Time series forecasting in retail sales using LSTM and Prophet

Clony Junior
Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Portugal

Pedro Gusmão
Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Portugal

José Moreira
University of Aveiro and Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Portugal

Ana Maria Tomé
University of Aveiro and Institute of Electronics and Informatics Engineering of Aveiro (IEETA), Portugal

ABSTRACT

Data science highlights fields of study and research such as time series, which, although widely explored in the past, gain new perspectives in the context of this discipline. This chapter presents two approaches to time series forecasting, Long Short-Term Memory (LSTM), a special kind of Recurrent Neural Network (RNN), and Prophet, an open-source library developed by Facebook for time series forecasting. With a focus on developing forecasting processes by data mining or machine learning experts, LSTM uses gating mechanisms to deal with long-term dependencies, reducing the short-term memory effect inherent to the traditional RNN. On the other hand, Prophet encapsulates statistical and computational complexity to allow broad use of time series forecasting, prioritizing the expert's business knowledge through exploration and experimentation. Both approaches were applied to a retail time series. This case study comprises daily and half-hourly forecasts, and the performance of both methods was measured using the standard metrics.

Keywords: Time Series Forecasting, Neural Networks, Deep Learning, Long Short-Term Memory (LSTM), Prophet, Machine Learning, Retail.

1. INTRODUCTION

The rise of data science as a discipline with application in very different fields drives many developments in data mining, machine learning (ML) or data visualization. One of the most popular topics is time series (TS) forecasting, where the challenge is to find models to represent past observations and to estimate future values. Besides traditional statistical methods that have been used for a long time, new forecasting methods are appearing that include or are built entirely on ML techniques, such as decision trees, support vector machines and neural networks.

This work presents two recent TS forecasting approaches, and the motivating case study is to predict client visits in retail stores. TS forecasting in retail mainly focuses on predicting products' demand using ARIMA, SVR, multilayer perceptron (MLP) and hidden Markov models (İşlek & Gunduz Oguducu, 2017; Maaß et al., 2014). For its part, the solutions used by current commercial systems to predict the number of visiting clients and sales build on empirical methods.

The first approach uses Long Short-Term Memory (LSTM), a special kind of Recurrent Neural Network (RNN) that can deal with long sequential data such as TSs. Some ANN features are challenging to achieve using other TS forecasting approaches because ANNs are self-adaptive, support nonlinear models, and deal with multivariate problems, missing values, and noise. LSTM adds mechanisms to reduce short-term memory's effects on the other kinds of ANNs, i.e., LSTM can learn which data should be kept and which data should be forgotten. The reason for choosing the LSTM, however, comes from its forecasting potential as there are studies that demonstrate clear superiority of LSTMs against ARIMA which is a classical model that is very much established in the TS forecasting scene (Kolarik & Rudorfer, 1994; Siami-Namini et al., 2019). The second approach uses Prophet, an open-source library developed by Facebook for time series forecasting. Prophet uses a generalized additive model (Hastie & Tibshirani, 1987) to represent trends, seasonality and holidays. It provides an intuitive set of parameters that are easy to interpret even without in-depth knowledge of the underlying model.

While LSTM focuses on developing forecasting processes by experts in data mining or ML, Prophet aims to use general-purpose methods, keep domain experts in the loop, and provide judgmental forecasts. The main challenges in both approaches are to build the model that best fits a TS, and to avoid biased estimations and complex models. This includes understanding the data and the problem, choosing and tuning the forecasting model, and validating the model using adequate metrics. These issues are explained and discussed by example in this chapter.

The target audience is practitioners and academics interested in understanding and implementing TS forecasting solutions. The case study consists in predicting the number of client visits in retail stores, but the contents and issues presented in this chapter are also of interest in other use cases.

The chapter is organized as follows. Section 2 presents the background on time series forecasting using neural networks and deep learning (LSTM), and off-the-shelf tools for business scenarios (Prophet). Section 3 presents the data and the case study. Section 4 presents the methodology and the implementation of the case study using LSTM and Prophet. Section 5 presents the results of the two implementation strategies and a discussion of both approaches. Section 6 presents the conclusions and guidelines for future research.

2. BACKGROUND

TS forecasting is an important task that helps organizations with capacity planning. For instance, in the retail business, sales forecasting is behind all strategic and planning decisions. An accurate sales forecast helps with efficient inventory and logistics management.

A TS is a sequence of values measured at successive time instants. The time elapsed between the measurements is usually equally spaced, e.g., minutes, hours, days or months. TS can have three types of components:

- **Trend.** A trend exists when there is a general tendency of a time series to increase, decrease or stagnate over time. For instance, the increase or decrease of the unemployment rate, mortality rate or the number of houses in a city can hold for a considerable time.

Commented [PG1]: Incluir artigo de MLP vs ARIMA: Time series forecasting using neural networks

LSTM vs ARIMA: A Comparison of ARIMA and LSTM in Forecasting Time Series

Commented [PG2]: novo

- **Seasonal.** A seasonal pattern occurs when the data is influenced by factors every year, month, day of the week, etc. For instance, ice-cream sales increase in summer, woolen clothes sales increase in winter and the number of visitors to a museum increases during weekends.
- **Irregular.** This component represents random variations of a time series caused by unpredictable factors or incidences, such as strikes, floods, etc. There is no statistical method for measuring random fluctuations.

There are two preconditions for using TS forecasting methods: 1) there must be numerical data about the past, and 2) it should be reasonable to assume that past patterns will continue in the future. Many forecasting methods exist, some specializing in specific problems, and the most appropriate one for a given use case depends mainly on the data. This section presents two methods that will be used later to implement the same problem.

2.1 Long Short-Term Memory Neural Networks

The computational models of primitive ANNs were inspired by the brain's neural activity (Rosenblatt, 1957). These pioneer proposals are simple linear models since efficient algorithms to optimize multilayer networks' connections were missing. However, with the application of the backpropagation algorithm proposed in ANNs (Werbos, 1982) these models have been accomplishing great achievements in different fields, like medicine, economy, and so on (Abiodun et al., 2018). During the last decade, with the availability of large datasets and computing power, these models have evolved to efficient and versatile computational models. Modern architectures like GoogleNet (Szegedy et al., 2015) are composed of repeatable units. These units are made up of several blocks that can have different functionalities. The Long Short-Term Memory (LSTM) is an example of such architecture.

LSTM is a special variation of the Recurrent Neural Network (RNN). RNNs belong to a distinctive type of ANNs designed to model problems with a sequential or time-related aspect. RNNs can "memorize" by preserving an internal state using feedback loops. These loops occur in the network's neural units; thus, they are called recurrent neural units. In contrast to standard ANNs, a recurrent neuron has a connection to itself, making it possible to use past outputs.

