# Forecasting Retail Client Flow with LSTMs on Inconsistent Time Series

Pedro Gusmão, IEETA, Portugal, pedrogusmao@ua.pt
Ana Tomé, DETI, Portugal, ana@ua.pt
José Moreira, DETI, Portugal, jose.moreira@ua.pt

***Abstract***

*An important variable in retail future planning is forecasting client flow in stores. This research aims at introducing two Long Short-Term Memory network architectures for time series forecasting of client flow in retail stores. These models are allied with three main data preprocessing approaches: a data imputation method that standardizes store schedules; a harmonic regression method that captures and removes the seasonal and trend components of the time series and a sliding window sampling method to construct the network's training phase. Results were not extensively optimized but the framework leaves an open door for further improvements.*

***Keywords:*** *Neural Networks; Long Short-Term Memory; Time Series Forecasting; Data Imputation; Sliding Window.*

## 1. INTRODUCTION

Client flow forecasting can help retail companies optimize labor planning and, thus, reduce costs. However, the recording of data done by companies can be quite inconsistent or the working schedules can vary immensely.

This work experiments with two data preprocessing methods for dealing with univariate time series (TS) data of two retail stores, one aims to mitigate store schedule inconsistencies and the other tries to ease the models' forecasting task. The evaluated models are two Long Short-Term Memory (LSTMs) networks as they were created for modelling sequential data (Hopfield, 1982).

## 2. BACKGROUND

### *2.1. Time Series*

TS is a genre of data that is present in many fields such as astronomy, signal processing and other subjects that deal with temporal measurements. TS data is frequently collected in areas that employ forecasting tasks, for example, meteorology with weather forecasting or seismology with earthquake predictions.


*ISSN 2183-489X*

Technically, a TS is any collection of values $x(n)$ observed at successive points in time $n$. In order to better understand TSs and aid forecasting processes, it is common to separate them in individual components, such as:

- **Trend-Cycle** $\big(m(n)\big)$**:** a trend is characterized by when there is a long-term rise or drop in the data, it does not need to be linear, and it can change directions. A cyclic pattern refers to periodic fluctuations that are not of fixed period (not to be confused with seasonality). Cycles can be shorter than a calendar year but commonly describe longer term behaviours (more than a year). Cycle is usually considered to be merged with the trend component;
- **Seasonal** $\big(s(n)\big)$**:** seasonality takes place when the data demonstrates periodic fluctuations that usually occur in particular calendar seasons that recur (e.g., coat sales increase in the winter);
- **Noise** $\big(\varepsilon(n)\big)$**:** represented by irregular random sources of level variations or fluctuations.

Then, presupposing an additive model, a TS $x(n)$ can be mathematically represented as

$$x(n) = m(n) + s(n) + \varepsilon(n) \tag{1}$$

at time $n \in \{0, 1, 2, \dots, N-1\}$ with $N$ being the length of the TS.

### *2.2. Detrending and Deseasonalizing with Harmonic Regression*

Certain statistical qualities in TSs can be crucial to create skillful forecasts. Stationarity is one of those desired features (Oliver & Gujarati, 1993). A TS is stationary when its observations are not time-dependent, meaning that they are not greatly described by any trend or seasonality. Therefore, a way to aim for stationarity is by removing those components. For that goal, a harmonic regression, i.e., a linear regression with trigonometric terms, over Fourier terms can be used. In (Dong, Yang, Reindl, & Walsh, 2013), it is shown that there is a higher probability of achieving stationarity with such a method than with other popular methods.

Fourier terms are sine and cosine pairs where each represents one seasonal pattern. This means that if a TS is defined by several seasonal patterns, then the same number of terms must be included in the model. This is useful as high-frequency TSs are prone to demonstrate multiple seasonalities.

A linear trend can be expressed by a simple linear regression (Hyndman & Athanasopoulos, 2018), such as,

$$m(n) = \beta_0 + \beta_1 n \tag{2}$$

with $n$ being the regressor, $\beta_0$ being the intercept and $\beta_1$ being the slope. In (Taylor & Letham, 2018), it is demonstrated that a seasonal component $\big(s(n)\big)$ described by a single seasonal period ($z$) can be captured through a harmonic regression over Fourier terms, mathematically,

$$s(n) = \sum_{k=1}^{K} \left( a^{(k)} cos\left(\frac{2\pi kn}{z}\right) + b^{(k)} sin\left(\frac{2\pi kn}{z}\right) \right) \quad (3)$$

where $K$ is a smoothing factor, increasing it allows for more fluctuating seasonal patterns, and $a$ and $b$ are the regressors.

Considering the additive model (Equation (1)), the last two equations can be combined to capture both trend and seasonality components. For example, assuming $K = 1$, a TS $x(n)$ that is described by two seasonal patterns with periods $z_1$ and $z_2$ may be expressed by

$$x(n) = \beta_0 + \beta_1 n + \beta_2 cos\left(\frac{2\pi n}{z_1}\right) + \beta_3 sin\left(\frac{2\pi n}{z_1}\right) + \beta_4 cos\left(\frac{2\pi n}{z_2}\right) + \beta_5 sin\left(\frac{2\pi n}{z_2}\right) + \varepsilon(n) \quad (4)$$

Then, by removing the calculated trend and seasonality the forecasting model is left to approximate only the residual component $\varepsilon(n)$, i.e.,

$$\varepsilon(n) = x(n) - \left( \beta_0 + \beta_1 n + \beta_2 cos\left(\frac{2\pi n}{z_1}\right) + \beta_3 sin\left(\frac{2\pi n}{z_1}\right) + \beta_4 cos\left(\frac{2\pi n}{z_2}\right) + \beta_5 sin\left(\frac{2\pi n}{z_2}\right) \right) \quad (5)$$

An important assumption of this model is that the seasonal periods ($z$) are previously known, which may not always be the case. A valid approach for finding out the most relevant periods is through the application of a Fast Fourier transform (FFT) on the TS in question. A FFT is an algorithm that computes the Discrete Fourier transform converting the TS from a time domain representation to a frequency domain representation.

