Newton methods. The algorithm uses Gauss-Newton when the algorithm is close to the optimum and Gradient Descent otherwise. Thanks to this approach, the LM algorithm is efficient in finding the minimum for problems without a large number of parameters. BR is used for improving the generalization of the network.

Both methods are implemented using Matlab function *trainbr* that combines them. A simplified version of the function is presented in Appendix A. During backpropagation, the Jacobian matrix with respect to weights and biases is computed and is used to update the parameters, that is done using $\mu$: $dX = -(JJ + I * \mu)\ JE$, where $JJ = JX * JX$, $JE = JE * JE$, $JX$ is the Jacobian matrix with respect to the parameters and $JE$ is the Jacobian with respect to the errors. Initial $\mu = 0.005$ is updated iteratively according to increase factor $\mu_{inc} = 10$ or decrease factor $\mu_{dec} = 0.1$ to improve the performance measure. The algorithm stops when the maximum or minimum level of $\mu$, the optimization goal or the maximum number of epochs of 1000 are reached. Trials showed that a high number of epochs is rarely reached.

Finally, trials showed that there can be a big variation in forecasting results when training the network once. This can be caused by different initial weights and biases. Hence to generalize the forecasting results, the ANN is trained 100 times and averages of the forecasted values are used. Simplified algorithms that are implemented in Matlab for both techniques are presented in Appendix A.

## 4.4 LSTM

The Long Short Term Memory model has a more complex structure that allows it to approach specific problems more efficiently. Its name derives from the its ability to incorporate information from the past without underestimating or overestimating it like simple Recurrent Neural Networks. The RNN structure can be pictured as in Figure 3 (Olah, 2015), which shows a compressed version where vectors and layers are compressed into single units. The main difference relative to the ANN is the capacity of the RNN to train based on the outputs of previous time points. Every cell A has both an output and a connection to the next cell. The problem with the basic RNN structure is that during training and more specifically the backpropagation, the gradient can become vanishing or exploding. This means that long-term dependencies are not used properly for training, which results in suboptimal networks.

LSTM solves this issue with the special cell architecture presented in Figure 4 (Olah, 2015). This version is used in this paper as more complex structures were shown not to bring significant improvements (Breuel, 2015; Greff et al., 2016). The figure presents a simplified version of the cell for a better interpretation of the model. Some things to be noted are: two lines connected into one means that two vectors are concatenated into one; one line separated into two means that new lines carry copies of the vector, the yellow cells signify neural network layers with multiple hidden nodes; pink nodes signify pointwise operations.

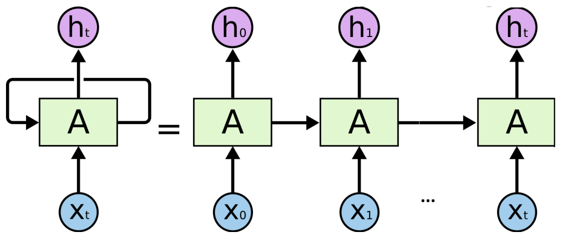

Figure 3: Basic RNN architecture

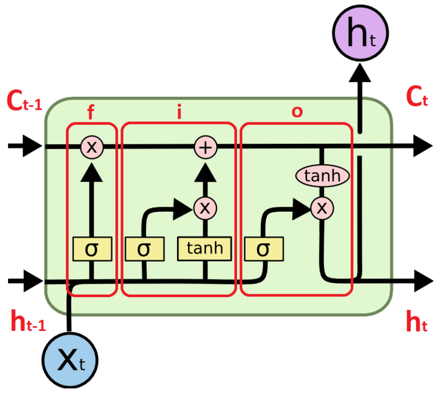

Figure 4: LSTM cell architecture

A crucial element is the cell state, represented by the upper most horizontal line which carries the $C_t$ variable. This is the main connection between states that can be modified with each time stamp. The $h_t$ on the other hand is called the hidden state. It is the output of the cell. Both $C_t$ and $h_t$ variables are vectors with the size equal to the number of hidden nodes, chosen as the measure of information that needs to be passed from state to state and is also used in all neural network layers explained below. Generally a high number of hidden nodes can lead to too much complexity and the optimal number depends on the inputs and the problem.

**Forget section:** The $f$ part in the Figure 4 is the mechanism responsible for determining which information from the previous cell will be excluded from the cell state. It operates on both the output from previous cell $h_{t-1}$ and the new data $x_t$. These values are passed into a neural network layer with the specified number of hidden nodes and a sigmoid activation function. The output of this is a vector with values between 0 and 1. By multiplying the output and the cell state, elements that are multiplied with small values or 0 will be forgotten, and vice versa for the ones multiplied with bigger values. This part with the multiplication has the name of forget gate.

**Input section:** The $i$ part in the Figure represents the mechanism for adding new information to the cell state. The same mechanism with a neural network layer with sigmoid activation and a forget gate is present. The previous hidden state and present inputs are first passed through a neural network layer with a tanh activation function $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ and then passed through the gate. The values that are preserved after passing through the gate are ultimately added to the cell state $C_t$ originating from the forget section.

**Output section:** The third section is meant to create the connection to the next cell and create the output. The same mechanism from the other two parts with forget gate is present here as well. First a copy of the cell state $C_t$ is transformed using the tanh function and then passed through the gate determined by past outputs and present inputs. This way the $h_t$ values are computed which are both passed to the next cell as hidden state and as output of the cell.

The three sections can be expressed mathematically as follows:

$$\text{Forget: } C_t = C_{t-1} \cdot sigmoid(w_{f,0} + w_f \cdot [h_{t-1}x_t]) \tag{10}$$

$$\text{Input: } C_t = C_t + tanh(w_{i2,0} + w_{i2} \cdot [h_{t-1}x_t]) \cdot sigmoid(w_{i1,0} + w_{i1} \cdot [h_{t-1}x_t]) \tag{11}$$

$$\text{Output: } h_t = tanh(C_t) \cdot sigmoid(w_{o,0} + w_o \cdot [h_{t-1}x_t]) \tag{12}$$

where $w_f$, $w_{i1}$, $w_{i2}$, $w_o$ and the biases are the weights to be updated.

The output $h_t$ is passed through a neural network layer with one output to obtain the value of sales in period $t$. This forward process through the cells is followed by backpropagation, which involves more advanced computations due to the architecture of the model than in the ANN. For LSTM computing gradients with respect to the weights is more complex given the connections between cells. The loss function becomes a sum of all losses for the time stamps by applying the

Chain Rule and summing the gradients as follows:

$$\frac{\partial Error}{\partial w} = \frac{\partial \sum_{t=1}^{n} Error_t}{\partial w} = \sum_{t=1}^{n} \frac{\partial Error_t}{\partial w} \tag{13}$$

With each gradient of the error containing information from other cells. All weights are updated according to the gradients and the Adam optimization algorithm. Adam is chosen because of the high number of weights to be updated and it being an efficient optimizer used for Deep Learning applications (Kingma & Ba, 2014). The main idea behind the algorithm is storing a bias corrected exponential moving average of both past gradients and squared gradients. Parameters are updated using $\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t$, where $m_t$ and $v_t$ are estimates of the mean and of the uncentered variance of the gradients respectively (Ruder, 2016).