When RNNs were first proposed (Hopfield, 1982), their popularity was short-lived because of the vanishing gradient problem occurring during ANN training (Bengio et al., 1994 and Gers et al., 2000), which is even more frequent with RNNs. LSTM was proposed to overcome this problem (Hochreiter & Schmidhuber, 1997). The overall control flow of an LSTM is similar to an RNN, differing only in the repeating module, i.e., the recurrent unit's internal processes..

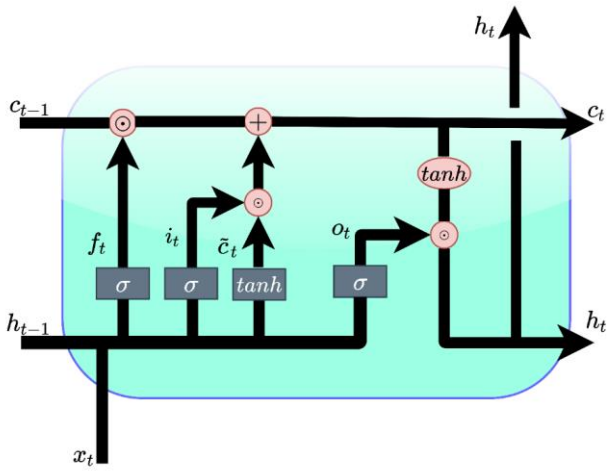

Figure 1. Recurrent processes in an LSTM unit. The grey rectangles refer to the recurrent activations.

An LSTM unit has three main concepts (*Figure 1*):

- The cell state $\mathbf{c}_t$, consists of an encoded version of the information gathered from all the previously processed timesteps. As a result, this term is also called the long-term state;
- The block output $\boldsymbol{h}_t$, is similar to the cell state, but it only holds information from the previous timestep. Thus, it is also called the short-term state;
- The gating mechanisms are called input gate ($\boldsymbol{i}_t$), output gate ($\boldsymbol{o}_t$) and forget gate ($\boldsymbol{f}_t$). The input gate determines what information from the candidate memory $(\tilde{\boldsymbol{c}}_t)$ should be appended to the cell state. The output gate controls which values from the cell state ($\mathbf{c}_t$) should be outputted within $\boldsymbol{h}_t$. The forget gate ($\boldsymbol{f}_t$) determines what information should be removed or "forgotten" from the cell state ($\mathbf{c}_t$).

In this study, the input is a vector $\boldsymbol{x}_t$ defined as,

$$\boldsymbol{x}_t = (x(j),\ x(j-1), \dots,\ x(j-p+1))^T, \quad (1)$$

formed by embedding p samples using a sliding window on a TS $x(j)$ with $j \in 0, \dots, J$, J being the number of training records of $x(t)$. The LSTM unit can be expressed following an intuitive order of operations, 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 block output $\boldsymbol{h}_{t-1}$, and $\boldsymbol{W}$ and $\boldsymbol{R}$ refer to the matrices of weights of the connections. The forget gate can be formally written as

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{R}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}\right), \quad (2)$$

The symbol σ refers to a specific activation function, e,g, a sigmoid, 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 or added.

In the next step, the unit decides what information is to be appended to the cell state ($\boldsymbol{c}_t$). First, the input gate ($\boldsymbol{i}_t$) decides which values should be updated, and second the new candidate values $(\tilde{\boldsymbol{c}}_t)$ that should be appended to the cell state are created. This is formally written as

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{R}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right) \quad (3)$$

Commented [PG3]:

Commented [JM4]: A função de ativação não é uma escolha que se pode fazer, e não existem outras opções para além da sigmoid e tanh, por exemplo, GRU?

Commented [PG5R4]: O bloco por defeito apresenta essas funções de ativação, na prática podem-se trocar mas esas é a definição, eu menciono isso na config do modelo

Commented [PG6]: uma forma boa para inserir equações mais complexas em latex para word: 1 – colocar aqui o latex http://www.msumathonline.com/LatexConverter/
2- copiar o MathML e colar no ficheiro word

$$\tilde{\boldsymbol{c}}_t = tanh\left(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{R}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}\right), \quad (4)$$

where $tanh$ an activation function (hyperbolic tangent) and transforms all values into $]-1,1[$. After this, the cell state is updated from $\boldsymbol{c}_{(t-1)}$ to $\boldsymbol{c}_{(t)}$, where the $\odot$ operator represents the element-wise product,

$$\boldsymbol{c}_t = tanh(\boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t), \quad (5)$$

The last steps assemble the unit's output. First, the output gate decides through the sigmoid function which values of the updated cell state should be carried on in the block output $\boldsymbol{h}_t$. Second, the element-wise product of the output gate and the updated cell state resulting vectors is calculated. This is written as,

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

$$\hat{\boldsymbol{y}}_t = \boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{c}_t, \quad (7)$$

where, in this context, the block output $\boldsymbol{h}_t$ becomes the model's output, i.e., the forecast $\hat{\boldsymbol{y}}_t$ at the last processing step. Additional details are presented in section 4.1.2 .

The training of standard ANNs is done using the backpropagation algorithm. However, for a RNN such as the LSTM an extension is necessary. For that goal, the backpropagation through time (BPTT) was proposed in (Gers et al., 2000). Fundamentally, it is processed as the backpropagation algorithm but while considering every timestep t.

2.2 Prophet

TS data in business problems tend to hold several common features and follow specific patterns, making it practical to develop specialized software packages for the sector. This is the case of Prophet, a free software package provided by Facebook to study time series data. Besides TS forecasting, it also includes visualization tools and cross-validation facilities to measure forecasting errors using historical data.

Prophet uses a generalized additive model (Hastie & Tibshirani, 1987) to fit the time series. Therefore, the model is described by additive components representing trend ($g(t)$), seasonality ($s(t)$) and holidays ($h(t)$):

$$x(t) = g(t) + s(t) + h(t) + \epsilon(t) \quad (8)$$

where $\epsilon(t)$ denotes the residues or any properties that are not fitted in the model.

Commented [CJ7]: alterar para residues

2.2.1Trend Model

Prophet implements two trend models: a linear model and a nonlinear saturating growth model. Both models use a piecewise strategy by incorporating changepoints, i.e., points where the growth rate can change. Changepoints are defined as $(s_j, \delta_j), j = 1, 2, \cdots S$, where δ_j is the change in the growth rate at s_j. The vector $\boldsymbol{\delta}$ denotes all growth rate changes of the time series. Thus, the growth rate at any time t is the sum of all growth rate changes before t, calculated as follows. Let $\boldsymbol{a}(t)$ denote a vector with binary entries $\{0,1\}$ such that $a_j(t) = 1$, if $t \geq s_j$, otherwise $a_j(t) = 0$. The growth rate at time t is $k + \boldsymbol{a}(t)^T\boldsymbol{\delta}$, where k is the initial logistic growth rate.