### *2.3. Long Short-Term Memory Network*

For the convenience of the reader, the LSTM internal processes are reviewed. A LSTM unit has three main concepts:

- The cell state $\mathbf{c}_t$, that consists of an encoded version of the information gathered from all the previously processed steps;
- The hidden state and output $\boldsymbol{h}_t$, which is similar to the cell state but its information is more focused on the previous step;
- Three gating mechanisms. An input gate ($\boldsymbol{i}_t$) that determines what information from the candidate memory $(\tilde{\boldsymbol{c}}_t)$ should be appended to the cell state. A forget gate ($\boldsymbol{f}_t$) that determines what information should be removed or "forgotten" from the cell state ($\mathbf{c}_t$). Lastly, an output gate ($\boldsymbol{o}_t$) that controls which values from the cell state ($\mathbf{c}_t$) should be appended to $\boldsymbol{h}_t$.

The unit can be expressed in three steps, where all gates and the candidate memory share the same format. They all deal with a combination of the input $\boldsymbol{x}_t$, the previous output $\boldsymbol{h}_{t-1}$ and their own learnable weights $\boldsymbol{W}$ and $\boldsymbol{R}$. As a first step let the forget gate be defined as

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{R}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}\right), \quad (6)$$

The symbol σ refers to the sigmoid function, that transforms all values into ]0,1[. If the values are close to zero, the data is to be removed, and if they are close to one, then it is to be kept.

In the second step, the unit decides what information is to be appended to the cell state ($\boldsymbol{c}_t$). The input gate ($\boldsymbol{i}_t$) is built to decide which values should be added and the new candidate values $\left(\tilde{\boldsymbol{c}}_t\right)$ for the cell state are created. Formally,

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right) \quad (7)$$

$$\tilde{\boldsymbol{c}}_t = tanh\left(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{R}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}\right), \quad (8)$$

After this, the cell state is updated from $\boldsymbol{c}_{(t-1)}$ to $\boldsymbol{c}_{(t)}$ with the element-wise product,

$$\boldsymbol{c}_t = tanh(\boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t), \quad (9)$$

The last step assembles the unit's output $\boldsymbol{h}_t$. First, the output gate decides through the sigmoid which values of the updated cell state should be carried on in the hidden state $\boldsymbol{h}_t$. Then, the element-wise product to apply that is done. Mathematically,

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{R}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}\right) \quad (10)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{c}_t, \quad (11)$$

The training of standard ANNs is done using the backpropagation algorithm. However, an extension is necessary for a RNN such as the LSTM. For that goal, the backpropagation through time (BPTT) algorithm was developed (Gers, Schmidhuber, & Cummins, 2000).

## 3. CASE STUDY AND DATA PREPARATION

The data used in the experiments are univariate half-hourly TSs (Figure 1) that belong to two retail stores (A and B). Store A presents records between 2015-01-02 and 2019-07-23. Concretely, 1649 days were recorded between those dates as 14 days are missing (e.g., Christmas days). On the other hand, store B has 976 days registered between 2017-06-12 and 2020-10-18 (248 missing days). It is also worth noting that the schedules in both stores were not the same in all the registered days.

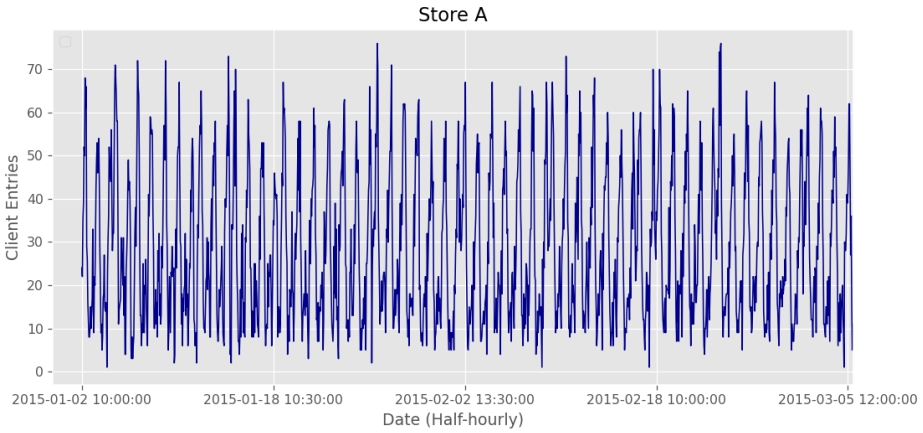


Figure 1. Client flow in store A (first 1400 observations).

The objective is to predict the number of client entries in the last 30 registered days using only past observations of that same variable. The TSs are, therefore, divided into train[1] and test set. For store A the test set (692 observations) begins in 2020-06-23 and for store B the test set begins in 2020-09-19 (746 observations), observations before those dates form the training set. It should be mentioned that store B displays the effect of the COVID-19 lockdown (Figure 2) which might influence forecasting results.