The inputs for the implementation of the basic LSTM are the dummies and trend used in the ANN and additional 12 time lags given the yearly seasonality of the data. Moreover, the inputs, output variables and the trend are standardized as it improves the training process and the accuracy. The method is implemented in Matlab using the functions $lstmLayer$, $fullyConnectedLayer$ and $regressionLayer$. Several hyperparameters need to be specified before training the model. Given the specificity of Deep Learning models, there is no precise theory for setting these and a trial and error method is required. Hence, a grid search based on forecasting the last 12 months is implemented for finding the optimal combination of hyperparameters across all periods. The training algorithm consists of all months up to the last year. Given the connections between different time steps in the LSTM, forecasts are done by inputting all training data combined with the test data. This is done in order to initialise the cell and hidden states before doing the forecasts for the last 12 months. The algorithms for the forecasting implemented in Matlab can be found in Appendix A. Additionally similar to the ANN, the model is trained 20 times and the average MAPE is used.

The grid search results, that can be found in Appendix B showed that a learning rate of 0.05 is the most suitable in all scenarios and given the limited computational capacity it was kept fixed. Hence, the grid search focused on finding the optimal number of hidden nodes and maximum number of epochs. For all three periods it is found that the most optimal combination with consistently good errors across all three is 30 hidden nodes and 90 epochs. These parameters are hence used in both one step and multi-step forecasts for the three periods.

## 4.5 ARIMA-LSTM

This method is based on the idea of combining the econometric and the Machine Learning methods for capturing both linear and non-linear patterns in data. Even though Box-Jenkins ARIMA is highly efficient for linear models, many non-linear patterns are not adequately captured. Given the complexity of the relations that might dictate the patterns in the residuals, it is interesting to test whether the LSTM model can be used to detect the non-linear part of the data. The forecasts are hence produced in the following manner:

$$\hat{y_t} = \hat{L_t} + \hat{N_t} \text{ for time step t} \tag{14}$$

where $\hat{L_t}$ is the forecast of the ARIMA model, representing the linear part, and $\hat{N_t}$ is the forecast of the ARIMA residuals obtained using the LSTM. The model is a correction of the linear model and the final goal is to improve it significantly as to justify the use of a more complex model as a second step.

The best ARIMA model found for each period and the respective forecasts for one-step and multi-step types are used for the linear part. While for the LSTM, the inputs become the residuals of the ARIMA model, that are standardized. A 12 period lag is not as appropriate as it was for pure LSTM model as most seasonality is explained by the linear model. After multiple trials, a lag of three is determined to be the most suitable. Hence the LSTM is trained on three lags of ARIMA residuals, while keeping the monthly dummies and the trend as features as well. This is done in order to tackle any seasonality or linearity left in the residuals. The hyperparameters of the used LSTM model are the same as in basic application with the exception of the learning rate: 30 nodes, 90 maximum epochs and a learning rate of 0.001 is used. The change is done because the predicted residuals are more accurate when using a smaller initial learning rate based on trials. In general the grid search results in Appendix B for ARIMA-LSTM showed less variation than in the case of the LSTM. The forecasts of the residuals are done in the same way as the forecasts of the sales in the LSTM displayed in Appendix A.

## 5 Results

The results of the research are obtained on a computer with Intel Core i5-7200U CPU running at 2.50 GHz and using an installed memory of 8.00 GB on Microsoft Windows 10.

In general, the implemented methods are accurate in predicting the aggregate retail sales in all three studied periods, with small exceptions and variations. The forecasts can be seen in Figure 5.

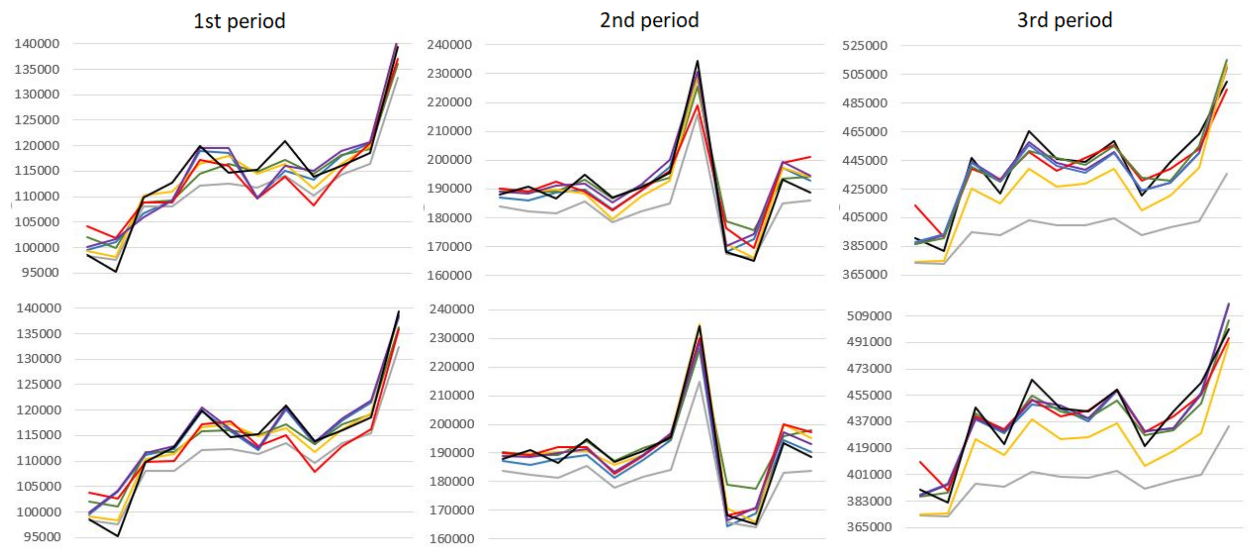

Figure 5: One-step and multi-step ahead forecasts for the three periods (actual: black, ARIMA: blue, Winters: green, Regression: grey, ANN: yellow, LSTM: red, ARIMA-LSTM: purple)

Most methods, especially the econometric models, with the exception of the regression follow accurately the trend and fluctuations for individual months, meaning that the seasonality and trend are incorporated. The Machine Learning methods also result in accurate results, but it can be seen that the LSTM in the first period underestimates the values for most months, which is similar to the ANN in the extended period. Nevertheless, it is most reasonable to compare the statistical errors and the test results to make solid conclusions about the research.

## 5.1 Comparison of errors

Table 1: Average MAPEs with rankings of the implemented models over all periods

| | ARIMA | Winters | Regression | ANN | LSTM | ARIMA-LSTM |
|---|---|---|---|---|---|---|
| Tot. Average | 1.94 (1) | 2.03 (3) | 5.49 (6) | 2.88 (5) | 2.56 (4) | 2.00 (2) |

Based on the average errors across the three different periods in Table 1 it can be noticed that the ARIMA model, its hybridization using LSTM and the Winters exponential smoothing are the most suitable methods for forecasting the Aggregate Retail Sales. The errors of nearly 2% show that it is possible to get accurate forecasts of aggregate retail sales for both one-step and multi-step ahead predictions into the future. Nonetheless given large variations across the periods, it is most reasonable to look at each separately. Moreover, it can noticed that compared to Alon et al. (2001), the results and rankings for the first four methods hold in general with slightly bigger errors in most cases, which could be explained by different model choices. The biggest divergence from the initial paper is the results of the ANN in the second period, where the authors obtained more satisfactory results. This could be influenced by special data preparation or training choices which are not explained in depth and unfortunately could not be replicated fully.

