Optimized Grid Search Approach to Select and Calibrate Exponential Smoothing, SARIMA and LSTM Models for Demand Forecasting.

Rafael Motta Rapp (Escola Politécnica, Universidade de São Paulo)

Prof. Dr. Daniel de Oliveira Mota (Escola Politécnica, Universidade de São Paulo)

Forecasting accuracy allows the possibility to better allocate and plan resources improving sales and operations planning (S&OP). Our approach defines an optimized process to select and calibrate models for time series data applied in a consumer electronics company, minimizing the need of expert interventions. We highlight the impact of choosing the properties that govern the behavior of the models (Hyperparameters) and propose a process for model selection between Exponential Smoothing and SARIMA forecasting methods. We expand this approach to deep learning LSTM model. Also, the methods for each model are reviewed based on multi-criteria performance metrics. The results show the advantages of the proposed process for model selection and the reduced complexity and implementation for such models in a business environment.

Keywords: Time Series Forecasting, Demand Planning, Hyperparameters, Grid Search, Exponential Smoothing, ARIMA, LSTM, Forecasting Accuracy.

1. Introduction

Traditionally, demand forecast has served as one important foundation for production planning and supply chain. It interferes directly in activities such as sourcing, manufacturing, distribution. Besides being a technique widely used its accuracy remains a challenge for many companies due to uncertainty present in customer behavior either in B2C and B2B environment (Syntetos et al. 2016; Abolghasemi et al., 2020). Demand forecast predicts two vital pieces of information for a business: (1) the expected demand in a given period, and (2) how much uncertainty is associated with a prediction.

There are several quantitative models available in the literature to address forecasting problems. They range from simple (and very useful) moving average, passing by exponential smoothing (considering or not trends and/or seasonality), autoregressive moving average models (Gooijer; Hyndman 2006), until moderns machine learning such as neural networks. (Tealab, 2018). One question remains: what is the best technique to have predictions that are more accurate and feasible to maintain?

Forecasting is often modelled in the context of time series data, which consists of values observed at discrete points in time. The available methods today for modelling time series forecasting provide increasing complexity if one wants to extract all the potential from the most recent research (Parmezan; Souza; Batista, 2019). In consequence, it is harder for business to complete explore and choose which forecasting model to apply and adopt a process to maintain. In the era of big data (Boone et al. 2019) powered by computing processing, there has been a shift in applying not only traditional forecasting techniques but also machine learning algorithms such as Neural Networks (NN). Neural Networks, motivated by their data driven approximation of any linear or nonlinear function, contributed to the largest number of publications of any area in operational research according to Fildes et al. (2008) and they formed one of the top four areas of growth in forecasting journals in 2006 according to Crone, Hibon and Nikolopoulos (2011).

Looking to quantify how better NN models perform in forecasting is a discussion introduced by Crone, Hibon and Nikolopoulos (2011) and later in Makridakis, Spiliotis and Assimakopouloset (2018), through a series of competitions known as Makridakis Competitions. The Makridakis (M) Competitions are a series of open competitions to evaluate and compare the accuracy of different forecasting models including NN. Professor Spyros Makridakis from University of Nicosia (UNIC) started these competitions to evaluate the performance of existing and new forecasting models (Makridakis; Hyndman; Petropoulos, 2020). There have been six

M Competitions since 1982 (M6 started in 2022), each differing in the number of time series used, the forecasting methods experimented with and other features regarding the structure of data.

In those competitions as well in the context of this work, traditional forecasting models are referred to exponential smoothing (ES) methods and autoregressive moving average (ARIMA) models. These models have a pre-defined functional form with parameters that govern the predictions. For this reason, they are also called parametric models. Machine learning methods do not have a pre-defined functional form as this will change according to different learning features specific for each type of algorithm. Because of this they are referred to as non-parametric. How well these machine learning models perform compared against traditional models is what these competitions explore.

Although the results from the M4 forecasting competition have clearly shown the potential of NNs (Makridakis; Spiliotis; Assimakopoulos, 2020), ES and ARIMA have traditionally supported forecasting in a univariate context not only from their accuracy but also being relatively simple to implement comparing to complex Recurrent Neural networks (RNN) models. In RNNs connections between neurons in a layer can create a cycle, allowing output from previous nodes to affect subsequent input nodes. This allows it to exhibit temporal dynamic behavior.