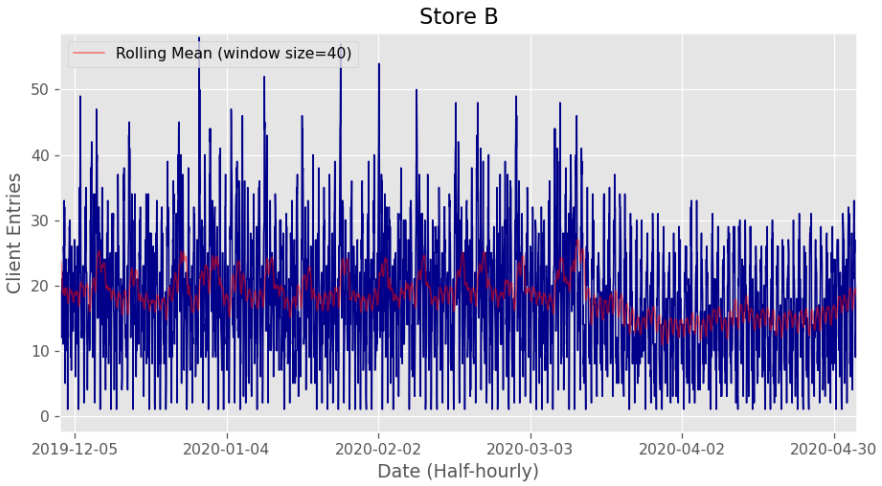


Figure 2. The effect of the COVID-19 lockdown starts near the beginning of March and lasts until the end of April.

### *3.1. The Sliding Window Method*

In TS forecasting, future values have their basis on their past values. Because of this, the problem of forecasting is classified as a specific regression problem, an autoregression. In ML, regression is a type of supervised learning problem that refers to discrete outputs. In this context, supervised learning refers to adjusting a model by comparing the forecasted values with the real values. This adjustment is possible with a TS' past values and, as such, the data can be framed in that way.

[1] In the descriptions of the data preprocessing methods the term TS is sometimes used in reference it to its train set.

The sliding window sampling method (Figure 3) was the chosen approach and it consists in transforming the data into several fixed-size combined input and output. For that, three parameters are considered:

1. The stride $s$ in which the window is to be slid.
2. The number of lagged values $p$ in the input;
3. The number of values in the outputs that is equal to the forecast horizon $H$.

As this research studies univariate TS problems the inputs are scalars from vectors $\boldsymbol{x}_t$ defined as

$$\boldsymbol{x}_t = (v(j-p+1),\ v(j-p+2), \dots ,\ v(j))^T, \tag{12}$$

that were formed by embedding $p$ samples using the sliding window. Where $v(j)$ can either represent a TS, $x(j)$, $j \in p-1, \dots, J-H-1$, where $J$ is the number of training records, or, if the trend and seasonality were removed through the harmonic regression (e.g., Equation (5)), its residual component $\varepsilon(j)$. The outputs $\boldsymbol{y}_t$ are similarly defined as,

$$\boldsymbol{y}_t = \big(v(j+1), \dots, v(j+H)\big), \tag{13}$$

Knowing this, the number of training samples $g$ created by the sliding window method on some TS is given by $g = J - p - H + 1$.

It is worth noting that just as $H$, $p$ is a sensitive parameter. While windows with few lagged values may prove to have insufficient information, large windows can increase complexity and reduce the network's learning capabilities. Therefore, if the architecture of the forecasting model allows, various $p$ values should be tested. In this work, the stride $s$ was set to 1 in every experiment in order to maximize the number of training samples.

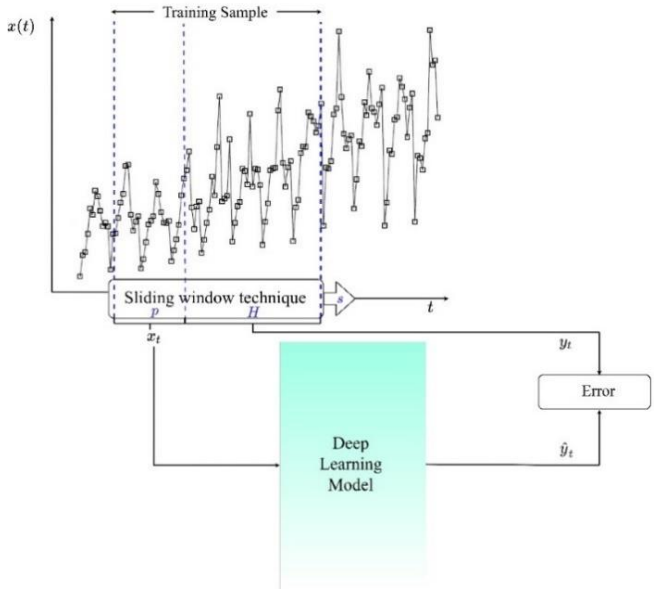


Figure 3. The sliding window technique in training. Adapted from: (Paoli, Voyant, Muselli, & Nivet, 2010)

### *3.2. Standardizing the Store Schedules*

As mentioned, the schedules in the stores may vary from day to day. Since the forecasting problem is modelled as a univariate forecasting problem no contextual information is given and that includes information regarding schedules. Thus, the model does not know for what half-hours in the day it is trying to predict. A solution can be achieved with schedule standardization by constraining the TSs to a specified schedule. For that, missing samples are imputed and extraneous samples are removed.