Table 2: MAPE results with ranking in the brackets

| | 1978 - | 1985 | | 1986 - | 1995 | | 1967 - | 2018 | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | One | Multi | Aver. | One | Multi | Aver. | One | Multi | Aver. |
| ARIMA | 2.61(3) | 1.81(2) | 2.21(3) | **1.82**(1) | 1.79(2) | **1.8**(1) | 1.9(3) | 1.76(2) | **1.83**(3) |
| Winters | 2.27(2) | 2.03(4) | 2.15(2) | 2.07(3) | 2.37(4) | 2.22(3) | 1.85(2) | **1.65**(1) | **1.75**(1) |
| Regression | 2.99(5) | 3.29(5) | 3.14(5) | 3.59(6) | 4.02(6) | 3.81(6) | 9.46(6) | 9.63(6) | 9.55(6) |
| ANN | **1.78**(1) | **1.69**(1) | **1.73**(1) | 2.91(4) | 2.96(5) | 2.94(5) | 3.77(5) | 4.2(5) | 3.99(5) |
| LSTM | 3.31(6) | 3.32(6) | 3.32(6) | 2.94(5) | 1.90(3) | 2.42(4) | 2.13(4) | 1.76(2) | 1.94(4) |
| A-LSTM | 2.89(4) | 1.87(3) | 2.37(4) | 1.99(2) | **1.64**(1) | **1.81**(2) | **1.72**(1) | 1.87(4) | **1.79**(2) |

The **first period** from 1978 to 1985 with a forecast period of the year 1985, is a period generally described by increased macroeconomic volatility in the US. For both one-step and multi-step ahead forecasts, the Artificial Neural Network proposed by Alon et al. (2001) performs the best with an average of 1.73%. This can be explained by the ability of Neural Networks to capture non-linear patterns which are assumed to be higher in this period. Nonetheless, the LSTM model which is built to spot non-linear patterns across time, has the worst performance for the period with an error of 3.32%. Given the LSTM's performance in this period, the ARIMA-LSTM model also fails to improve the forecasts of the ARIMA model .

In general, apart from the multiple regression and LSTM, all methods perform better for the 12 months ahead forecasts. Even if this is encouraging from the point of the practicality of the models, this can also show that the forecasts are negatively influenced by incorporating more observations. This creates doubts about the ability of the studied models to predict sales in years with unexpected trends or economic shocks. Given all these results, it can be concluded that the ANN is the best choice for this period characterized by increased volatility.

For the **second period**, the rankings are different. As this period had more macroeconomic stability, the econometric models that capture the linear patterns are expected to perform better. Results show that ARIMA and the hybrid model ARIMA-LSTM perform the best for one-step ahead forecasts with a MAPE of 1.82% and multi-step ahead forecasts with 1.64% respectively. Nevertheless, the improvements of the hybrid model are marginal and the averages for both models stand at nearly 1.8%. The ANN has a worse performance in this period, but the LSTM has better accuracy especially for the multi-step ahead forecasts making it a rival even for ARIMA. This is contrary to the expectation that the LSTM would fare worse with more linear data.

When comparing one-step and multi-step ahead forecasts, half of the methods fare worse for the multi-step ahead forecasts. An interesting finding is that the LSTM is better both for its initial application and in the hybrid model for the multi-step ahead forecasts. This means that additional information might unnecessarily complicate the model. All in all, in the second period with low volatility, the best model is the linear Box-Jenkins ARIMA model given its high accuracy and its ease of implementation compared to the ARIMA-LSTM which fails to deliver significant improvements.

The **third period** is by far the longest of the studied covering all years from 1967 to 2018 with a forecasting window of 12 months in the last year. Contrary to the expectations, the ANN performs worse than in the short periods with nearly a 4 % error. But the LSTM has improved results on both techniques. It is only marginally worse than the Winters model with a 1.76 % error in the multi-step ahead forecast. The Box-Jenkins ARIMA model and the Winters model both produce highly accurate predictions with averages of 1.83% and 1.75% respectively. The ARIMA-LSTM model only marginally improves the forecast for one-step ahead technique and worsens them for the multi-step, meaning that adding the residuals forecasted by LSTM does not improve accuracy.

In addition to this, it can be noticed that most models are consistent across one and multi-step ahead forecasts. ARIMA, Winters and LSTM all perform better on a multi-step ahead prediction of sales, meaning that these can have provide good practical applications. For this period the econometric models ARIMA and Winters smoothing are the best-performing models given the obtained accuracy and relative implementation ease compared to the Machine Learning techniques, but the low average errors of LSTM and its hybridization also point to satisfactory performance of these models on a longer time span.

## 5.2 Test results

The results of the Diebold Mariano and Wilcoxon Signed-Rank tests as implemented in Alon et al. (2001) for the multi-step ahead forecasts are discussed below. Comparisons are done for each separate Machine Learning models: Artificial Neural Network (ANN), Long short term memory

(LSTM) and ARIMA-LSTM.

Table 3: Comparisons with the ANN

| | 1978 - 1985 | | 1986 - 1995 | | 1967 - 2018 | |
|---|---|---|---|---|---|---|
| | DM | SR | DM | SR | DM | SR |
| ARIMA vs ANN | 0.73(0.23) | **-1.73(0.04)** | 0.72(0.24) | **2.59(0.0)** | **-3.13(0.0)** | **-3.06(0.0)** |
| Winters vs ANN | **1.45(0.07)** | -1.26(0.1) | **1.66(0.05)** | -0.86(0.19) | **-4.31(0.0)** | **-3.06(0.0)** |
| Regr. vs ANN | **3.09(0.0)** | **3.06(0.0)** | **2.54(0.01)** | **3.06(0.0)** | **5.0(0.0)** | **3.06(0.0)** |

Table 3 shows that in the first period the ANN significantly outperforms the multiple regression according to both DM and SR tests, also outperforms Winters model according to the DM, yet it is outperformed by ARIMA according to the SR test. The same test results hold in the second period except for the sign of the comparison with ARIMA, which is surprising given its smaller error. In the extended period, both tests show that the ANN predictions are worse than the ARIMA and Winters models, but it is better than the multiple regression forecasts. This is completely in line with the findings from the previous section where the ANN is outperformed by ARIMA and Winters.

Table 4: Comparisons with the LSTM

| | 1978 - 1985 | | 1986 - 1995 | | 1967 - 2018 | |
|---|---|---|---|---|---|---|
| | DM | SR | DM | SR | DM | SR |
| ARIMA vs LSTM | **-1.91(0.03)** | **-2.04(0.02)** | -0.29(0.39) | **3.06(0.0)** | 0.45(0.32) | 0.47(0.32) |
| Winters vs LSTM | **-2.83(0.0)** | **-1.8(0.04)** | 1.24(0.11) | -0.94(0.17) | -0.49(0.31) | 1.18(0.12) |
| Regr. vs LSTM | 0.4(0.34) | **2.51(0.01)** | **2.23(0.01)** | **3.06(0.0)** | **5.13(0.0)** | **3.06(0.0)** |
| ANN vs LSTM | **-3.29(0.0)** | -0.94(0.17) | **-2.17(0.01)** | -0.08(0.47) | **3.32(0.0)** | **3.06(0.0)** |

Table 4 displays the results of the comparisons between the initial four methods from Alon et al. (2001) and the LSTM. In the first period, it can be seen that the LSTM is outperformed by ARIMA and Winters model according to both DM and SR, which is in line with the errors. Also DM shows that it is outperformed by ANN, and it only outperforms the regression according to SR. In second period, both DM and SR find that the LSTM outperforms the regression and it outperforms ARIMA according to the SR test. In the last period, both tests shows that when compared to the Regression and the ANN, LSTM outperforms these two models significantly.