Regardless of the recent successes of RNNs in forecasting, practitioners may still be reluctant to try RNNs as an alternative since they may not have the expert knowledge to achieve satisfactory accuracy. Additionally, no established guidelines exist as to when traditional statistical methods will outperform RNNs, which architecture should be used or how their parameters should be tuned for improving forecasting accuracy.

Even for traditional forecasting methods, implementation does become a challenge when involves multiple time series. Export knowledge is key, starting with the identification of potential issues with the series (data preprocessing), moving to which model to choose, how to calibrate its parameters and finally how to measure performance, making this whole process lengthy and complex.

Therefore, adopting an optimized processual approach that can implement and measure performance between forecasting methods can help companies and practitioners explore different type of models available in the literature and easy scale across different business.

The objective of this investigation is to propose an optimized model calibration method using grid search algorithm for the best hyperparameters. Searching algorithms works to select the

hyperparameters first, so the resulting functional form of the prediction equation can be defined. After, the parameters of this function are found by minimizing an objective function.

This method aims to be applied in two different forecasting model types: (1) parametric statistical forecasting ES and Seasonal ARIMA; and (2) non-parametric RNN LSTM model, using data resampling and error performance techniques. Based on the calibration delivered by the method, forecast practitioners should be able to select the most suitable model.

2. Literature Review

The main characteristics of many time series are trends and seasonal variations that can be modelled deterministically with mathematical functions of time. Another important feature is that observations which are close together in time tend to be correlated (serially dependent). Much of the methodology in a time series analysis is aimed at explaining this correlation through statistical models, descriptive methods and more recent non-linear data driven models. Once a model is fitted to data and evaluated, it can be used to forecast future values.

Exponential Smoothing started in the 1950s and 1960s and provided a simple but useful classification of the trend and the seasonal patterns depending on whether they are additive or multiplicative. Exponential smoothing methods were improved later in 1985 to include additive damped trend and in 2003 methods with a multiplicative damped trend formed the set of equations we use in our method (GOOIJER; HYNDMAN 2006).

In 1976, Box and Jenkins integrated the existing knowledge at that time of, autoregressive and moving average concepts in one model and applied in a three-stage cycle of time series identification, model estimation (parameters) and verification (error metrics). The Autoregressive Integrated Moving Average (ARIMA) model as it is known had an enormous impact on the theory and practice time series analysis and implementation since then. With the popularization of the computer in the 90´s it popularized the use of these models for both ARIMA and Exponential Smoothing methods, improving forecasting performance by better selection and fitting it´s parameters (GOOIJER; HYNDMAN 2006).

After some initial skepticism discussed within M4 competition, Recurrent Neural Networks has shown superior feature extraction ability and has made a great contribution to improving the accuracy of time series forecasting. Additionally, having historical time series with an adequate length and additional features (multivariate) presents an opportunity to explore if more complex models than those obtained with traditional time-series provides better performance.

Recurrent Neural Networks can automatically handle temporal structures like trends and seasonality, learn arbitrary complex mappings, support multiple inputs and outputs and principally address long-term information preservation. One of the earliest approaches to address this was the Recurrent Neural Network called Long-Short term Memory LSTM defined by Hochreiter and Schmidhuber (1997).

Most of publications that approach model selection criteria for time series laid out the best statistics and optimization methods to find the parameters such as Billah, Hyndman, Koehler (2003). Usually, it is considered that pre-defined model hyperparameters that govern the general behavior of those models is known and was already selected. Error metrics are then used to compare the best model.

Questions remain on what are the different hyperparameters and how to apply it in the dataset? What are the performance metrics and how to measure?

According to Yang and Shami (2020) model parametrization is challenging and search algorithms for model calibration has attracted the attention of the scientific community. The idea of optimizing models by automatically searching the best hyperparameters was shared by Bergstra and Bengio (2012) who introduced the Random Search algorithm to compare with Grid Search, then a more complex iterative procedure called Bayesian search was discussed by Gelbart; Snoek; Adams (2014).

Let denote a functional form $\hat{f}$ with N hyperparameters. The domain of the n-th hyperparameter by $\hat{f}_n$ and the overall hyperparameter configuration space applied to $\hat{f}$ as $\hat{F} = \hat{f}_1 \times \hat{f}_2 \times \ldots \hat{f}_N$. A vector of hyperparameters is denoted by $\lambda \in \hat{F}$, and $\hat{F}$ with its hyperparameters initiated to λ is denoted by $\hat{F}_\lambda$.