So that data integrity is not greatly threatened the chosen schedule is the schedule that is most frequent. Curiously, both stores have the same most frequent schedule, which is from 09:00:00 to 21:00:00 (25 samples). As it is of low complexity, linear interpolation was the chosen imputation method. For all the standardized TSs, $H = 750$ as it corresponds to 30 days with 25 samples each. This standardization, however, originates two drawbacks. First, the forecasting model is forced to forecast a predefined schedule. In second, during the evaluation phase imputed entries cannot be considered and, as such, complexity rises in comparing results with unstandardized TSs.

One option that counters the first drawback is creating more forecasting models that fill remaining schedules with predictions. This, however, is not explored in this work.

Regarding the second point, in order to compare forecasting performance with the imputed versions it should be known that artificial samples and samples that are outside the 09:00:00 to 21:00:00 schedule are invalid for evaluation. Knowing this, store A possesses 679 and store B possesses 743 predictions valid for comparison.

### *3.3. Discovering the Seasonal Periods*

There are two prerequisites in order to use the harmonic regression method presented in subsection 2.2:

1. The TSs schedules must be standardized;
2. The relevant seasonal periods must be discovered.

The first condition is satisfied with the already described standardization method. The second is met by applying FFTs[2] to the TSs and observing the corresponding plots. For both of the standardized TSs two seasonal periods are dominant, a daily and half-daily period (Figure 4), i.e., $z_1 = 25$ and $z_2 = 12.5$.

[2] This method should not be used if the TS has a highly expressive trend.

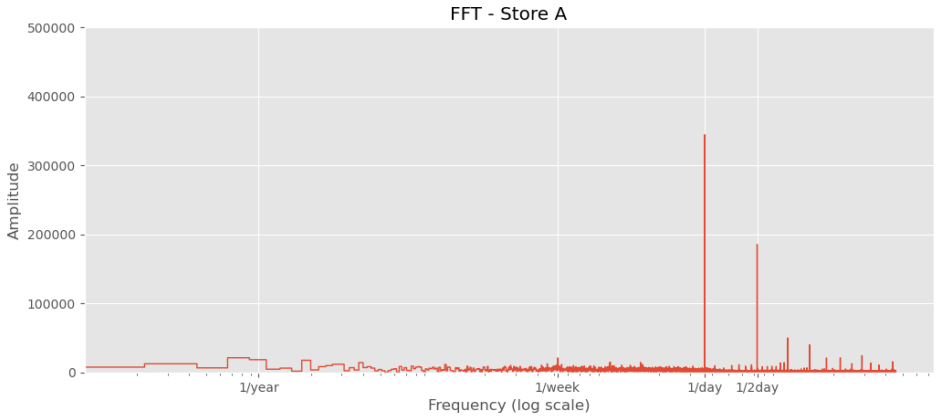


Figure 4. Store A FFT plot.

## 4. EXPERIMENTAL SETUP

This section further details the data preprocessing methods and the technologies used to develop the DL models.

### *4.1. Technology and Libraries*

The neural models and related operations were developed using Python 3.7.9 with the following libraries:

- *Pandas*[3] was used to extract the data, split the data into train and test sets and to apply the imputation method;
- *Numpy*[4] provides efficient and optimized functions to deal with multidimensional arrays. It was used in various phases;
- *Scikit-Learn*[5] was used for the preprocessing harmonic regression model;
- *TensorFlow*[6], developed by the *Google Brain Team*, is an open-source library for distributed numerical and auto-differentiable computations. These computations are the primary support for many developed deep learning algorithms. It was initially implemented in C++, but a native Python API is also available.

The Tensorflow module has a sub-module containing an API for the Keras library and provides a user-friendly interface to compose various deep learning models.

[3] pandas.pydata.org
[4] numpy.org
[5] scikit-learn.org
[6] tensorflow.org

### *4.2. LSTM Forecasting Model Architectures*

So that a LSTM model provides a multi-step forecast with a horizon $H$ two possibilites are considered in this work (Figure 5).

The first option (LSTM-1) consists in having a linear fully-connected layer (FC) with $H$ units acting as the output layer (Wang, Zhu, & Li, 2019 and Masum, Liu, & Chiverton, 2018). In this output layer, every unit is connected to every element in $\boldsymbol{h}_t$. Thus, a forecast $\widehat{\boldsymbol{y}_t}$ is described by,

$$\widehat{\boldsymbol{y}_t} = \boldsymbol{W}^{(d)}\boldsymbol{h}_t + \boldsymbol{b}, \tag{14}$$

where $\boldsymbol{h}_t$ is a reduced context representation of $\boldsymbol{x}_t$ outputted from the LSTM layer at the last processing step $t = p - 1$, $\boldsymbol{W}^{(d)}$ is the weights of the FC layer and $\boldsymbol{b}$ is the bias parameter. This architecture is more popular in classification problems (Rao, Huang, Feng, & Cong, 2018).