Table 5 shows the results for the forecast comparison between all methods and ARIMA-LSTM. In the first period, the DM test shows that the method is outperformed by ARIMA and surpris-

Table 5: Comparisons with the ARIMA-LSTM

| | 1978 - 1985 | | 1986 - 1995 | | 1967 - 2018 | |
|---|---|---|---|---|---|---|
| | DM | SR | DM | SR | DM | SR |
| ARIMA vs A-LSTM | **-1.4(0.08)** | **3.06(0.0)** | 0.91(0.18) | **3.06(0.0)** | **1.27(0.1)** | 1.02(0.15) |
| Winters vs A-LSTM | -0.32(0.38) | **1.8(0.04)** | **1.86(0.03)** | **-1.73(0.04)** | -1.02(0.15) | **1.96(0.02)** |
| Regr. vs A-LSTM | 1.13(0.13) | **3.06(0.0)** | **2.65(0.0)** | **3.06(0.0)** | **5.35(0.0)** | **3.06(0.0)** |
| ANN vs A-LSTM | -0.8(0.21) | **2.12(0.02)** | -0.27(0.39) | **-1.33(0.09)** | **3.26(0.0)** | **3.06(0.0)** |
| LSTM vs A-LSTM | **-1.63(0.05)** | **-2.2(0.01)** | **-1.31(0.1)** | **2.04(0.02)** | 0.25(0.4) | 0.47(0.32) |

ingly by the LSTM. The SR test results all show significant relations, with the ARIMA-LSTM being again outperformed only by the LSTM. In the second period, the forecasts of ARIMA-LSTM are significantly better than those of Winters and Regression and worse than those of the LSTM according to the DM. The SR test again shows significant differences across all comparisons with it being outperformed only by the ANN and Winters model contrary to the DM results, that can be explained by the focus of the DM test on the size of the error, while the SR test is focused mostly on the rank. The last period is marked by the key finding that all significant differences show that ARIMA-LSTM outperfoms all models except for insignificant differences with the LSTM.

Based on the results from all the comparisons for all the periods, we can conclude that the tests in large part follow the results from Table 2 of the errors. Hence the majority of the results from this section hold and can be used to discuss the final implications of the research.

## 6 Discussion and conclusion

This research paper had the goal of studying whether two advanced Machine Learning methods LSTM and ARIMA-LSTM outperform the methods used by Alon et al. (2001) for predicting aggregate retail sales in the US in two limited periods of time that are different in terms of macroeconomic conditions. Two additional goals of the research were to identify the best of the two proposed methods and compare the performance of the models on an extended period of 50 years.

To answer the main research question, it can be stated that the LSTM and ARIMA-LSTM methods bring only partial significant improvements in forecasting accuracy. The LSTM has unsatisfactory performance in the first period which is also the shortest and most volatile, but it improves in the second period. Even though differences in errors with the best models were approximately 1-2%, such differences can still have great practical implications. For example in the first period, the small difference of 1.59% between the ANN and LSTM means an additional 20 billion US dollars in incorrect predictions over the prediction year. Such numbers only tend to increase over time as the aggregate sales also increase.

The forecasting accuracy of the ARIMA-LSTM method is better than that of the LSTM in both periods, but this can be explained by the fact that it is based on the forecasts of ARIMA. Hence when comparing the two methods, it can be seen that ARIMA-LSTM only manages to improve the forecasting accuracy of ARIMA in the multi-step ahead forecasts of the second period, but on average it is nearly the same as ARIMA. Overall, using the second step with the LSTM depends on how justified it is. LSTM has a considerably exhaustive model selection in form of grid search or trial and error methods, and a training process which can be very computationally difficult, both depending on the input data. Hence there needs to be a justified trade-off between computational efficiency and forecast improvement, which could not be observed in this research.

Regarding the results of the extended period of time, it can be said that the econometric methods ARIMA and Winters model only marginally outperform the LSTM techniques. Both results of the errors and of the tests point to LSTM and ARIMA-LSTM being two models which perform the best on longer periods. This might suggest some findings in relation to the data and the model architectures themselves. It seems that the LSTM performs better on larger data sets with less clear non-linear patterns. Similar to the ARIMA and Winters models, they seem to incorporate the trend and seasonal patterns efficiently. The ANN on the other, has an overall unsatisfactory performance in the extended period, but this might be explained by its simple architecture and input features without lags.

All in all, the results point to the fact that econometric models are efficient, robust and accurate models for forecasting time series while models in the field of Machine Learning also provide evidence of good performance but only in certain scenarios as pointed out by Alon et al. (2001). This is also largely in line with the literature that discusses the absence of a consistent conclusion on the performance of Machine Learning techniques. Aggregate retail sales like most economic variables do not incorporate many non-linear aspects or extreme volatility as time series in financial markets or natural sciences. Hence, it is reasonable to conclude that the choice of forecasting method should be based mostly on the data that is studied. Economic data can be tackled efficiently by simple models that are based on incorporating linear patterns, while more complex processes can be analyzed by more advanced methods such as Artificial Neural Networks and Long Short-Term Memory methods.

There are some limitations of this study that have to be taken into account when considering the results. Due to the relatively small data sets, it was difficult to make a test sample from a part of the training sample without losing valuable information from the data. Hence all models have been selected by testing them on the forecasting window, which makes the models biased towards those periods of time. The subsequent loss of robustness of the models can be caused by this fact. Moreover, given the limited computational capacity it was not possible to test more variations of the LSTM model. During trials, it was noticed that the training times and computing requirements were highly affected by bigger number of epochs and hidden nodes. It is not excluded that a certain untested combination of hyperparameters could yield better forecasting results.

Furthermore, there are some suggestions that future research of this topic can consider based on the findings of this paper. First, building more robust and consistent models using the LSTM method can be studied by using different data sets or simulated data. Moreover, different preprocessing techniques can be studied. Deseasonalition or data transformations that amplify certain patterns can improve the forecast accuracy of the LSTM method. Finally, more focus can be set on the LSTM architecture itself. Given that many variations of the model such as Gated Recurrent Unit (GRU) exist, it might be interesting to study the performance of these types as well besides the classic LSTM. Trying out different variations also means that different combinations of hyperparameters become possible. Hence, the large number of possibilities makes this field worth exploring and open to different interpretations.

As stated in the introduction, this study was created with the intention of expanding the methodology of forecasting retail sales and economic time series in general. The overview of the methods and the comparison between them will hopefully inspire future researchers in developing new methods or finding the most suitable one. Furthermore, the found insights can be helpful for policymakers and retail businesses with the aim of having accurate forecasts of sales in the short and long-run. The LSTM model like many other Machine Learning techniques is widely used and applied, but until there is a clear theoretical framework, the model selection depends on the choices of the researcher. Hence, attention needs to be set primarily on the scenario that the models are applied to and whether the improvements given the implementation difficulty can be justified.

## References

Adhikari, R., & Agrawal, R. (2012). Forecasting strong seasonal time series with artificial neural networks.

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5-6), 594–621.

Alon, I., Qi, M., & Sadowski, R. J. (2001). Forecasting aggregate retail sales:: a comparison of artificial neural networks and traditional methods. *Journal of Retailing and Consumer Services*, *8*(3), 147–156.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control.* John Wiley & Sons.

Breuel, T. M. (2015). Benchmarking of LSTM networks. *arXiv preprint arXiv:1508.02774*.

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE international conference on big data (big data)* (pp. 2823–2824).

Choi, H. K. (2018). Stock price correlation coefficient prediction with ARIMA-LSTM hybrid model. *arXiv preprint arXiv:1808.01560*.

Daliakopoulos, I. N., Coulibaly, P., & Tsanis, I. K. (2005). Groundwater level forecasting using artificial neural networks. *Journal of Hydrology*, *309*(1-4), 229–240.

Elmasdotter, A., & Nyströmer, C. (2018). *A comparative study between LSTM and ARIMA for sales forecasting in retail.*

Feng, X., Li, Q., Zhu, Y., Hou, J., Jin, L., & Wang, J. (2015). Artificial neural networks forecasting of pm2. 5 pollution using air mass trajectory based geographic model and wavelet transformation. *Atmospheric Environment*, *107*, 118–128.

Fildes, R., Ma, S., & Kolassa, S. (2019). Retail forecasting: Research and practice. *International Journal of Forecasting*.