Nonlinear saturating growth with changepoints model

In this case, trend is a logistic saturating growth function and so it has the following template:

$$f(t) = \frac{L}{1 + e^{-k(t-m)}} \tag{9}$$

where L is the maximum in the curve, k is the logistic growth rate and m is the value of t at the sigmoid's midpoint. The nonlinear saturating piecewise growth model is defined as:

$$g(t) = \frac{c(t)}{1 + e^{-(k+\boldsymbol{a}(t)^T\boldsymbol{\delta})\left(t-(m+\boldsymbol{a}(t)^T\boldsymbol{\gamma})\right)}} \tag{10}$$

where $\boldsymbol{c}(t)$ is the saturation or carrying capacity. It is a business-related parameter, e.g., market size, population or GDP, which can change over time. The vector $\boldsymbol{\gamma}$ is used to adjust the values of m (see Equation (11)) to achieve the continuity of the function at the changepoints as follows:

$$\gamma_j = \left(s_j - m - \sum_{l<j} \gamma_l\right)\left(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l\leq j} \delta_l}\right) \tag{11}$$

Linear with changepoints model

This model is used when the time series does not incorporate any saturating growth and the trend is defined as

$$g(t) = (k + \boldsymbol{a}(t)^T\boldsymbol{\delta})t + (m + \boldsymbol{a}(t)^\tau\boldsymbol{\gamma}) \tag{12}$$

where the entries of the adjustment vector are $\gamma_j = -s_j\delta_j$.

Changepoint selection

The changepoints can be assigned by a business expert, e.g., by entering the starting date of promotions, or calculated automatically from an initial grid of equally space changepoints defined over the whole history T of the time series. The number of changepoint candidates is 25 by default. The selection of changepoints is achieved using a prior $\delta_j \sim Laplace(0, \tau)$ in Equation (**Error! Reference source not found.**). This imposes a tradeoff between minimizing the Mean Squared Error (MSE) between the model and the data and setting as many δ_j to zero as possible. When τ tends to zero, the trend component becomes a non-piecewise model ($j = 1$).

Commented [PG8]: corrigir isto

2.2.2 Seasonality

Seasonality components are modeled using Fourier Series. The amplitudes of the sines and cosines are the parameters to be adjusted and the periods (hourly, daily, weekly, monthly, etc.) are adjusted by the users. Fitting the seasonality is defined as

$$s(t) = \sum_{k=1}^{K}\left(a_k \cos\left(\frac{2\pi kt}{P}\right) + b_k \sin\left(\frac{2\pi kt}{P}\right)\right) = \boldsymbol{X}(t)\boldsymbol{\beta}. \tag{13}$$

In this formulation, the sum is replaced by an algebraic formula where the sinusoidal function is written as entries of a row matrix $\boldsymbol{X}(t)$ and the column vector $\boldsymbol{\beta} = (a_1, a_2, \cdots a_K, b_1, b_2, \cdots, b_K)^T$. P is a user-defined parameter for seasonality. For instance, $P = 7$ means weekly, $P = 365.25$ means yearly, etc. The number of terms (K) should not be too large to avoid overfitting. The documentation of Prophet recommends using N =3 for weekly and N =10 for yearly seasonality.

2.2.3 Holidays and Events

Prophet allows one to configure a parameter with a list of past and future events that contemplate known holidays and local events affecting the business. The use of the holiday list in the model was designed to make the process as simple as possible. Let D_i be the set of past and future holidays. An indicator function is added that represents whether time t is during a holiday i, and it is assigned each holiday a parameter $\varkappa_i$

which is the corresponding change in the forecast. This is done in a similar way to seasonality, generating a matrix of regressors (Taylor & Letham, 2018).

$$\boldsymbol{Z}(t) = [1(t \in D_1), \cdots, (t \in D_L)] \tag{14}$$

and taking,

$$h(t) = \boldsymbol{Z}(t)\boldsymbol{\varkappa}. \tag{15}$$

Prophet also offers a means of including the effects for the days around a specific holiday. There are additional parameters to delimit the days surrounding a holiday and basically, each of these days are considered holidays themselves.
The trend uncertainty is estimated by assuming that the future will see the same average frequency and magnitude of rate changes seen in history. As projected forward, it will produce wide uncertainty intervals (Taylor & Letham, 2018).

2.3 Metrics

The evaluation of a quantitative forecast is usually treated as a measure of how accurate or how well a model fits or reproduces the data. For that, various statistical error measures that give different insights were created. Before detailing some of the existing metrics, a definition of error is important. Formally, the error is

$$e(t) = y(t) - \hat{y}(t) \tag{16}$$

where $y(t)$ is the observed value and $\hat{y}(t)$ the value forecasted both for the same time t. It is worth noting that this definition is allocated in a single-step forecast scenario, meaning that the forecasted value $\hat{y}(t)$ was built on observations before $y(t)$. Thus, $e(t)$ is a one-step forecast error.
However, forecasting methods usually produce forecasts with more than one step, i.e., feature a forecasting horizon $H > 1$. The following standard statistical accuracy metrics can be used:

$$\begin{aligned}
\text{Mean Error} &= \frac{1}{H}\sum_{t=1}^{H} e(t) \\
\text{Mean Absolute Error} &= \frac{1}{H}\sum_{t=1}^{H} |e(t)| \\
\text{Mean Square Error} &= \frac{1}{H}\sum_{t=1}^{H} e(t)^2 \\
\text{Root Mean Square Error} &= \sqrt{\frac{1}{H}\sum_{t=1}^{H} e(t)^2}
\end{aligned} \tag{17}$$

The mean forecast error is simply the average of the errors for all H timesteps. Usually, this metric is not very useful since it does not provide a clear error description when there are negative and positive values (they cancel each other out). However, this metric can suggest whether the model is consistently over forecasting or under forecasting. That is why the mean forecast error is also referred to as forecast bias. The other metrics circumvent this issue by molding the error to a positive value. Before averaging, the MAE

metric turns the error into a positive value by taking its absolute value, while both MSE and RMSE metrics achieve that through squaring.