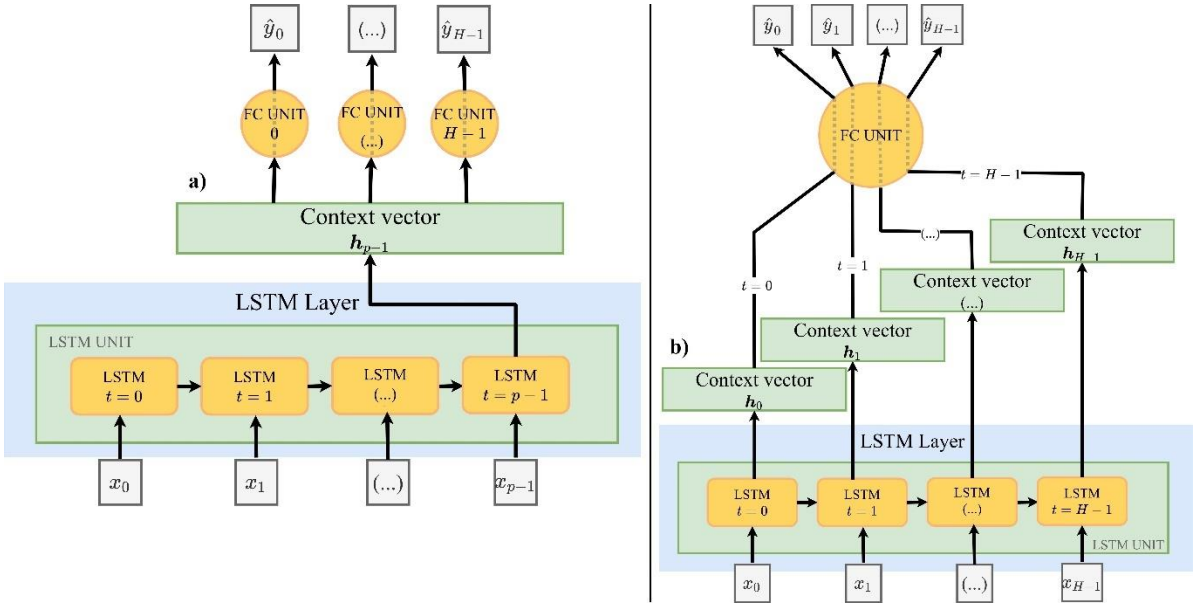


Figure 5. a) The first LSTM model architecture (LSTM-1). The LSTM blocks represent one LSTM unit unrolled through $t$. b) The second LSTM model architecture (LSTM-2).

The second architecture (LSTM-2) comprises in constructing one output $\widehat{y_t}$ at each processing step. This means that instead of capturing $\boldsymbol{h}_t$ only at the last step, every constructed $\boldsymbol{h}_t$ is now returned in each one. Then, a FC layer with one unit is applied in every step, one at a time, updating its weights only after concluding the last. This approach adds one constraint to its functioning: in order to produce a multi-step forecast of size $H$ its input needs to be of the same size ($p = H$), which makes the training phase computationally costly. In contrast, LSTM-1 has more flexibility since different $p$ values can be experimented in the first option.

### *4.3. Parameter Configuration*

In both models the LSTM layer hyperparameter *units* was set to 32 in every instance. This parameter refers to the size of the LSTM output $\boldsymbol{h}_t$ and affects the memory capacity of the LSTM. In order to

keep a low complexity, the rest of the configurations have their default values as defined by Keras[7] except for the LSTM-2 where the parameter *return_sequences* is set to *True* so that $\boldsymbol{h}_t$ is outputted in every processing step of the LSTM section.

In Table 1, the number of learnable weights plus biases for each model is presented. Since in LSTM-2 the weights of the FC unit are reused in each step it shows a considerable reduction in comparison to the FC layer present in the LSTM-1 and the model overall. Yet, despite the noticeable difference the LSTM-2 model takes much longer to train due to its input vectors having the size of $H$.

Table 1. Number of learnable parameters (weights and biases).

| Model | LSTM | FC | Total |
|---|---|---|---|
| LSTM-1 | 4352 | 24750 | 29102 |
| LSTM-2 | 4352 | 33 | 4385 |

In all settings, the networks were trained for 50 epochs with early stopping ($patience = 5$) to prevent overfitting. The mean square error was assigned as the loss function and ADAM was used as the gradient descent optimizer (Kingma & Ba, 2015). No hyperparameter-tuning was conducted to improve results.

## 5. RESULTS

In this section, the results for each store are presented. A template is followed where results produced from both LSTM models on unstandardized versions of the data are compared with:

1. Performances obtained from standardized data;
2. Performances obtained from standardized data that was also preprocessed with the harmonic regression method. For simplicity, a $K = 1$ is assumed.

The performance of a baseline naive model is also shown to not only compare performances between the three data settings but also between the models. This baseline model uses the previous $H$ observations as forecasts and that makes it the same in both the standardized settings.

The used metrics are the mean absolute error (MAE), the root mean square error (RMSE) and the mean arctangent absolute percentage error (MAAPE) (Kim & Kim, 2016). Additionally, $p$ values of $\{10, 25, 50, 100, 150, 175\}$[8] (maximum corresponds to a week) were tested on the LSTM-1 model with the best being shown in the following subsections.

[7] www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

[8] Greater values were not tested due to computational constraints.

### *5.1. Store A*

The following tables (Table 2, Table 3 and Table 4) present the results for the different store A's settings. Also, Figure 6 and Figure 7 portray the effects of the first and third data settings.