The domain of a hyperparameter can be real-valued (e.g., learning rate), integer valued (e.g., number of layers), binary (e.g., whether to use early stopping or not), or categorical (e.g., choice of optimizer). For integer and real-valued hyperparameters, the domains are bounded to feasible values for reduced optimization time (FEURER; HUTTER, 2019).

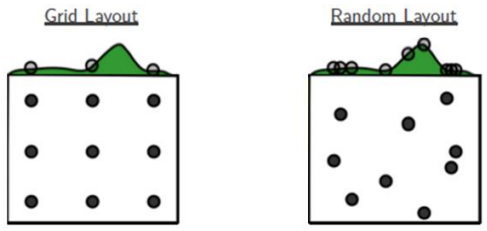
Studies devoted to facilitate this optimization search process includes strategies such as grid search (Feurer; Hutter, 2019), random search (Feurer; Hutter, 2019; Bergastra; Bengio, 2012), evolutionary (Metaheuristic) algorithms (Rashid; Fattah; Awla, 2018; Rere; Fanany; Arymurthy, 2016), Bayesian optimization (Feurer; Hutter, 2019 ; Gelbart; Snoek; Adams 2014) and differentiable architecture search (Liu; Simonyan; Yang, 2019).

The two basic hyperparameter optimization algorithms are the Grid Search and Random search. Grid search is also known as full factorial design, and it suffers from the curse of dimensionality

since the required number of function evaluations grows with the dimensionality of the configuration space.

A simple alternative to grid search is random search. It uses sampling of $\hat{F}$ at random until a certain allowed number for the search is exhausted. This works better than grid search when some hyperparameters are much more important than others (Bergastra; Bengio, 2012).

Figure 1 - Comparison of grid search and random search for minimizing a function with one



Source: Adapted from Bergstra and Bengio (2012)

Figure 1 illustrates how hyperparameter space is populated by different search schemes. With Grid search nine trials are tested, and these hyperparameters are defined discretely while in Random Search the same nine trials are explored with hyperparameters defined continuously, increasing the chance to find a better result for the objective function.

The choice of grid search approach was properly laid out by Bergastra and Bengio (2012) where he describes that this technique gives researchers some degree of insight into hyperparameter response function; There is no technical overhead or barrier to optimization and is simple to implement. On the other hand, depending on the hyperparameter selection, processing power and time is needed for grid search.

Using λ as a vector of hyperparameters. It can be composed of two or more hyperparameters for the functional form $\hat{f}$ with N hyperparameters $\hat{f}_N$. Let's define:

$$\text{a.} \quad \lambda_N^1 = \{\lambda_1, \lambda_2, \dots, \lambda_N\} \qquad (2)$$

where λ_N^1 is the first set of hyperparameters options for $\hat{f}_N$

$$\text{b.} \quad \Omega = \sum_{i=1}^{T} \lambda_N^i, \qquad (3)$$

then Ω is total Grid Search space containing all possible configurations for model training.

3. Methodology

The time series of daily observation is presented in figure 2 on the left-hand side. It is intended to forecast the next month's daily levels. This decision is used to plan supply and human resources to fulfil those orders. Considering this, our approach is summarized as follows:

a) Our dataset is composed by time series daily observations; It contains multiple features that can be used as additional regressors to the model;
b) We selected the ES method with damped parameter, Seasonal ARIMA and Recurrent Neural Network LSTM as possible forecasting models;
c) The hyperparameters search space is defined for each model;
d) To overcame issues with overfitting we will apply a resampling process to the data while searching for the best hyperparameters;
e) Validation measures are chosen to select the best hyperparameters in each model;
f) An automated process is implemented for the items above;

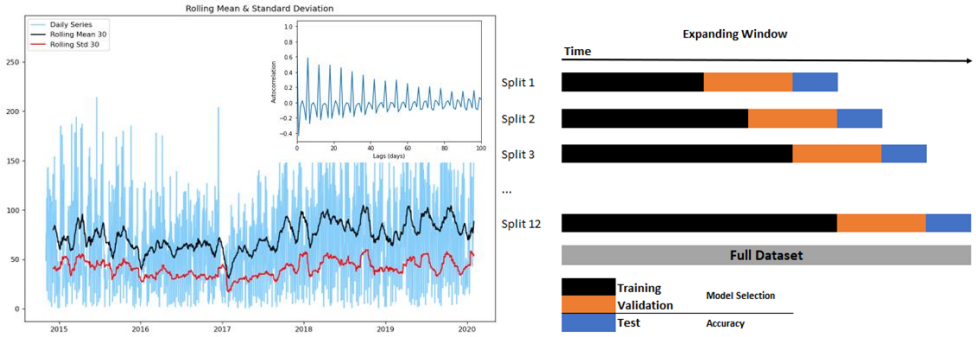
When evaluating a model, we are interested in the performance of the model on data that was not used to train it. The size of our dataset provides us an opportunity to perform the out of sample validation in three steps as shown in figure 2 on the right-hand side:

a) Training set: Fit the models with the chosen hyperparameters and the objective function;
b) Validation set: Evaluate how well your model was trained by comparing the performance of the fitted models (best candidates).
c) Test set: How the best candidates for each model perform on unseen real-world data.

To simulate a real case scenario, we chose the test set to be a month daily forecast and decided to do this for 12 months which means 12 training, validation, and test dataset splits.

Additionally, we defined the periodicity (or frequency) for time series as 6 as shown in the embedded left-hand side in figure 2. This is the number of observations before the seasonal pattern repeats and is a crucial hyperparameter for all 3 models here used. Periodicity can be determined by looking in the auto-correlation function after removing the trend of the series.

Figure 2 – Left-hand side Time Series and it's periodicity. Right-hand side the expanding window approach



Source: Elaborated by the author

3.1. Exponential Smoothing (ES) – Holt-Winters Method Hyperparameters

The Holt Winters method (Holt, 1957 and Winters, 1960), belongs to a class of exponential

smoothing methods that aims to capture the behavior of the time series, separating it in trend, season and error terms.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 1998).

By considering variations in the combinations of the trend and seasonal components, fifteen exponential smoothing methods are possible to simulate. This taxonomy was used in our simulation to calibrate and choose the best model and was introduced by Makridakis, Wheelwright and Hyndman (1998).

As the damping parameter (value from 0 to 1) can only be applied to the trend component, we removed the N(None) option. We want to damp the trend component, so it flattens over time instead of being linear.

Table 1 - Hyperparameters Grid Search for Holt-Winters Model

Hyperparameters	Value
Trend	Additive, Multiplicative
Damping	True or False
Seasonal	Additive, Multiplicative or None
Seasonal Frequency	6
Damping phi parameter	0.2, 0.4, 0.6, 0.8, 0.95
Target Variable Transform	Yes (Box-Cox), No

Source: Elaborated by the author

Additionally, we added the well-known Box–Cox transform of the time series as a hyperparameter. The purpose of these transformations is to simplify the patterns in the historical data by making it more consistent (normal distributed) across the whole data set. Simpler patterns may lead to more accurate forecasts.

From equation 2 we have N = 6 consisting of all hyperparameters λ_N possibilities for fitting an ES model and from equation 3 the total grid search space $\Omega = 120$ configurations to be fitted and evaluated in the validation partition.

3.2 SARIMA Method Hyperparameters

The Box–Jenkins Autoregressive Integrated Moving Average (ARIMA) model (Box; Jenkins, 1976) has two variations, the non-seasonal model is written as ARIMA(p,d,q) where p is the

order of auto-regression (AR), d is the degree of first differencing involved, and q is the order of moving average (MA), while the seasonal model is an extension written as SARIMA(p,d, q)(P,D,Q)s where s denote the periodicity and P, D and Q are seasonal equivalents p, d and q. The possibilities of our grid search space are a lot higher for this model than for exponential smoothing. To narrow down possible candidates for each of the hyperparameters (autoregressive, stationarity, moving average and seasonal equivalents) we use the following definitions below:

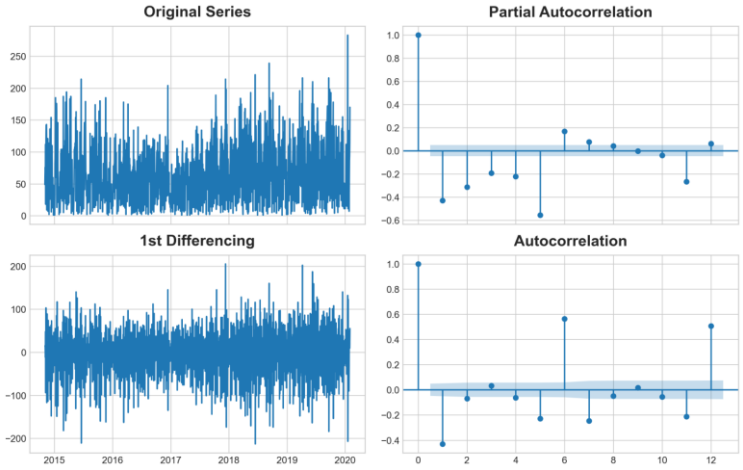
a) Autocorrelation function (ACF): by measuring the autocorrelation $c_k(y_t, k) = \frac{1}{n}\sum_{t=1}^{n-k}(y_t - \bar{y})(y_{t+k} - \bar{y})$ for different number of time steps t we define the lags which are significant for the hyperparameter. This is done by plotting the ACF correlogram. The ACF plot can provide answers to what order can the observed time series be modeled with an MA model;

b) Partial Autocorrelation Function: The partial autocorrelation at lag k is the autocorrelation between yt and y$t-k$ that is not accounted for by lags 1 through $k-1$;

$$pc_k(y_t, k) = \frac{Covariance(y_t, y_{t-k} | y_{t-1}, y_{t-2}, \dots, y_{t-k+1})}{\sqrt{Variance(y_t | y_{t-1}, y_{t-2}, \dots, y_{t-k+1}) Variance(y_{t-k} | y_{t-1}, y_{t-2}, \dots, y_{t-k+1})}}$$

The PACF plot can provide answers to what order can the observed time series be modeled with an AR model.

Figure 3 - Differencing on left-hand side and ACF/PACF on the right-hand side



Source: Elaborated by the author

c) Augmented Dickey Fuller Test for Stationarity (ADF): statistical significance test which the null hypothesis says that a unit root exists. Unit root is a characteristic of a time series y_t that makes it non-stationary, in other words, $y_t = \alpha * y_{t-1} + e$, where $\alpha = 1$. If series is non-stationary, we use levels up to 2 of the differencing order;

Table 2 - Hyperparameters Grid Search for SARIMA model

Hyperparameters	Type
Autoregressive Order	1, 2, 3, 4, 5
Differencing Order	0, 1
Moving Average order	1, 2, 4, 5
Seasonal Autoregressive Order	0, 1 ,2
Seasonal Differencing Order	0, 1
Seasonal Moving Average order	1, 2
Seasonal Frequency	6 days
Target Variable Transform	Yes (Box-Cox), No

Source: Elaborated by the author

Based on the ACF and PACF plots, table 2 shows the selected SARIMA orders. The autoregressive and moving average order of 6 was not included due to Seasonal order of 1 plays the same role in the model. From equation 2 and 3 we have N = 8, resulting in $\Omega = 960$ possible hyperparameters. Also, the Box–Cox transform options was added as a hyperparameter.

3.3 LSTM Method Hyperparameters

For LSTM we use the data normalization in the entire series y_t. Having different features with widely different scales fed to the model will cause the network to weight the features not equally. This can cause a false prioritization of some features over the others.

$$y'_t = y_t - \min(y)/(\max(y) - \min(y)) \qquad (4)$$

Like SARIMA, the grid space possibilities can grow exponentially and there are some hyperparameters such learning rate and epoch size that significantly increase the model fitting time. Table 3 summarizes the hyperparameters and how the range were defined. To define the grid search space LSTM, we require the dataset input dimension and the network topology.

Table 3 - Hyperparameters Grid Search for LSTM model

Hyperparameter	Type	Limit Definition	Values
Time Steps	Time-stamp input lag.	scatter plots of lags against the original series.	1, 6, 12
LSTM Layer	How deep the model will be. Can help generalize the model.	One/two hidden layers can theoretically model most complex functions.	1, 2, 3

Hyperparameter	Type	Limit Definition	Values
Units (Neurons) in each LSTM cell (Hidden States)	Learning capacity of the network Accuracy vs overfitting.	size between the input layer and the size of the output layer.	50, 100, 150
Batch size	Number of training samples to work through before the model's internal parameters are updated.	range within the periodicity of the series.	2, 4, 6, 12
Learning rate	Controls how quickly the model is adapted to the problem. Learning Algorithm is gradient descent.	Algorithm default is 0.01. Varying this rate to 0.1 and 0.001.	0.1, 0.01, 0.001
Noise Layer	Add noise to an existing model to prevent overfitting.	default value plus a lower value.	0.01, 0.001
Epoch Size	Number of complete passes through the training dataset.	observe the learning curve convergence.	200

Source: Elaborated by the author

From equation 2 and 3 we have N = 7, resulting in $\Omega = 648$ possible hyperparameters.