A problem that is common to the observed metrics is that they are not unit-free, i.e., they depend on the scale of the given data. Therefore, alternative metrics are required to render possible the comparison of forecast performances across TSs with different units. With that goal, metrics that provide relative errors are the usual approach. The relative error is given as,

$$p(t) = \left(\frac{y(t) - \hat{y}(t)}{y(t)}\right) \times 100 \tag{18}$$

Due to their intuitive interpretations and high popularity, MAPE (Equation (19)) and the absolute error metrics MAE and RMSE are the chosen evaluation functions in this work.

$$\text{Mean Absolute Percentage Error } = \frac{1}{H}\sum_{t=1}^{H} |p(t)| \tag{19}$$

3. CASE STUDY

The case study consists in predicting the number of client entries in a retail store, by day and every 30 minutes, within a horizon (H) of 28 days. The TS consists of records with the number of clients who entered the store between 2020-03-18 and 2020-05-26, every 30 minutes (Figure 2).

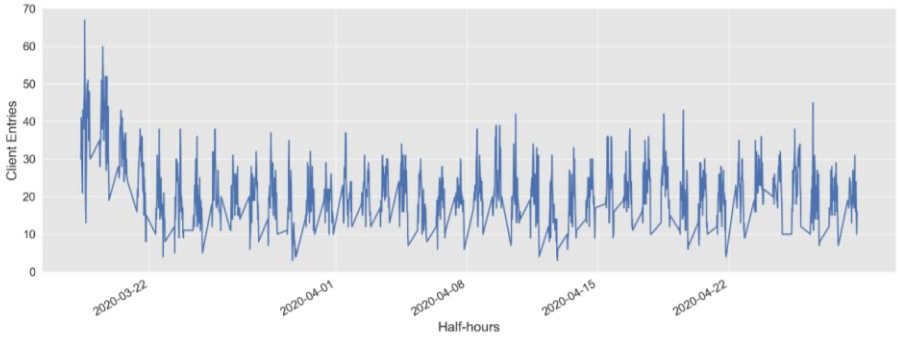

Figure 2. Number of client entries every half-hour.

During the first two days (March 18 and 19, 2020), the number of clients is greater than during the remaining days in the time series. This period coincides with the beginning of the lockdown due to COVID-19. The store is open 7 days a week and the small gaps in the TS represent the time intervals when the store is closed. A closer observation of the data showed that the opening schedule was not the same for all days of the week and that even for the same day of the week, there were changes during that time. To make the formulation of the problem easier, some samples were deleted to have the same schedule for all stores. In the new dataset, all samples are from 08:30 to 20:30. Figure 3 displays the average number of client entries by half-hour.

Commented [PG9]: Para já na parte com redes neuronais só tenho preparado os modelos para as receipts (entradas)

Commented [JM10R9]: Já tinha inserido um comentário abaixo prevendo a possibilidade de usarmos apenas uma das variáveis. A figura que o Clony inseriu abaixo mostra o nº de produtos vendidos, mas vamos usar as entradas.

Commented [JM11R9]:

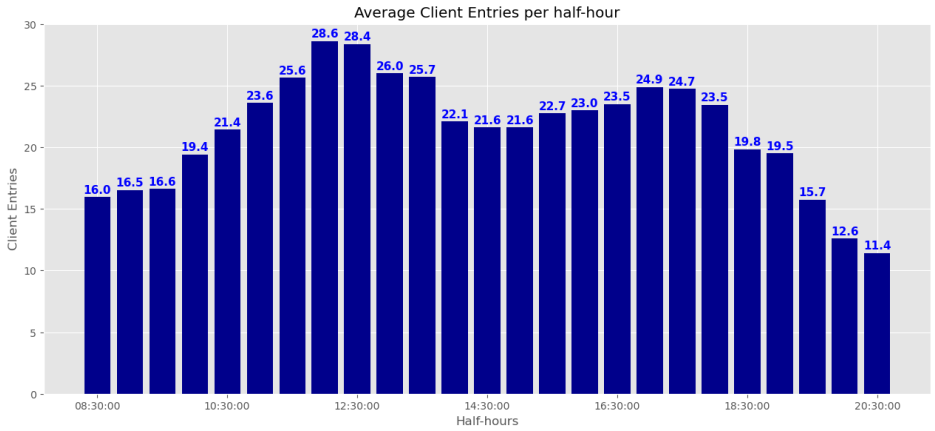

Figure 3. Average number of client entries in each half-hour.

Two datasets were used in this study, the original and a new one derived from this one, with the data aggregated by day. In both cases, the TS is split into train and test datasets. The training dataset has 1050 observations (42 days) and the test dataset has 700 observations (28 days). The first two days of the time series, where the number of clients differs from the usual due to the lockdown, are in the training set, influencing the forecasting results. Figure 4 shows the average number of clients per day of the week. The days that show less clients are at Mondays and Tuesdays, i.e., the beginning of the week. Thursday was the weekday where more clients entered the store.

Commented [JM12]: Clony: fazendo as contas dá 950 e não 1050, confirmas?

Commented [PG13R12]: Penso que é 1050 também, pelo menos foram as que utilizei n omeu treino

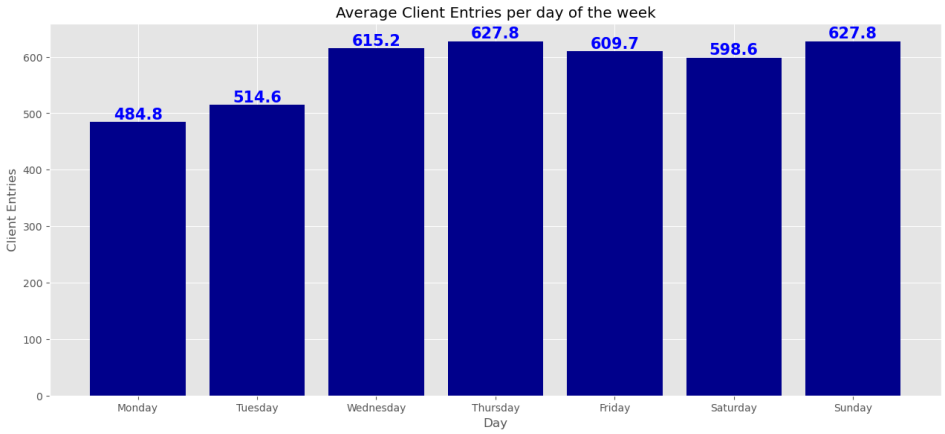

Figure 4. Average number of client entries in each day of the week.