Table 2. Results with unstandardized data (Store A).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 18.131 | 21.741 | 54.379 |
| LSTM-1 ($p = 150$) | 11.145 | 13.594 | 38.728 |
| LSTM-2 ($p = 692$) | 12.553 | 14.935 | 43.684 |

Table 3. Results with standardized data (Store A).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 8.315 | 10.899 | 29.543 |
| LSTM-1 ($p = 50$) | 4.859 | 6.329 | 18.205 |
| LSTM-2 ($p = 750$) | 6.584 | 9.401 | 18.953 |

Table 4. Results with standardized data that was detrended and deseasonalized with the harmonic regression method (Store A).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 8.315 | 10.899 | 29.543 |
| LSTM-1 ($p = 25$) | 4.795 | 6.279 | 17.954 |
| LSTM-2 ($p = 750$) | 5.294 | 6.908 | 19.481 |

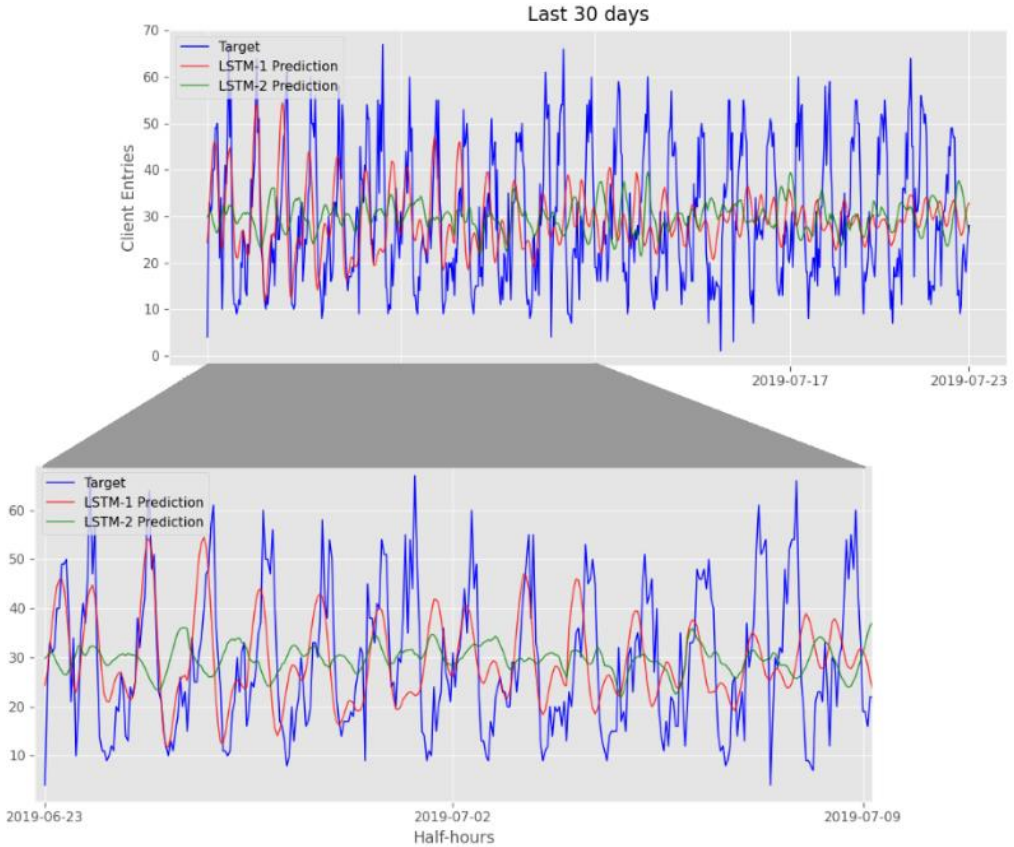


Figure 6. LSTM-1 ($p = 150$) and LSTM-2 performance in store A's unstandardized TS (first 16 days zoomed in).

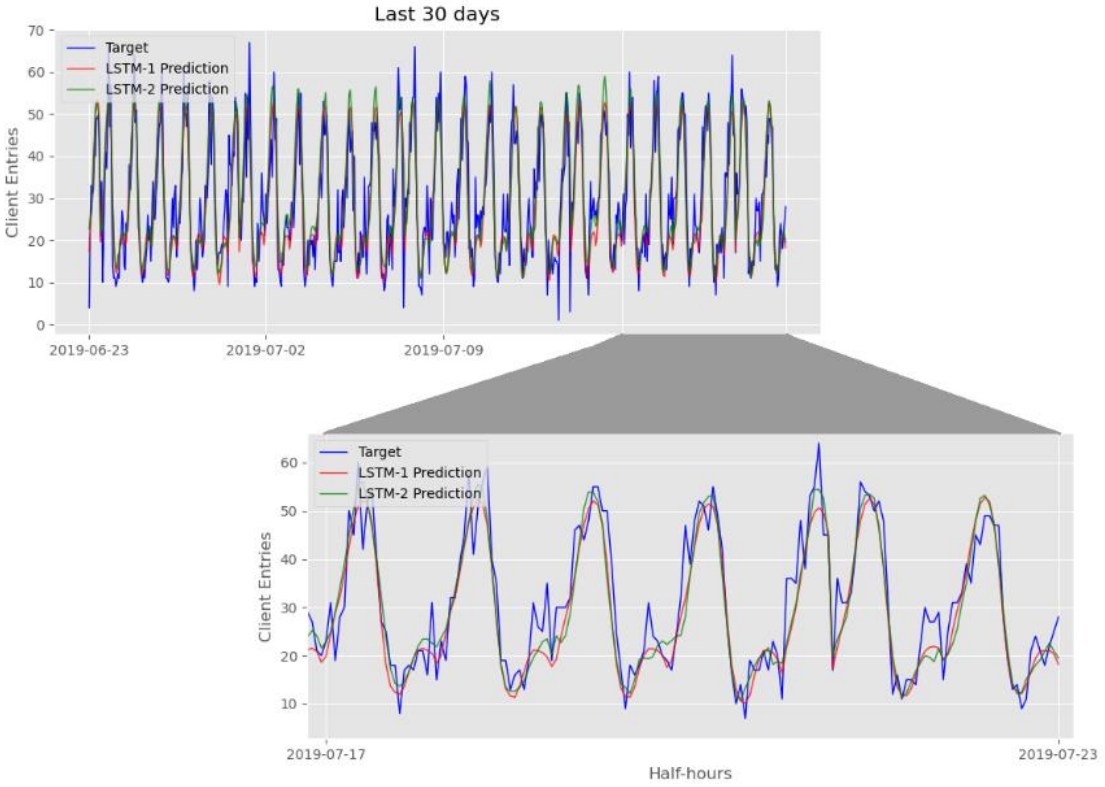


Figure 7. LSTM-1 ($p = 25$) and LSTM-2 performance in store A's standardized TS that was previously deseasonalized and detrended.