Fisher, M., & Raman, A. (2018). Using data and big data in retailing. *Production and Operations Management*, *27*(9), 1665–1669.

Franses, P. H., et al. (1998). *Time series models for business and economic forecasting.* Cambridge university press.

FRED. (2020, Apr). *Value Added by Private Industries: Retail Trade as a Percentage of GDP.* FRED. Retrieved from `https://fred.stlouisfed.org/series/VAPGDPR`

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222–2232.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*(8), 10389–10397.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hsieh, T.-J., Hsiao, H.-F., & Yeh, W.-C. (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, *11*(2), 2510–2525.

Jain, A., & Kumar, A. M. (2007). Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, *7*(2), 585–592.

Kermanshahi, B. (1998). Recurrent neural network for forecasting next 10 years loads of nine Japanese utilities. *Neurocomputing*, *23*(1-3), 125–133.

Khashei, M., & Bijari, M. (2010). An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications*, *37*(1), 479–489.

Khashei, M., & Bijari, M. (2011). A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing*, *11*(2), 2664–2675.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mandal, S., & Prabaharan, N. (2006). Ocean wave forecasting using recurrent neural networks. *Ocean engineering*, *33*(10), 1401–1410.

More, A., & Deo, M. (2003). Forecasting wind with neural networks. *Marine structures*, *16*(1), 35–49.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419–1426).

Nguyen, D., & Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *1990 IJCNN International Joint Conference on Neural Networks* (pp. 21–26).

Olah, C. (2015). *Understanding lstm networks.* Retrieved from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Pai, P.-F., & Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, *33*(6), 497–505.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ruder, S. (2016). *An overview of gradient descent optimization algorithms.*

Siami-Namini, S., & Namin, A. S. (2018). Forecasting economics and financial time series: ARIMA vs. LSTM. *arXiv preprint arXiv:1803.06386*.

Tkáč, M., & Verner, R. (2016). Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, *38*, 788–804.

Tkacz, G. (2001). Neural network forecasting of Canadian GDP growth. *International Journal of Forecasting*, *17*(1), 57–69.

Tong, T., Shah, M., Cherukumalli, M., & Moulehiawy, Y. (2018). Investigating Long Short-Term Memory Neural Networks for Financial Time-Series Prediction. *Available at SSRN 3175336*.

US Census Bureau. (2020, May). *US Census Bureau Retail Trade.* Author. Retrieved 5/05/2020, from `https://www.census.gov/retail/historic_releases.html`

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: https://doi.org/10.1038/s41592-019-0686-2

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science*, *6*(3), 324–342.

Yu, Q., Wang, K., Strandhagen, J. O., & Wang, Y. (2017). Application of long short-term memory neural network to sales forecasting in retail—a case study. In *International Workshop of Advanced Manufacturing and Automation* (pp. 11–17).

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1), 35–62.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, *160*(2), 501–514.

## A Algorithms for Machine Learning methods

**Algorithm 1:** Bayesian Regularization with Levenberg Marquadt optimization

```
Result: Train neural network efficiently with generalized results
Initialize weights WB, JJ, JE, ii;
Initialize N = numObservations, $\gamma$ = numParameters;
if $error = 0$ then $\beta$=1;
else $\beta = (N - \gamma)/(2 * error)$;
$\alpha$=$\gamma$/(2 * WB * WB);
perf = $\beta$ * $error$ + $\alpha$ * WB * WB;
While stoppingcriterion = false:
while True do
    if $\mu > \mu_{max}$ then
        break;
    end
    Compute derivative dX=-($\beta$ * JJ + ii * $(\mu + \alpha)$)/($\beta$ * JE + $\alpha$ * WB);
    WB2 = WB + dX;
    Compute error2;
    perf2 = $\beta$ * $error2$ + $\alpha$ * WB2 * WB2;
    if perf2 < perf then
        Update WB = WB2, perf = perf2;
        $\mu$=$\mu$ * $\mu_{dec}$;
        if $\mu$ <1e-20 then
            $\mu$=1e-20
        end
    end
    $\mu = \mu$ * $\mu_{inc}$;
end
Update $error$, JE, JJ;
if $\mu < \mu_{max}$ then
    $\gamma$ = N - $\alpha$ * trace(inv($\beta$ * JJ + ii * $\alpha$));
end
```

**Algorithm 2:** Continuation

```
if $WB * WB = 0$ then
    $\alpha = 1$;
else $\alpha = \gamma/(2 * WB * WB)$
if $error = 0$ then
    $\beta$ =1;
else $\beta = (N - \gamma)/(2 * error)$;
$perf = \beta * error + \alpha * WB * WB$;
```

**Algorithm 3:** ANN implementation - One-step ahead

```
Result: Obtain forecasts of next 12 months
create X, Y, YTest variables;
forecasts = empty vector of size 1 x 12;
for t from 1 upto iterations do
    for p from 0 upto 11 do
        Update XTrain = X(1 to (length(trainingset) + p));
        Update YTrain = Y(1 to (length(trainingset) + p));
        Update XTest = X(length(trainingset) + p + 1);
        select ANN architecture;
        Train network using XTrain, YTrain;
        forecasts(p + 1) = forecasts(p + 1) + Predict(network, XTest);
    end
end
forecasts = forecasts/iterations;
Compute MAPE(forecasts, YTest);
```

**Algorithm 4:** ANN implementation - Multi-step ahead

```
Result: Obtain forecasts of next 12 months
create XTrain, YTrain, XTest, YTest variables;
forecasts = empty vector of size 1 x 12;
for t from 1 upto iterations do
    select ANN architecture;
    Train network using XTrain, YTrain;
    forecasts = forecasts + Predict(network, XTest);
end
forecasts = forecasts/iterations;
Compute MAPE(forecasts, YTest);
```

**Algorithm 5:** LSTM implementation - One-step ahead

```
Result: Obtain forecasts of next 12 months
standardize data;
create lagged variables;
create X, Y, YTest variables;
select architecture of the LSTM model;
forecasts = empty vector of size 1 x 12;
for p from 0 upto 11 do
    Update XTrain = X(1 to (length(trainingset) + p));
    Update YTrain = Y(1 to (length(trainingset) + p));
    Update XTest = X((length(trainingset) + p + 1) to end);
    for t from 1 upto iterations do
        Train network using XTrain, YTrain;
        YPred = Predict(network, concatenated(XTrain, XTest));
        forecast = forecast + YPred(length(YTrain) + 1);
    end
    forecasts(p + 1) = forecast/iterations;
end
Destandardize forecasts;
Compute MAPE(forecasts, YTest);
```

**Algorithm 6:** LSTM implementation - Multi-step ahead

```
Result: Obtain forecasts of next 12 months
standardize data;
create lagged variables;
create X, Y, XTrain, YTrain, XTest, YTest variables;
select architecture of the LSTM model;
forecasts = empty vector of size 1 x 12;
for t from 1 upto iterations do
    Train network using XTrain, YTrain;
    for p from 0 upto 11 do
        if p>0 then
            Update XTest using forecasts;
        end
        YPred = Predict(network, concatenated(XTrain,XTest));
        forecasts(p + 1) = forecasts(p + 1) + YPred(length(trainingset) + p + 1);
    end
end
forecasts = forecasts/iterations;
Destandardize forecasts;
Compute MAPE(forecasts, YTest);
```