In some forecasting methods, certain statistical qualities in TS can be crucial to create skillful forecasts. In classical models, some of those qualities are even necessary preconditions. Stationarity is one of those desired features (Oliver & Gujarati, 1993), as it provides statistical balance in the TS. As such, various methods to check were created. In this work, we used one of the most popular ones, the augmented Dickey-

Commented [PG14]: Remover provavelmente

Fuller (ADF) (Fuller, 1996), a unit root statistical hypothesis test. In Table 1 the test's results are presented. In both cases, the TSs are stationary and so it was not necessary to perform any data transformations towards that goal.

Table 1. The ADF test results.

	***t*-statistic**	***p*-value**	**Critical Value (1%)**	**Critical Value (5%)**	**Critical Value (10%)**
Half-hourly TS	-20.688	0.000	-3.434	-2.863	-2.568
Daily TS	-4.126	0.001	-3.539	-2.909	-2.592

Commented [JM15]: *Figure 5. Nº of Clients and Products solds, by day*
Confirmando-se que não vamos usar as vendas, substituir por uma figura só com o número de clientes e mantendo a ordem normal dos dias da semana: domingo, 2ª , 3ª, etc.

4. EXPERIMENTAL SETUP

TS forecasting is not an easy task. On the one hand, it requires in-depth knowledge of ML and statistical methods to use and optimize models of increasing complexity. On the other hand, domain knowledge is crucial for requirements elicitation, model selection, and the results' interpretation. Typically, ML experts do not know the business in-depth, and domain experts do not understand ML. Therefore, how to combine the knowledge of professionals with these two profiles is an interesting question, which still needs answers. This section presents the implementation of the case study using the two approaches studied in this chapter. The first, LSTM, required in-depth analysis to evaluate different configurations to fit the TS model. The second, Prophet, is already prepared to deal with this kind of problem and the main task is to enter the properties of the TS using special-purpose parameters.

Commented [PG16]: Transparecer melhor

4.1 LSTM

4.1.1 Technology and Libraries

The neural model and related operations were developed using Python 3.7.9. The following libraries were used:

- *Pandas*[1] was used to extract the data and to split the data into training and test sets.
- *Numpy*[2] provides efficient and optimized functions to deal with multidimensional arrays. This is important as these arrays are required for the network's operations.
- *TensorFlow*[3], 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.

[1] pandas.pydata.org
[2] numpy.org
[3] www.tensorflow.org

4.1.2 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 (observed) values. This adjustment is possible with the TS' past values and, as such, the data must be framed in that way.

In (Dietterich, 2002), a method to frame TSs as a supervised learning problem is presented. It is called the sliding window method (Figure 5) and it consists in transforming the data into several fixed-size combined input and output. Three variables are considered:

1. The stride s in which the windows are to be slid.
2. The number of timesteps of the inputs, i.e., the number of lagged values p in each window;
3. The number of timesteps of the outputs, i.e., the forecast horizons H in each window.

For the model's training phase, the width of the input $\boldsymbol{x}_t$ (Equation (1)) and the output are defined as,

$$\boldsymbol{y}_t = (x(j+1), \dots, x(j+H)), \quad (1)$$

Commented [PG17]: J training observations

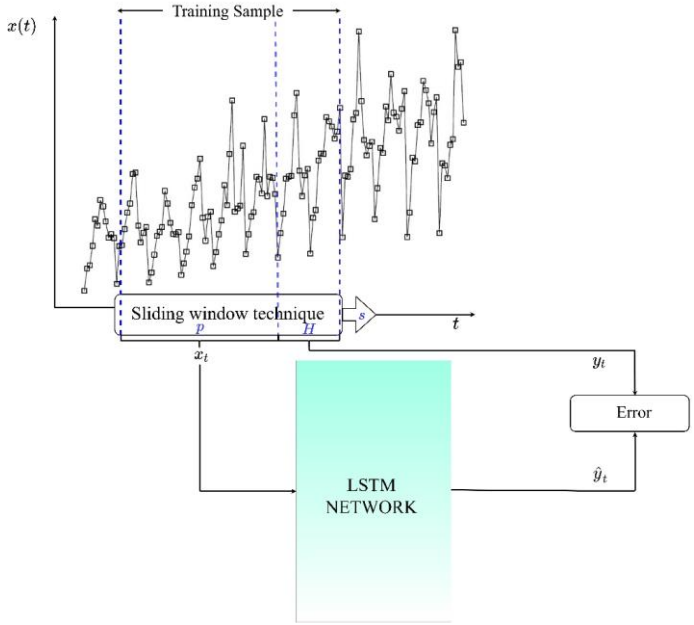

Figure 5. *The sliding window technique. Adapted from:* (Paoli et al., 2010)

It is worth noting that H and p are sensitive parameters. While windows with few lagged values may prove to have insufficient information, large windows can increase the task's complexity and reduce the network's learning capabilities. In that respect, since no optimization was done, various p values are crudely picked for the half-hourly and the daily 28 days prediction. As for the stride s, a value of 1 is chosen, for all times, in order to maximize the number of samples that are provided to the model.

4.1.3 Model Configuration

For all instances, the networks are trained for 30 epochs and are composed by two layers with the following configuration:

1. The input layer, which accepts windows of size p for every feature (in this case, only the variable of interest is used);
2. The LSTM layer has the hyperparameter *units* fixed at H. This parameter refers to the dimensionality of $\boldsymbol{h_t}$ that will be outputted. The other parameters use the default as defined by Keras[4] and no optimizations were done since the results are only of illustrative nature. As such, the activation functions (output and recurrent) are the same as in Figure 1.

Finally, for training purposes, the network uses MSE (Equation (1717) as the loss function and ADAM as ADAM as the gradient descent optimizer (Kingma & Ba, 2015).

4.2 Prophet

A two components model of Prophet were applied: linear trend (12) and additive seasonality (13). Both terms of the model were adapted using the default parameters except the one related with the strength of seasonal fluctuations (seasonality_prior_scale=0.7) and the one that modulates the trend's flexibility (changepoint_prior_scale). For the half-hourly TS, the latter was the only parameter changed (to 0.001), as its default value (0.05) is used in the daily example. The changepoint_prior_scale is the regularizer of the effects that changepoints cause in the TS. The remaining default parameters are the seasonality_mode (additive), the types (weekly and daily for half-hourly TS) seasonalities and the number of terms of the Fourier series (K=3 for both seasonalities).

Prophet needs to make a future data frame based upon parameters indicating the frequency and periods. Because of this, two TSs were required. For the daily TS, the forecasting horizon (the parameter period) is 28 days. In the other, the horizon is also 28 days, but it is also necessary to define the frequency of the predictions (freq='30min'). This information was added to the model after it being fitted with the training dataset.