3.4 Training and Evaluation Metrics

To evaluate the performance of the models during validation and testing, several used statistical metrics can be used (Shcherbakov et al. 2013). Table 4 shows the selected measures used in the algorithm both in the validation partition and test partition. A combination of these metrics is proposed for best hyperparameters and model selection.

Table 4 - Evaluation Metrics for validation and testing partitions

Measure	Formula	Objective	Type
Mean Squared Error	$MSE = x_1 = \frac{1}{T}\sum_{1}^{T}(y_t - \hat{y}_t)^2$	Model Performance	Absolute Error
Geometric Mean Absolute Error	$GMRAE = x_2 = \sqrt{\prod_{t=1}^{T}\frac{\lvert y_t - \hat{y}_t\rvert}{\lvert y_t - y_t^*\rvert}}$	Model Selection	Relative Error
Mean Direction Accuracy	$MDA = x_3 = 1 - \frac{1}{N}\sum_{t} 1_{sgn(y_t - y_{t-1}) = sgn(\hat{y}_t - \hat{y}_{t-1})}$	Model Performance Model Selection	Absolute Error
Multi-Criteria Performance (MCP)	$MCP = (x_1x_2 + x_1x_3 + x_2x_3)\frac{\sin\left(\frac{2\pi}{3}\right)}{2}$	Model Selection	Relative Error

Source: Elaborated by the author

For the relative error metric GMRAE we chose the benchmark value y_t^* to be the previous time series step observation also known as Naïve Method. The result we expect when choosing a model measure both performance and selection type of errors (column Objective of table 4). For this reason, a combination of those metrics was used in what we define as multi-Criteria Performance index. A proposal of this metric was used by Parmezan, Souza and Batista (2019). To evaluate if the best results were achieve from the training dataset, the validation partition is used and the performance is measured.

4. Validation and Test Partition Results

For each of the 12 splits of our dataset, we choose the hyperparameters with the lower MCP. Table 5 shows the best hyperparameters chosen for each split by measuring the validation partition with the multicriteria performance index.

Table 5 - Validation Results for Exponential Smoothing

Split	ES Best Hyperparameters	ES MCP	SARIMA Best Hyperparameters	SARIMA MCP	LSTM Best Hyperparameters	LSTM MCP
1	['mul', True, 'mul', 6, 0.2, False]	0.65	[4, 0, 4, 2, 1, 1, 6, 'c', False]	0.34	[1, 3, 50, 4, 0.1, 200, 0.001]	0.31
2	['mul', True, 'mul', 6, 0.95, False]	0.57	[3, 1, 4, 0, 1, 2, 6, 'c', False]	0.42	[6, 2, 50, 12, 0.01, 200, 0.01]	0.42
3	['mul', True, 'mul', 6, 0.2, False]	0.4	[4, 1, 5, 1, 1, 2, 6, 'c', True]	0.33	[6, 1, 50, 4, 0.1, 200, 0.01]	0.25
4	['mul', False, 'add', 6, 0.95, True]	0.43	[3, 1, 1, 1, 1, 2, 6, 'c', True]	0.28	[1, 2, 50, 2, 0.1, 200, 0.001]	0.24
5	['mul', False, 'mul', 6, 0.8, False]	0.45	[1, 0, 1, 1, 1, 1, 6, 'c', False]	0.32	[1, 3, 100, 2, 0.1, 200, 0.001]	0.26
6	['add', True, 'mul', 6, 0.4, False]	1.02	[4, 0, 4, 0, 1, 1, 6, 'c', False]	0.37	[1, 3, 50, 2, 0.1, 200, 0.001]	0.29
7	['mul', False, 'mul', 6, 0.8, False]	0.43	[4, 1, 5, 0, 1, 1, 6, 'c', True]	0.33	[6, 3, 50, 12, 0.1, 200, 0.01]	0.27
8	['mul', False, 'mul', 6, 0.95, False]	0.63	[5, 0, 2, 0, 1, 2, 6, 'c', False]	0.34	[12, 3, 50, 2, 0.01, 200, 0.001]	0.32
9	['add', False, 'add', 6, 0.2, True]	0.38	[4, 1, 4, 2, 1, 2, 6, 'c', True]	0.31	[12, 1, 50, 12, 0.01, 200, 0.01]	0.32
10	['mul', False, 'mul', 6, 0.95, False]	0.68	[5, 1, 5, 0, 1, 2, 6, 'c', False]	0.3	[12, 1, 100, 4, 0.01, 200, 0.001]	0.25
11	['mul', True, 'mul', 6, 0.2, False]	0.99	[2, 0, 4, 1, 1, 2, 6, 'c', False]	0.26	[12, 1, 50, 6, 0.01, 200, 0.001]	0.26
12	['mul', False, 'add', 6, 0.2, True]	0.37	[1, 0, 1, 0, 1, 2, 6, 'c', False]	0.23	[12, 1, 50, 12, 0.01, 200, 0.001]	0.25