## B Grid search results

| Results for first period for LSTM | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes:10 | epochs:90 | lr:0.005 | MAPE:3.0225 | Nodes:70 | epochs:200 | lr:0.005 | MAPE:3.1033 |
| Nodes:10 | epochs:120 | lr:0.005 | MAPE:3.1925 | Nodes:90 | epochs:70 | lr:0.005 | MAPE:2.9083 |
| Nodes:10 | epochs:150 | lr:0.005 | MAPE:3.2512 | Nodes:90 | epochs:90 | lr:0.005 | MAPE:2.9919 |
| Nodes:10 | epochs:200 | lr:0.005 | MAPE:3.2347 | Nodes:90 | epochs:120 | lr:0.005 | MAPE:2.8917 |
| Nodes:20 | epochs:70 | lr:0.005 | MAPE:2.8534 | Nodes:90 | epochs:150 | lr:0.005 | MAPE:2.9002 |
| Nodes:20 | epochs:90 | lr:0.005 | MAPE:3.0342 | Nodes:90 | epochs:200 | lr:0.005 | MAPE:3.0701 |
| Nodes:20 | epochs:120 | lr:0.005 | MAPE:2.968 | Nodes:120 | epochs:70 | lr:0.005 | MAPE:3.0219 |
| Nodes:20 | epochs:150 | lr:0.005 | MAPE:2.9977 | Nodes:120 | epochs:90 | lr:0.005 | MAPE:2.9371 |
| Nodes:20 | epochs:200 | lr:0.005 | MAPE:3.191 | Nodes:120 | epochs:120 | lr:0.005 | MAPE:3.1018 |
| Nodes:30 | epochs:70 | lr:0.005 | MAPE:3.2926 | Nodes:120 | epochs:150 | lr:0.005 | MAPE:2.9476 |
| **Nodes:30** | **epochs:90** | **lr:0.005** | **MAPE:2.7564** | Nodes:120 | epochs:200 | lr:0.005 | MAPE:3.1496 |
| Nodes:30 | epochs:120 | lr:0.005 | MAPE:2.8038 | Nodes:150 | epochs:70 | lr:0.005 | MAPE:3.228 |
| Nodes:30 | epochs:150 | lr:0.005 | MAPE:2.889 | Nodes:150 | epochs:90 | lr:0.005 | MAPE:2.8666 |
| Nodes:30 | epochs:200 | lr:0.005 | MAPE:3.1005 | Nodes:150 | epochs:120 | lr:0.005 | MAPE:2.8918 |
| Nodes:50 | epochs:70 | lr:0.005 | MAPE:2.8407 | Nodes:150 | epochs:150 | lr:0.005 | MAPE:2.9939 |
| Nodes:50 | epochs:90 | lr:0.005 | MAPE:2.8523 | Nodes:150 | epochs:200 | lr:0.005 | MAPE:3.1197 |
| Nodes:50 | epochs:120 | lr:0.005 | MAPE:2.8811 | Nodes:200 | epochs:70 | lr:0.005 | MAPE:3.1242 |
| Nodes:50 | epochs:150 | lr:0.005 | MAPE:2.9039 | Nodes:200 | epochs:90 | lr:0.005 | MAPE:3.0316 |
| Nodes:50 | epochs:200 | lr:0.005 | MAPE:3.064 | Nodes:200 | epochs:120 | lr:0.005 | MAPE:3.2158 |
| Nodes:70 | epochs:70 | lr:0.005 | MAPE:2.9669 | Nodes:200 | epochs:150 | lr:0.005 | MAPE:3.2967 |
| Nodes:70 | epochs:90 | lr:0.005 | MAPE:2.8858 | Nodes:200 | epochs:200 | lr:0.005 | MAPE:3.1443 |
| Nodes:70 | epochs:120 | lr:0.005 | MAPE:2.8742 | | | | |

Results for second period for LSTM

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes:10 | epochs:70 | lr:0.005 | MAPE:7.5664 | Nodes:70 | epochs:150 | lr:0.005 | MAPE:3.5238 |
| Nodes:10 | epochs:90 | lr:0.005 | MAPE:4.3672 | Nodes:70 | epochs:200 | lr:0.005 | MAPE:3.5042 |
| Nodes:10 | epochs:120 | lr:0.005 | MAPE:4.0522 | Nodes:90 | epochs:70 | lr:0.005 | MAPE:3.556 |
| Nodes:10 | epochs:150 | lr:0.005 | MAPE:4.223 | Nodes:90 | epochs:90 | lr:0.005 | MAPE:3.6993 |
| Nodes:10 | epochs:200 | lr:0.005 | MAPE:4.3418 | Nodes:90 | epochs:120 | lr:0.005 | MAPE:3.5786 |
| Nodes:20 | epochs:70 | lr:0.005 | MAPE:3.8562 | Nodes:90 | epochs:150 | lr:0.005 | MAPE:3.3207 |
| Nodes:20 | epochs:90 | lr:0.005 | MAPE:3.4041 | Nodes:90 | epochs:200 | lr:0.005 | MAPE:3.5524 |
| Nodes:20 | epochs:120 | lr:0.005 | MAPE:3.2946 | Nodes:120 | epochs:70 | lr:0.005 | MAPE:3.5539 |
| Nodes:20 | epochs:150 | lr:0.005 | MAPE:3.3189 | Nodes:120 | epochs:90 | lr:0.005 | MAPE:3.6601 |
| Nodes:20 | epochs:200 | lr:0.005 | MAPE:3.5408 | Nodes:120 | epochs:120 | lr:0.005 | MAPE:3.569 |
| Nodes:30 | epochs:70 | lr:0.005 | MAPE:3.4711 | Nodes:120 | epochs:150 | lr:0.005 | MAPE:3.6083 |
| **Nodes:30** | **epochs:90** | **lr:0.005** | **MAPE:3.1707** | Nodes:120 | epochs:200 | lr:0.005 | MAPE:3.4188 |
| Nodes:30 | epochs:120 | lr:0.005 | MAPE:3.2091 | Nodes:150 | epochs:70 | lr:0.005 | MAPE:4.0122 |
| Nodes:30 | epochs:150 | lr:0.005 | MAPE:3.4064 | Nodes:150 | epochs:90 | lr:0.005 | MAPE:4.0291 |
| Nodes:30 | epochs:200 | lr:0.005 | MAPE:3.585 | Nodes:150 | epochs:120 | lr:0.005 | MAPE:3.9205 |
| Nodes:50 | epochs:70 | lr:0.005 | MAPE:3.2428 | Nodes:150 | epochs:150 | lr:0.005 | MAPE:3.7386 |
| Nodes:50 | epochs:90 | lr:0.005 | MAPE:3.3595 | Nodes:150 | epochs:200 | lr:0.005 | MAPE:3.2702 |
| Nodes:50 | epochs:120 | lr:0.005 | MAPE:3.6606 | Nodes:200 | epochs:70 | lr:0.005 | MAPE:4.3909 |
| Nodes:50 | epochs:150 | lr:0.005 | MAPE:3.781 | Nodes:200 | epochs:90 | lr:0.005 | MAPE:4.4493 |
| Nodes:50 | epochs:200 | lr:0.005 | MAPE:3.5708 | Nodes:200 | epochs:120 | lr:0.005 | MAPE:4.1299 |
| Nodes:70 | epochs:70 | lr:0.005 | MAPE:3.3471 | Nodes:200 | epochs:150 | lr:0.005 | MAPE:4.3491 |
| Nodes:70 | epochs:90 | lr:0.005 | MAPE:3.6525 | Nodes:200 | epochs:200 | lr:0.005 | MAPE:4.2861 |
| Nodes:70 | epochs:120 | lr:0.005 | MAPE:3.4751 | | | | |