Commented [CJ18]: referenciar eq 12 e 13 para citar a trend linear e additive seasonality

5. RESULTS

This section presents the results of applying the two approaches to the case study presented above. Note that the main objective of this work is to explain the methods behind each approach and not to optimize the models to obtain the best possible results. Therefore, the results of each approach are not compared to each other and are only illustrative.

5.1 Daily Predictions

5.1.1 LSTM

For the daily predictions ($H = 28$), the LSTM model could not learn any seasonality component. It looks as if only the trend was captured and, because of that, unskillful forecasts are obtained. Values of p equal to 1, 4 and 7 were tested, and the results f were very similar. Theoretically, using weekly windows (Figure 6) should have helped the network capture the weekly seasonality, but the results were equivalent or even a bit worse Table 2. A probable cause for this is the lack of available training samples as ANNs, such as the LSTM, are meant to be trained on large volumes of data.

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

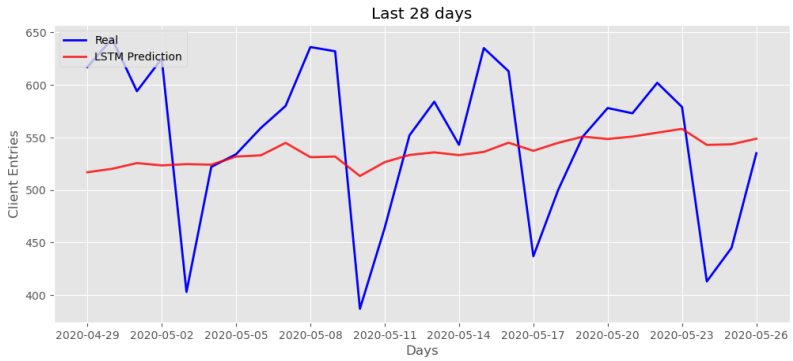

Figure 6. Daily LSTM predictions for the last 28 days using the sliding window approach with $p = 7$.

Table 2. Metrics for the daily LSTM prediction. Windows of one day offer best results.

Daily	**RMSE**	**MAE**	**MAPE(%)**
$p = 1$ (a day)	**74.280**	**58.602**	**11.776**
$p = 4$ (four days)	75.503	60.510	11.848
$p = 7$ (a week)	75.410	61.795	11.883

5.1.2 Prophet

The forecast presented in Figure 7 shows that Prophet can capture weekly seasonality. The performance results are displayed in Table 3.

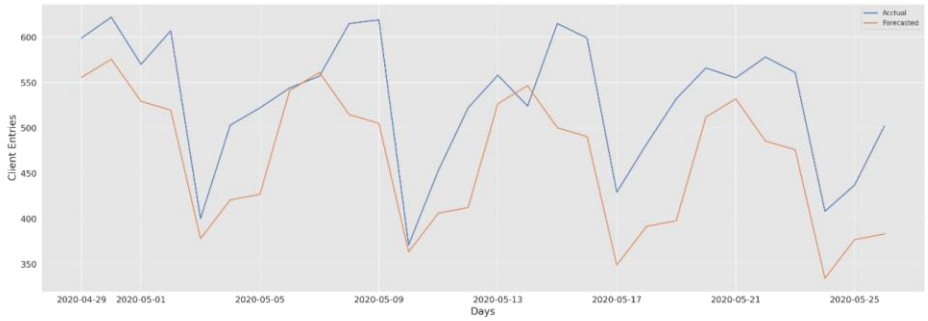

Figure 7. Daily Prophet predictions and seasonality component of the last 28 days.

Commented [CJ19]: Addicionar que na figura está a componente de sazonalidade

Table 3. Metrics for the daily Prophet prediction.

Prophet	RMSE	MAE	MAPE(%)
Daily	77.650	67.641	12.708

Prophet underestimated the number of clients who visited the store. The forecasts show a slightly decreasing trend possibly caused by clients' affluence above the usual during the first days in the training dataset. No change points were detected automatically and the weekly seasonality component is shown in Figure 8.

Figure 8. Weekly seasonality captured by Prophet.

5.2 Half-Hourly Predictions

5.2.1 LSTM

In the half-hourly TS ($H = 700$), the objective capturing weekly seasonality remains, yet, when dealing with half-hourly frequency, it is logical to consider daily seasonality as well. Multiple values of p up until 175 (weekly windows) were tested. The best results were obtained with $p = 50$ (Figure 9) as presented in Table 4. In contrast, Figure 10 shows the resulting predictions when the sliding window does not consider any seasonal period ($p = 10$).

Table 4. Metrics for the half-hourly LSTM prediction. Windows of two days offer best results.

Half-hourly	RMSE	MAE	MAPE(%)
$p = 10$ (five hours)	7.433	5.920	36.390
$p = 25$ (a day)	6.845	5.407	32.254
$p = 50$ (two days)	**6.450**	**5.097**	**30.101**
$p = 100$ (four days)	6.507	5.184	30.838
$p = 175$ (a week)	7.269	5.894	37.983

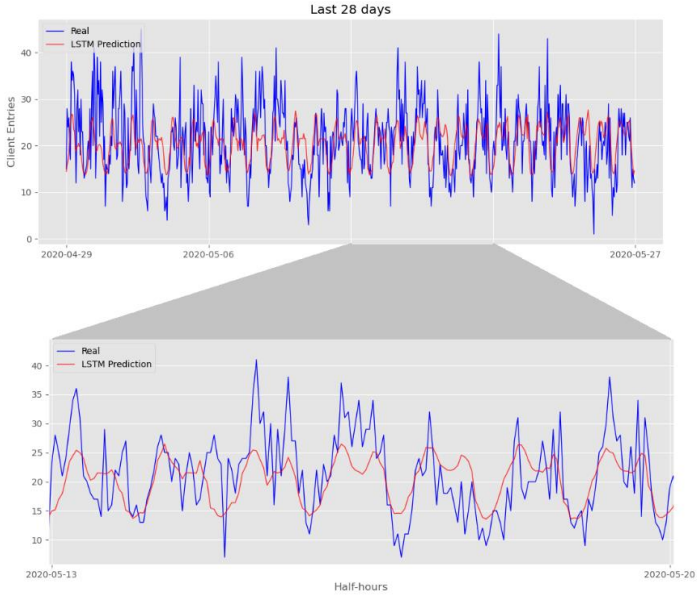

Figure 9. Half-hourly LSTM predictions of the last 28 days with $p = 50$ (third week zoomed in).