### *5.2. Store B*

Tables 5 to 7 show the models' performances in store B. As with store A, Figure 8 and Figure 9 display store B's forecasts.

Table 5. Results with unstandardized data (Store B).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 6.517 | 8.428 | 35.502 |
| LSTM-1 ($p = 50$) | 5.824 | 7.323 | 34.008 |
| LSTM-2 ($p = 746$) | 7.361 | 8.842 | 41.570 |

Table 6. Results with standardized data (Store B).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 5.983 | 7.678 | 32.912 |
| LSTM-1 ($p = 50$) | 4.018 | 5.060 | 25.690 |
| LSTM-2 ($p = 750$) | 4.646 | 5.970 | 28.273 |

Table 7. Results with standardized data that was detrended and deseasonalized with the harmonic regression method (Store B).

| Model | MAE | RMSE | MAAPE (%) |
|---|---|---|---|
| Naive | 5.983 | 7.678 | 32.912 |
| LSTM-1 ($p = 50$) | 4.019 | 5.063 | 25.489 |
| LSTM-2 ($p = 750$) | 4.341 | 5.498 | 27.108 |

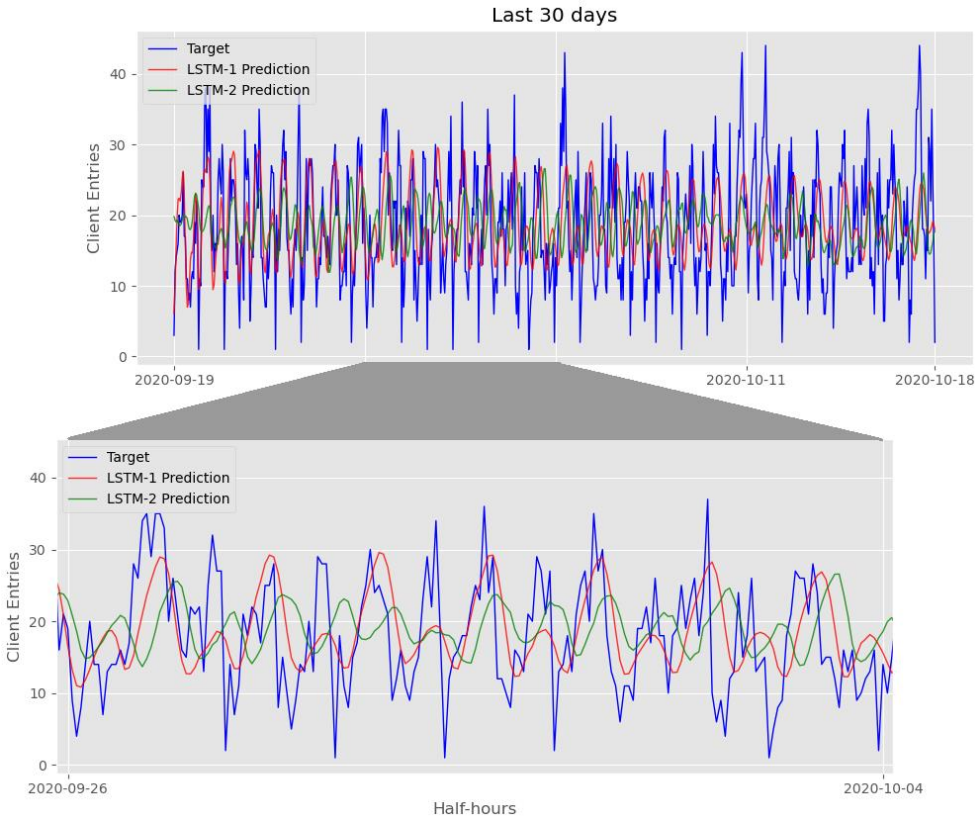


Figure 8. LSTM-1 ($p = 50$) and LSTM-2 performance in store B's unstandardized TS.