Source: Elaborated by the author

From table 5, exponential smoothing model has higher MCPs when comparing to SARIMA and LSTM in the validation partition, a result that was expected since the functional form has lower flexibility than SARIMA and LSTM. Whether the latter two have achieved a good generalization will be seen at the test partition, but overfitting was avoided as we have not seen any spikes in the MCP measure.

The difference set of hyperparameters observed in each split shows the importance to consider new available data to recalibrate the model for a new prediction.

We can now measure the test partition results to select the best model. By applying the best hyperparameters from table 5 in each test partition (simulated real life data) we can see how each model performed.

Table 6 - Absolute and Relative errors for measuring model performance and selection in test partition.

Split	MSE			GMRAE			1-MDA			MCP		
	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM
1	1319	1005	474	0.76	0.81	0.46	0.35	0.35	0.35	0.75	0.62	0.23
2	724	1431	318	0.59	0.88	0.37	0.35	0.22	0.17	0.38	0.76	0.10
3	1551	1459	1053	0.56	0.57	0.40	0.32	0.16	0.04	0.67	0.50	0.21
4	1723	1446	1560	0.63	0.43	0.53	0.32	0.32	0.40	0.80	0.53	0.72
5	834	729	480	0.69	0.66	0.62	0.31	0.23	0.23	0.45	0.35	0.24
6	2860	1712	1484	0.50	0.45	0.50	0.13	0.13	0.08	0.80	0.45	0.39
7	740	1030	529	0.52	0.54	0.39	0.27	0.19	0.27	0.31	0.37	0.20
8	358	288	395	0.45	0.41	0.39	0.15	0.19	0.23	0.12	0.11	0.14
9	3089	2347	1404	0.58	0.52	0.35	0.29	0.17	0.17	1.23	0.74	0.34
10	980	638	591	0.46	0.24	0.22	0.23	0.15	0.12	0.34	0.12	0.10
11	487	756	882	0.41	0.38	0.32	0.28	0.28	0.32	0.19	0.26	0.29
12	1777	1351	849	0.74	0.60	0.58	0.20	0.28	0.24	0.79	0.59	0.36
Avg	1370	1183	835	0.57	0.54	0.43	0.27	0.22	0.22	0.57	0.45	0.28

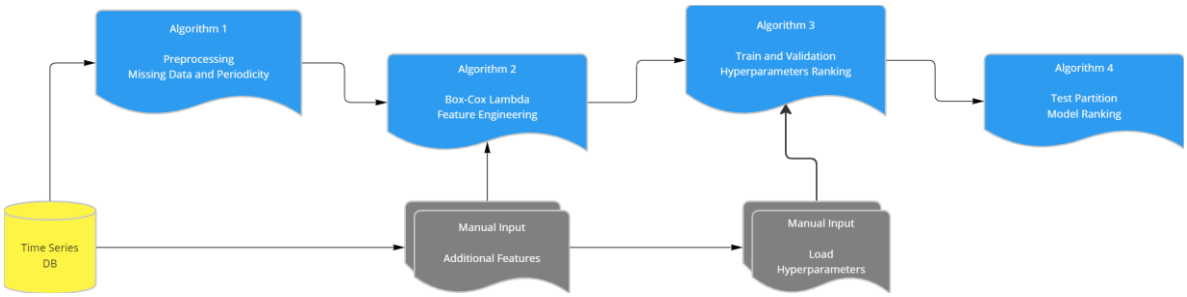
Source: Elaborated by the author

According to MCP criteria the LSTM model had better results in 9 months of the 12 months in the test partition. Overall average result for MCP is far better with LSTM than with the other two models. We can also see how all other metrics are lower on average for LSTM. This indicates that LSTM has indeed generalized better than ES and SARIMA and has not overfitted the data.