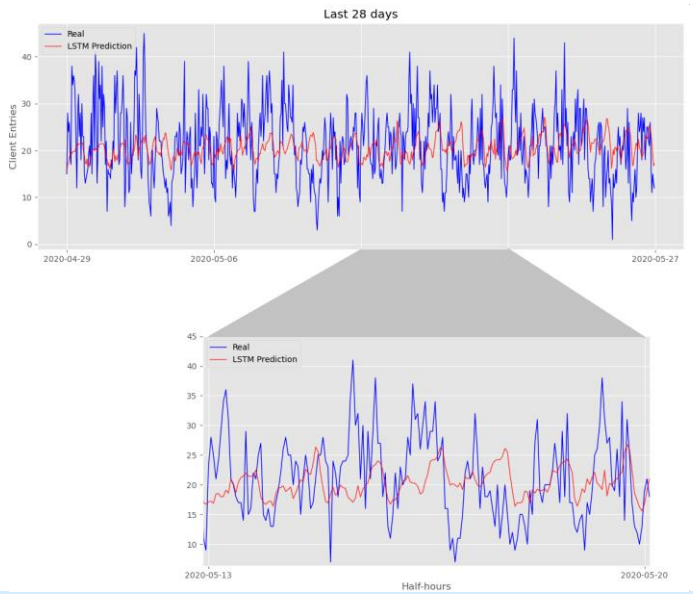

Figure 10. Half-hourly LSTM predictions of the last 28 days with $p = 10$ (third week zoomed in). As expected, five hourly windows are not suited for capturing daily seasonality.

Commented [JM20]: Esta figura não foi apresentada no texto. Temos duas hipóteses explica-la ou retirá-la.

Commented [PG21R20]: Está na discussão, como é um resultado não tinha a certeza se a deixava aqui ou colocava lá. Posso movê-la para lá e se calhar fica melhor.

Commented [PG22R20]: Comentei na discussão onde chamo esta figura.

Commented [JM23R20]: Como se trata de um resultado, acho que fica melhor nesta secção, mas convinha acrescentar 1 ou 2 frases a explicar e fazer referência à figura também nesta secção.

Commented [PG24R20]: Update

5.2.2 Prophet

Prophet's predictions are displayed in Figure 11 and the performance results in Table 5.

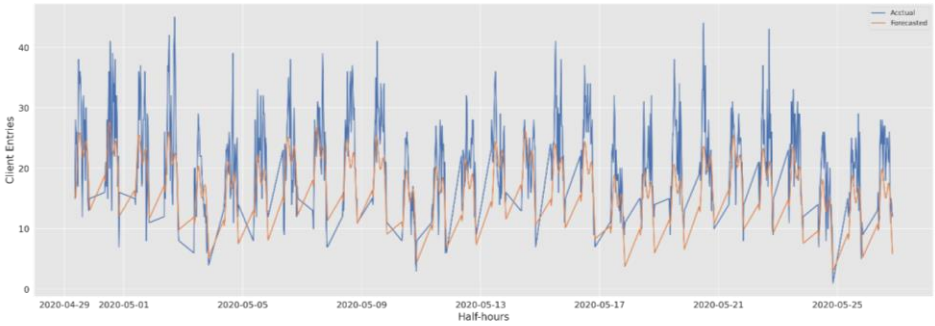

Figure 11. Prophet's half-hourly forecast against the real values.

Commented [PG25]: As previsões começam dia 29 e são de 28 dias, na imagem o eixo do X não me parece estar bem e a divisão pelos "riscos" também está um pouco disforme, nas previsões diárias também estão assim as "divisões"

Commented [CJ26R25]: Professor, revi a imagem e o dataframe utilizado para montar o gráfico, as datas estão corretas tanto no dataframe de teste quanto no previsto.

Table 5. Metrics for the half-hourly Prophet prediction.

Prophet	RMSE	MAE	MAPE %
Half-hourly	6.512	5.157	24.687

The gaps in the chart displaying the TS and the forecast correspond to the closing time. These gaps are not present in the charts in the previous sub-section, because LSTM handles the data as a sequence of values regardless of the timestamps. The captured daily seasonality is represented in Figure 12, which indicates a peak in the number of clients between 11 and 14 hours. These results can be compared with Figure 3.

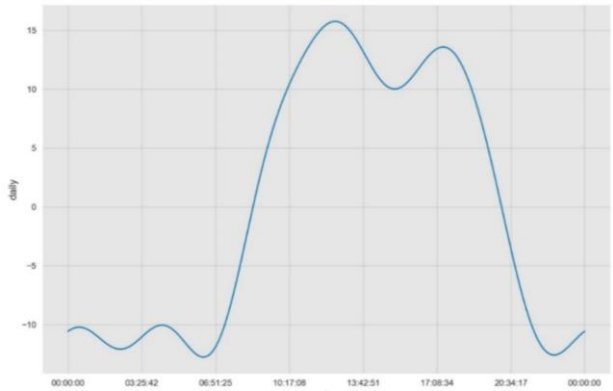

Figure 12. Prophet's daily seasonality.

5.3 Discussion

For the daily TS, it is evident that the sliding window configuration that offered more training samples to the LSTM model is the one that had better results. Nevertheless, results were not satisfactory as it is shown that the TS patterns were not captured. For small-size TSs such as this one, where DL approaches are unlikely to capture the seasonality, preprocessing methods to remove this component can improve performance.

For the half-hourly predictions, there are improvements in capturing seasonality. The lowest p values did not give the best performances but, surprisingly, neither a value of $p = 175$, which could have helped with the task of capturing the weekly seasonality.

With these results, it is possible to conclude that when modeling TS forecasting with LSTMs an equilibrium between p and the number of training samples should be achieved, especially for short TSs. Two additional concerns are valid when dealing with p:

1. The value should be small to lower the network's training complexity. Compared to $p = 50$, the model shows worse results when $p = 100$, meaning that larger windows do not necessarily mean better learning. This being related to the fact that, despite LSTMs having mitigated the problems of the traditional RNN, they still have limitations when maintaining lengthier long-term dependencies. For a shallow architecture as the one employed limitations may be accentuated;
2. It should also consider the existing seasonality periods to allow the model to learn the seasonality components that may exist in the data (Figure 10).

Prophet seeks to automate the forecasting process as much as possible and, for that purpose, many of its parameters are already preconfigured. Prophet activates weekly seasonality when there are 2 weeks or more of historical data and the gap between dates in historical data is less than 7 days, and activates daily seasonality if there are 2 days or more in historical data and the gap between dates is less than 1 day. Most of the parameters used the default configuration, except the changepoint_prior_scale on the half-hourly forecast, which allows less flexibility in the trend. Due to the higher temporal resolution and to improve the results, some changes in the parameters occurred on the half-hourly predictions.