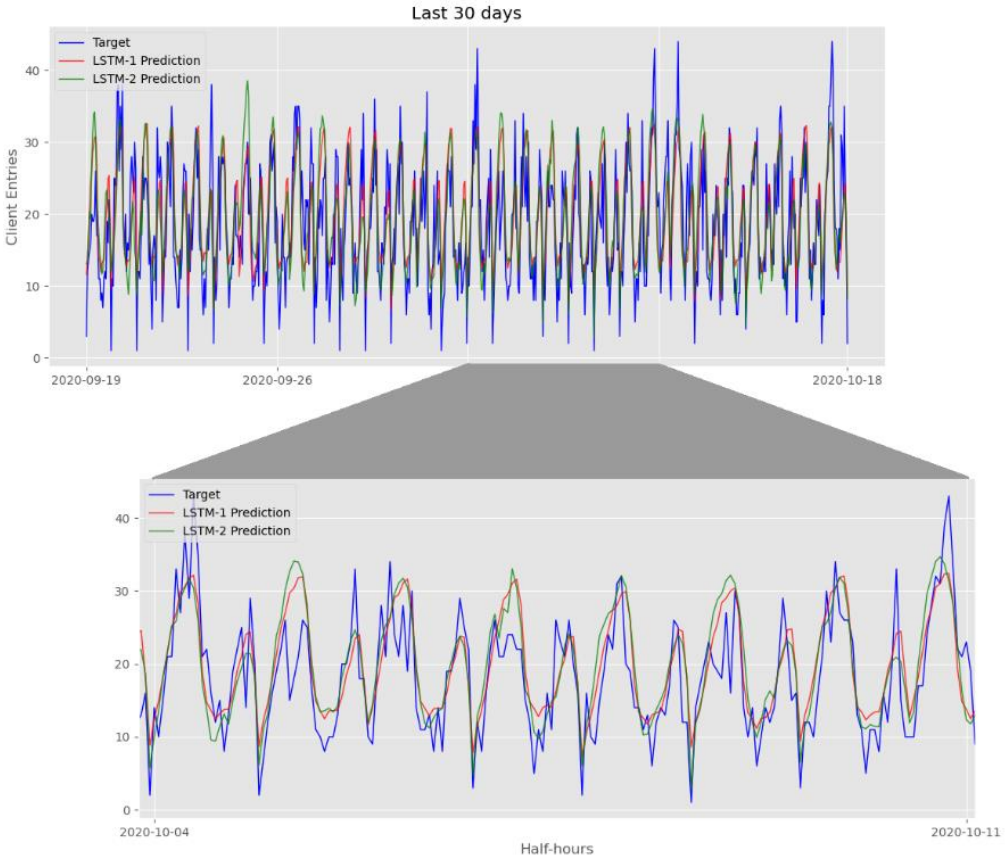


Figure 9. LSTM-1 ($p = 50$) and LSTM-2 performance in store B's standardized TS that was previously deseasonalized and detrended.

### *5.3. Discussion*

While comparing results between the settings three general conclusions are drawn:

1. Results with standardized data are always better than with unstandardized data;
2. Harmonic regression did not significantly improve the results, especially on store A. This is aligned with the intuition that ANNs are able to model TSs in an autonomous fashion (Gorr, 1994);
3. Overall, store A shows better results than store B and that might be justified by the negative influence of the COVID-19 lockdown observed in store B or that combined with the imputation method being more severe on the integrity of the data.

In between models some points also take place:

1. Both LSTM models were in great part better than the baseline model.
2. The LSTM-1 model configured with its best $p$ was the best model in all standardized settings. Despite the small difference in performance, a possible reason for the LSTM-2 model having worse results is the fact that its first predictions ($\widehat{y_0}$) do not have available any direct information of the past;
3. No relationship could be rigorously assumed between $p$ and the results obtained with the LSTM-1. Despite the need for more research, the crude deduction that p should be greater or at least equal to the longest seasonal period (in this case, $z = 25$) can be made.

## 6. CONCLUSION

In this paper, customer flow forecasts for two retail shops with high-frequency TSs were produced with two LSTM models on raw data and on two preprocessing setups. Both setups standardize the shops' schedules but one of them also applies a harmonic regression method to remove trend and seasonality. Each of the scenarios presents better results than the other but the demonstration of the positive impact of data standardization is the major contribution of this work. A disadvantage of standardization, however, is the constrainment of predicting a predefined schedule. Although this can be overcome by using models that are focused in other schedules, this option is left for future work such as the fine-tuning of the LSTMs and the preprocessing harmonic regression models in order to obtain preciser verdicts.

## REFERENCES

Dong, Z., Yang, D., Reindl, T., & Walsh, W. M. (2013). Short-term solar irradiance forecasting using exponential smoothing state space model. *Energy*, *55*, 1104–1113. https://doi.org/10.1016/j.energy.2013.04.027

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM.

*Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Gorr, W. L. (1994). Editorial: Research prospective on neural network forecasting. *International Journal of Forecasting*, *10*(1), 1–4. https://doi.org/10.1016/0169-2070(94)90044-2

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice. Retrieved 23 October 2020, from https://otexts.com/fpp2/

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669–679. https://doi.org/10.1016/j.ijforecast.2015.12.003

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, {ICLR} 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Retrieved from http://arxiv.org/abs/1412.6980

Masum, S., Liu, Y., & Chiverton, J. (2018). Multi-step time series forecasting of electric load using machine learning models. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10841 LNAI*(January), 148–159. https://doi.org/10.1007/978-3-319-91253-0_15

Oliver, F. R., & Gujarati, D. (1993). Essentials of Econometrics. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, *156*(2), 322. https://doi.org/10.2307/2982744

Paoli, C., Voyant, C., Muselli, M., & Nivet, M. L. (2010). Forecasting of preprocessed daily solar radiation time series using neural networks. *Solar Energy*, *84*(12), 2146–2160. https://doi.org/10.1016/j.solener.2010.08.011

Rao, G., Huang, W., Feng, Z., & Cong, Q. (2018). LSTM with sentence representations for document-level sentiment classification. *Neurocomputing*, *308*, 49–57. https://doi.org/10.1016/j.neucom.2018.04.045

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Wang, Y., Zhu, S., & Li, C. (2019). Research on Multistep Time Series Prediction Based on LSTM. *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, 1155–1159. IEEE. https://doi.org/10.1109/EITCE47263.2019.9095044