| Results for third period for LSTM | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes:10 | epochs:70 | lr:0.005 | MAPE:13.0638 | Nodes:70 | epochs:150 | lr:0.005 | MAPE:6.8869 |
| Nodes:10 | epochs:90 | lr:0.005 | MAPE:10.5843 | Nodes:70 | epochs:200 | lr:0.005 | MAPE:7.1865 |
| Nodes:10 | epochs:120 | lr:0.005 | MAPE:12.2822 | Nodes:90 | epochs:70 | lr:0.005 | MAPE:6.92 |
| Nodes:10 | epochs:150 | lr:0.005 | MAPE:10.2581 | Nodes:90 | epochs:90 | lr:0.005 | MAPE:5.972 |
| Nodes:10 | epochs:200 | lr:0.005 | MAPE:7.8569 | Nodes:90 | epochs:120 | lr:0.005 | MAPE:6.8478 |
| Nodes:20 | epochs:70 | lr:0.005 | MAPE:6.8477 | Nodes:90 | epochs:150 | lr:0.005 | MAPE:7.7013 |
| Nodes:20 | epochs:90 | lr:0.005 | MAPE:5.8401 | Nodes:90 | epochs:200 | lr:0.005 | MAPE:7.5976 |
| **Nodes:20** | **epochs:120** | **lr:0.005** | **MAPE:4.4406** | Nodes:120 | epochs:70 | lr:0.005 | MAPE:6.1938 |
| Nodes:20 | epochs:150 | lr:0.005 | MAPE:4.8645 | Nodes:120 | epochs:90 | lr:0.005 | MAPE:7.5656 |
| Nodes:20 | epochs:200 | lr:0.005 | MAPE:5.6452 | Nodes:120 | epochs:120 | lr:0.005 | MAPE:6.8945 |
| Nodes:30 | epochs:70 | lr:0.005 | MAPE:5.2488 | Nodes:120 | epochs:150 | lr:0.005 | MAPE:7.1839 |
| **Nodes:30** | **epochs:90** | **lr:0.005** | **MAPE:4.7355** | Nodes:120 | epochs:200 | lr:0.005 | MAPE:6.0006 |
| Nodes:30 | epochs:120 | lr:0.005 | MAPE:6.1051 | Nodes:150 | epochs:70 | lr:0.005 | MAPE:5.4432 |
| Nodes:30 | epochs:150 | lr:0.005 | MAPE:5.2965 | Nodes:150 | epochs:90 | lr:0.005 | MAPE:4.8576 |
| Nodes:30 | epochs:200 | lr:0.005 | MAPE:5.7807 | Nodes:150 | epochs:120 | lr:0.005 | MAPE:6.7819 |
| Nodes:50 | epochs:70 | lr:0.005 | MAPE:6.0295 | Nodes:150 | epochs:150 | lr:0.005 | MAPE:7.1331 |
| Nodes:50 | epochs:90 | lr:0.005 | MAPE:6.452 | Nodes:150 | epochs:200 | lr:0.005 | MAPE:5.5175 |
| Nodes:50 | epochs:120 | lr:0.005 | MAPE:6.7448 | Nodes:200 | epochs:70 | lr:0.005 | MAPE:5.8016 |
| Nodes:50 | epochs:150 | lr:0.005 | MAPE:7.2192 | Nodes:200 | epochs:90 | lr:0.005 | MAPE:6.6565 |
| Nodes:50 | epochs:200 | lr:0.005 | MAPE:6.0644 | Nodes:200 | epochs:120 | lr:0.005 | MAPE:7.214 |
| Nodes:70 | epochs:70 | lr:0.005 | MAPE:6.2617 | Nodes:200 | epochs:150 | lr:0.005 | MAPE:6.6908 |
| Nodes:70 | epochs:90 | lr:0.005 | MAPE:5.7472 | Nodes:200 | epochs:200 | lr:0.005 | MAPE:6.2051 |
| Nodes:70 | epochs:120 | lr:0.005 | MAPE:6.6726 | | | | |

| Results for first period for ARIMA-LSTM | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Nodes:10 | epochs:70 | lr | 0.001 | MAPE:1.7447 | Nodes:70 | epochs:150 | lr | 0.001 | MAPE:1.7223 |
| Nodes:10 | epochs:90 | lr | 0.001 | MAPE:1.755 | Nodes:70 | epochs:200 | lr | 0.001 | MAPE:1.7455 |
| Nodes:10 | epochs:120 | lr | 0.001 | MAPE:1.7845 | Nodes:90 | epochs:70 | lr | 0.001 | MAPE:1.7148 |
| Nodes:10 | epochs:150 | lr | 0.001 | MAPE:1.813 | Nodes:90 | epochs:90 | lr | 0.001 | MAPE:1.7045 |
| Nodes:10 | epochs:200 | lr | 0.001 | MAPE:1.8559 | Nodes:90 | epochs:120 | lr | 0.001 | MAPE:1.7046 |
| Nodes:20 | epochs:70 | lr | 0.001 | MAPE:1.7314 | Nodes:90 | epochs:150 | lr | 0.001 | MAPE:1.7168 |
| Nodes:20 | epochs:90 | lr | 0.001 | MAPE:1.7506 | Nodes:90 | epochs:200 | lr | 0.001 | MAPE:1.7194 |
| Nodes:20 | epochs:120 | lr | 0.001 | MAPE:1.7706 | Nodes:120 | epochs:70 | lr | 0.001 | MAPE:1.7051 |
| Nodes:20 | epochs:150 | lr | 0.001 | MAPE:1.7913 | Nodes:120 | epochs:90 | lr | 0.001 | MAPE:1.7022 |
| Nodes:20 | epochs:200 | lr | 0.001 | MAPE:1.8012 | Nodes:120 | epochs:120 | lr | 0.001 | MAPE:1.7012 |
| Nodes:30 | epochs:70 | lr | 0.001 | MAPE:1.7308 | Nodes:120 | epochs:150 | lr | 0.001 | MAPE:1.7171 |
| **Nodes:30** | **epochs:90** | **lr** | **0.001** | **MAPE:1.736** | Nodes:120 | epochs:200 | lr | 0.001 | MAPE:1.6611 |
| Nodes:30 | epochs:120 | lr | 0.001 | MAPE:1.7614 | Nodes:150 | epochs:70 | lr | 0.001 | MAPE:1.6932 |
| Nodes:30 | epochs:150 | lr | 0.001 | MAPE:1.7615 | Nodes:150 | epochs:90 | lr | 0.001 | MAPE:1.6817 |
| Nodes:30 | epochs:200 | lr | 0.001 | MAPE:1.7824 | Nodes:150 | epochs:120 | lr | 0.001 | MAPE:1.6827 |
| Nodes:50 | epochs:70 | lr | 0.001 | MAPE:1.7225 | Nodes:150 | epochs:150 | lr | 0.001 | MAPE:1.6543 |
| Nodes:50 | epochs:90 | lr | 0.001 | MAPE:1.7309 | Nodes:150 | epochs:200 | lr | 0.001 | MAPE:1.6378 |
| Nodes:50 | epochs:120 | lr | 0.001 | MAPE:1.7298 | Nodes:200 | epochs:70 | lr | 0.001 | MAPE:1.6761 |
| Nodes:50 | epochs:150 | lr | 0.001 | MAPE:1.7459 | Nodes:200 | epochs:90 | lr | 0.001 | MAPE:1.6736 |
| Nodes:50 | epochs:200 | lr | 0.001 | MAPE:1.78 | Nodes:200 | epochs:120 | lr | 0.001 | MAPE:1.6625 |
| Nodes:70 | epochs:70 | lr | 0.001 | MAPE:1.7174 | Nodes:200 | epochs:150 | lr | 0.001 | MAPE:1.6623 |
| Nodes:70 | epochs:90 | lr | 0.001 | MAPE:1.7202 | Nodes:200 | epochs:200 | lr | 0.001 | MAPE:1.7448 |