Prophet proved to be an algorithm that fits these conditions and can be applied to TS along with several adjustments that may improve its accuracy. On the other hand, despite showing worse results, the LSTM model shows potential as the presented architecture is shallow. More layers can be stacked to the present model, enhancing its learning capability and possibly its predictive power. Another advantage of ANN models is that various attributes can be included in the model without constraints. However, a disadvantage is that it requires a larger dataset to capture the TS characteristics (this is obvious in the daily case study). For the LSTM approach, it is worth noting that even though the TS does not show a significant trend, the seasonality component was not dealt separately. Deseasonalization methods are usually employed as a preprocessing task for DL models (Makridakis et al., 2018).

6. CONCLUSION

This work deals with time series forecasting in the retail business, using two recent approaches. The first uses LSTM, a general-purpose method based on recurrent neural networks, which can be applied to various problems where modeling temporal behavior is necessary. The second uses Prophet, an open-source library released by Facebook and specialized in business forecasting problems.

This chapter describes the methods used in each approach, explaining, at the same time, some of the main concepts and problems in TS forecasting. A simple case study is used to forecast the number of clients in retail stores per day and every 30 minutes, using real-world data. The concepts, problems and solutions are presented by example.

LSTM is a type of recurrent neuronal network that can learn and memorize early patterns in a data sequence. It was not designed for a specific use case, so it is more flexible and can be applied to a broader range of problems than Prophet, but it is also more challenging to use and the learning curve is longer. The

Commented [PG27]: Questão: Não sei se é dito em algum lado que essas alterações foram algo "suaves" e que os resultados são apenas ilustrativos, com esta frase dá ideia que houve alguma "otimização", posso estar errado.

Commented [CJ28R27]: Não percebi

Commented [PG29R27]: Quando dizes que alteras os parametros para teres melhores resultados acho que se devia referir que é apenas alterações leves, outra coisa que notei neste parágrafo é que tu dizes que se teve que fazer mudanças e depois explicas cada parâmetro e não as mudanças que são feitas, posso estar a interpretar mal...

Commented [PG30]: Isto já não foi falado no 4.2?

Commented [PG31]: Sugestões de substituição e adição

Commented [PG32]: "which can be applied to various problems where modelling temporal behavior is necessary"

application of LSTM to a specific use case requires in-depth knowledge of the method and on machine learning in general.
Prophet is a specialized library for business forecasting problems. The methods are already preconfigured and adapted to this application domain, and it is often possible to obtain acceptable results, even when using the default parameters. It is easier to use than applying LSTM models, but it is also less flexible because there are predefined choices by the library's developers. It also does not require in-depth knowledge of machine learning so that domain experts can use it more easily. Although, a basic understanding of programming in Python is assumed.

Commented [JM33]: Tirar também?

Commented [PG34R33]: Eu faço uma referência sobre isso na discussão que acho que fica melhor lá, até porque estamos a discutir resultados, vou comentá-la com "Aqui"

In future work, the detection, resolution and anticipation of outliers could also take part in improving forecasting performance. For these objectives, Prophet already has some means at hand. Another concept that can be relevant for retail companies is the incorporation of contextual data in forecasting problems, such as modeling the influence of weather conditions, promotional events, region-specific behaviors and many other variables.

Commented [PG35]: Fiz outra versão do ultimo parágrafo, opiniões? Se acharem que está pior não há problema em apagar, fiz isto porque na minha opinião acho a construção do parágrafo como não sendo a melhor e que desviava um pouco o tema

7. KEY TERMS AND DEFINITIONS

Time Series Forecasting
Future estimates on a set of data that represents an event and tend to repeat itself with some periodicity as a function of time (time series), based on statistical rules applied to occurrences of the same event in the past.

Neural Networks
In artificial intelligence, they are computational systems with the aim of solving problems of this area. They are inspired by the functioning of brain biological neural networks and interconnected nodes (which are like the synapses on a biological brains) that uses a mathematical model to process information.

Deep Learning
It is a subclass of machine learn algorithms. It uses multiple layers of nodes (giving sense to the word "deep" on its name) to represent the learning process of the desired feature, splitting the learning process through the network layers.

Long Short-Term Memory (LSTM)
It is a type of recurrent neural network related to deep learning and, as such, it has connections with information feedback capable of register patterns for long duration of time.

Prophet
It is an open tool provided by Facebook to forecast time series data in a way that reduces, for analysts, the complexity found in forecasting models based on neural network algorithms, by preparing the data and encapsulating its underlying stan statistical model.

Machine Learning
It is an approach of computer science to study algorithms of patterns recognitions and learning process

using data. It builds models based on sample data and validated by comparison to real data.

Retail
Process of selling services or goods using different channels in order to satisfy consumer needs on identified demands.

ACKNOWLEDGMENTS

(…)

REFERENCES

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), 1–41. https://doi.org/https://doi.org/10.1016/j.heliyon.2018.e00938

Dietterich, T. G. (2002). Machine learning for sequential data: A review. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2396*, 15–30. https://doi.org/https://doi.org/10.1007/3-540-70659-3_2

Fuller, W. A. (1996). *Introduction to Statistical Time Series* (V. Barnet, R. Bradley, N. Fisher, J. Hunter, J. Kadane, & D. Kendall (eds.); 2nd ed.). Wiley-Interscience. https://doi.org/10.2307/2965736

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

Hastie, T., & Tibshirani, R. (1987). Generalized Additive Models: Some Applications. *Journal of the American Statistical Association*, *82*(398), 371–386. https://doi.org/10.1080/01621459.1987.10478440

İşlek, İ., & Gunduz Oguducu, S. (2017). A Decision Support System for Demand Forecasting based on Classifier Ensemble. *Federated Conference on Computer Science and Information Systems (ACSIS)*, *Vol 13*, 35–41. https://doi.org/10.15439/2017F224

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*. http://arxiv.org/abs/1412.6980

Kolarik, T., & Rudorfer, G. (1994). Time series forecasting using neural networks. *Proceedings of the International Conference on APL : The Language and Its Applications, APL 1994*, 86–94. https://doi.org/10.1145/190271.190290

Maaß, D., Spruit, M., & de Waal, P. (2014). Improving short-term demand forecasting for short-lifecycle consumer products with data mining techniques. *Decision Analytics*, *1*(1), 4. https://doi.org/10.1186/2193-8636-1-4

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*, *13*(3). https://doi.org/10.1371/journal.pone.0194889

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

Rosenblatt, F. (1957). *The perceptron - A perceiving and recognizing automaton* (Issues 85-460–1).

Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). A Comparison of ARIMA and LSTM in Forecasting Time Series. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

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

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick & F. Kozin (Eds.), *System Modeling and Optimization* (pp. 762–770). Springer Berlin Heidelberg.