4.1. Choice of Model deployment

Through this methodology, it was possible to automate the model calibration and hyperparameter selection using the multi criteria performance index with a few manual inputs. Moving forward, LSTM is the choice for deployment through our sales forecasting portfolio. The downside is the grid search process that takes much more time than parametric models and depending on the number of splits in the dataset this can become computing cumbersome. We observed 12 times more length in the process for LSTM than SARIMA.

Figure 4 - Process Implementation



Source: Elaborated by the author

Figure 4 summarizes the automated process. After the model is selected, the hyperparameters

are evaluated monthly to capture changes in the trend levels, seasonality and other features that contribute to the forecasting. With our automated process this can be done without much intervention from expert knowledge.

5. Conclusion

In the proposed process, an automated approach based on grid search framework for parametric/non-parametric models is employed, aiming to reduce the dependence on expert knowledge. Such an automated system is more flexible, adaptative and efficient when dealing with multiple time series datasets that require forecasting.

The different models and calibration methods for time series can be challenging to implement in a business environment without a pre-defined process. The implementation time for better accuracy can conflict with the speed that analysts or management needs to have when making a decision. This can lead to very simple assumptions and not much accuracy. The process described here aims to overcome these challenges that once implemented can be easily scaled.

6. Bibliography

Aris A. Syntetos; Zied Babai; John E. Boylan; Stephan Kolassa; Konstantinos Nikolopoulos. Supply chain forecasting: Theory, practice, their gap and the future, EJOR, Volume 252, Issue 1, 2016, Pages 1-26.

Abolghasemi, M.; Beh, E.; Tarr, G.; Gerlach, R. Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion, Computers & Industrial Engineering, Volume 142, 2020, 106380.

Tealab A. Time series forecasting using artificial neural networks methodologies: A systematic review, Future Computing and Informatics Journal, Volume 3, Issue 2, 2018, Pages 334-340,

Parmezan, A.R.S.; Souza, V. M. A.; Batista, G. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model, Information Sciences, Volume 484, 2019, Pages 302-337,

Fildes, R.; Nikolopoulos, K.; Crone, S.F.; Syntetos, A.A. 2008. Forecasting and operational research: a review. Journal of the Operational Research Society, 59(9), pp.1150-1172.

Boone, T.; Ganeshan, R.; Jain, A.; Sanders, N., R. Forecasting sales in the supply chain: Consumer analytics in the big data era, International Journal of Forecasting, Volume 35, Issue 1, 2019, Pages 170-180.

Crone, S. F.; Hibon, M.; Nikolopoulos, K. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, IJF, Volume 27, Issue 3, 2011, Pages 635-660.

Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: Results, findings, conclusion and way forward, International Journal of Forecasting, Volume 34, Issue 4, 2018, Pages 802-808.

Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods, International Journal of Forecasting, Volume 36, Issue 1, 2020, Pages 54-74.

Makridakis, S.; Wheelwright, S. C.; Hyndman, R. J. Forecasting: methods and applications, New York (1998).

De Gooijer, J. G.; Hyndman, R. J. 25 years of time series forecasting, International Journal of Forecasting, Volume 22, Issue 3, 2006, Pages 443-473.

Hochreiter, S.; & Schmidhuber, J Long Short-Term Memory. Neural Computation 1997; 9 (8): 1735–1780.

Billah, B,; Hyndman, R.; Koehler, A. B. Empirical Information Criteria for Time Series Forecasting Model Selection. Journal of Statistical Computation and Simulation. 75. 2003.

Yang, L.; Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, pp.295-316.

Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. JMLR, 13. 281-305, 2012.

Gelbart, M. A.; Snoek, J.; Adams, R. P. Bayesian optimization with unknown constraints. UAI'14. 2014.

Feurer, M.; Hutter, F. Hyperparameter optimization, In: Hutter, F.; Kotthoff, L.; Vanschoren, J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham., 2019.

Rashid, T. A.; Fattah, P.; Awla, D. K. Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms, Procedia Computer Science, Volume 140, 2018, Pages 324-333.

Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search, arXiv:1806.09055 April 2019.

Shcherbakov, M.; Brebels, A.; Shcherbakova, N.L.; Tyukov, A.; Janovsky, T.A.; Kamaev, V.A. A survey of forecast error measures. World Applied Sciences Journal. 24. 171-176. 2013.