Results for second period for ARIMA-LSTM

| | | | | |
|---|---|---|---|---|
| Nodes:10 | epochs:70 | lr | 0.001 | MAPE:1.6793 |
| Nodes:10 | epochs:90 | lr | 0.001 | MAPE:1.6892 |
| Nodes:10 | epochs:120 | lr | 0.001 | MAPE:1.6897 |
| Nodes:10 | epochs:150 | lr | 0.001 | MAPE:1.7005 |
| Nodes:10 | epochs:200 | lr | 0.001 | MAPE:1.6988 |
| Nodes:20 | epochs:70 | lr | 0.001 | MAPE:1.7097 |
| Nodes:20 | epochs:90 | lr | 0.001 | MAPE:1.7075 |
| Nodes:20 | epochs:120 | lr | 0.001 | MAPE:1.7127 |
| Nodes:20 | epochs:150 | lr | 0.001 | MAPE:1.7214 |
| Nodes:20 | epochs:200 | lr | 0.001 | MAPE:1.718 |
| Nodes:30 | epochs:70 | lr | 0.001 | MAPE:1.717 |
| Nodes:30 | **epochs:90** | **lr** | **0.001** | **MAPE:1.7194** |
| Nodes:30 | epochs:120 | lr | 0.001 | MAPE:1.7201 |
| Nodes:30 | epochs:150 | lr | 0.001 | MAPE:1.7228 |
| Nodes:30 | epochs:200 | lr | 0.001 | MAPE:1.7327 |
| Nodes:50 | epochs:70 | lr | 0.001 | MAPE:1.7126 |
| Nodes:50 | epochs:90 | lr | 0.001 | MAPE:1.7192 |
| Nodes:50 | epochs:120 | lr | 0.001 | MAPE:1.721 |
| Nodes:50 | epochs:150 | lr | 0.001 | MAPE:1.7387 |
| Nodes:50 | epochs:200 | lr | 0.001 | MAPE:1.7751 |
| Nodes:70 | epochs:70 | lr | 0.001 | MAPE:1.7208 |
| Nodes:70 | epochs:90 | lr | 0.001 | MAPE:1.7272 |
| Nodes:70 | epochs:120 | lr | 0.001 | MAPE:1.7447 |
| Nodes:70 | epochs:150 | lr | 0.001 | MAPE:1.7862 |
| Nodes:70 | epochs:200 | lr | 0.001 | MAPE:1.8609 |
| Nodes:90 | epochs:70 | lr | 0.001 | MAPE:1.7268 |
| Nodes:90 | epochs:90 | lr | 0.001 | MAPE:1.7435 |
| Nodes:90 | epochs:120 | lr | 0.001 | MAPE:1.7844 |
| Nodes:90 | epochs:150 | lr | 0.001 | MAPE:1.8389 |
| Nodes:90 | epochs:200 | lr | 0.001 | MAPE:2.0729 |
| Nodes:120 | epochs:70 | lr | 0.001 | MAPE:1.7387 |
| Nodes:120 | epochs:90 | lr | 0.001 | MAPE:1.7738 |
| Nodes:120 | epochs:120 | lr | 0.001 | MAPE:1.8213 |
| Nodes:120 | epochs:150 | lr | 0.001 | MAPE:2.0783 |
| Nodes:120 | epochs:200 | lr | 0.001 | MAPE:2.2418 |
| Nodes:150 | epochs:70 | lr | 0.001 | MAPE:1.7739 |
| Nodes:150 | epochs:90 | lr | 0.001 | MAPE:1.8653 |
| Nodes:150 | epochs:120 | lr | 0.001 | MAPE:2.108 |
| Nodes:150 | epochs:150 | lr | 0.001 | MAPE:2.4894 |
| Nodes:150 | epochs:200 | lr | 0.001 | MAPE:2.7955 |
| Nodes:200 | epochs:70 | lr | 0.001 | MAPE:1.82 |
| Nodes:200 | epochs:90 | lr | 0.001 | MAPE:1.9641 |
| Nodes:200 | epochs:120 | lr | 0.001 | MAPE:2.3689 |
| Nodes:200 | epochs:150 | lr | 0.001 | MAPE:2.7987 |
| Nodes:200 | epochs:200 | lr | 0.001 | MAPE:3.0712 |

| Results for third period for ARIMA-LSTM | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes:10 | epochs:70 | lr:0.001 | MAPE:1.8823 | Nodes:70 | epochs:70 | lr:0.001 | MAPE:1.8653 |
| Nodes:10 | epochs:90 | lr:0.001 | MAPE:1.8799 | Nodes:70 | epochs:90 | lr:0.001 | MAPE:1.8596 |
| Nodes:10 | epochs:120 | lr:0.001 | MAPE:1.8739 | Nodes:70 | epochs:120 | lr:0.001 | MAPE:1.8314 |
| Nodes:10 | epochs:150 | lr:0.001 | MAPE:1.8587 | Nodes:70 | epochs:150 | lr:0.001 | MAPE:1.8138 |
| Nodes:10 | epochs:200 | lr:0.001 | MAPE:1.8077 | Nodes:70 | epochs:200 | lr:0.001 | MAPE:1.7768 |
| Nodes:20 | epochs:70 | lr:0.001 | MAPE:1.8789 | Nodes:90 | epochs:70 | lr:0.001 | MAPE:1.8669 |
| Nodes:20 | epochs:90 | lr:0.001 | MAPE:1.8697 | Nodes:90 | epochs:90 | lr:0.001 | MAPE:1.859 |
| Nodes:20 | epochs:120 | lr:0.001 | MAPE:1.8589 | Nodes:90 | epochs:120 | lr:0.001 | MAPE:1.8349 |
| Nodes:20 | epochs:150 | lr:0.001 | MAPE:1.8249 | Nodes:90 | epochs:150 | lr:0.001 | MAPE:1.817 |
| Nodes:20 | epochs:200 | lr:0.001 | MAPE:1.7633 | Nodes:90 | epochs:200 | lr:0.001 | MAPE:1.7732 |
| Nodes:30 | epochs:70 | lr:0.001 | MAPE:1.8718 | Nodes:120 | epochs:70 | lr:0.001 | MAPE:1.8683 |
| **Nodes:30** | **epochs:90** | **lr:0.001** | **MAPE:1.8612** | Nodes:120 | epochs:90 | lr:0.001 | MAPE:1.8576 |
| Nodes:30 | epochs:120 | lr:0.001 | MAPE:1.8484 | Nodes:120 | epochs:120 | lr:0.001 | MAPE:1.8266 |
| Nodes:30 | epochs:150 | lr:0.001 | MAPE:1.8197 | Nodes:120 | epochs:150 | lr:0.001 | MAPE:1.786 |
| Nodes:30 | epochs:200 | lr:0.001 | MAPE:1.7772 | Nodes:120 | epochs:200 | lr:0.001 | MAPE:1.7346 |
| Nodes:50 | epochs:70 | lr:0.001 | MAPE:1.8637 | Nodes:150 | epochs:70 | lr:0.001 | MAPE:1.8686 |
| Nodes:50 | epochs:90 | lr:0.001 | MAPE:1.8637 | Nodes:150 | epochs:90 | lr:0.001 | MAPE:1.8452 |
| Nodes:50 | epochs:120 | lr:0.001 | MAPE:1.8289 | Nodes:150 | epochs:120 | lr:0.001 | MAPE:1.8045 |
| Nodes:50 | epochs:150 | lr:0.001 | MAPE:1.8215 | Nodes:150 | epochs:150 | lr:0.001 | MAPE:1.7597 |
| Nodes:50 | epochs:200 | lr:0.001 | MAPE:1.802 | Nodes:150 | epochs:200 | lr:0.001 | MAPE:1.7108 |