Forecasting Hourly Retail Customer Flow on Intermittent Time Series with Multiple Seasonality

Martim Sousa[a,*], Ana Maria Tomé[a], José Moreira[a]

[a]*IEETA/DETI, University of Aveiro, Aveiro 3810-193, Portugal*

Abstract

In this article, we tackle a vaguely explored problem in the field of forecasting, which is multi-step ahead forecasting in retail customer flow time series. Retail time series often display complex seasonal patterns and intermittent behaviour. The foregoing features set up the conditions for a challenging forecasting problem. We provide the necessary theoretical background to attain competitive forecast accuracy and assess our methods in a large scale comparison along 98 retail time series, where week seasonal naïve is set as baseline. Specifically, we review strategies for multi-step ahead forecasting, models for time series with complex seasonal patterns and scale-independent performance measures for intermittent time series. In addition, we introduce a novel algorithm for outlier detection in seasonal time series, generalize seasonal naïve to multiple seasonal periods, and present a new single-output strategy for multi-step ahead forecasting in seasonal time series. Results demonstrate that our proposed ensemble model accomplishes eye-catching forecast accuracy.

Keywords: Multi-step ahead forecasting; Scale-independent performance measures; TBATS; Weighted average ensemble; Prophet;

1. Introduction

Workforce optimization is an activate research topic as it ensures non-negligible cost reductions in personnel for companies. Additionally, according to Portuguese law and many others, retail workers must be informed about their work schedule one month in advance. To do so, it is of

*Corresponding author
Email address: martimsousa@ua.pt (Martim Sousa)

utmost importance to provide high quality hourly customer entries forecasts for the upcoming month to accomplish proper shift planning. Even though work schedule allocation is not the goal of this article, for the sake of context, that is the final goal of our forecasting problem. To comply with the aforementioned demands, forecasting models have to provide hourly forecasts for the upcoming month, roughly 720 hours to forecast.

In this research, a real data set comprising 98 time series of customer entries is investigated which has information from January 2015 to November 2019. We propose a robust ensemble model, composed of well-known models in the literature, to outperform an established baseline model, namely week seasonal naïve. Note that the proposed ensemble model is not limited to our case study, it is adaptable to analogous problems, i.e., complex and intermittent time series with multiple seasonal periods. This might seem a straightforward task, yet week seasonal naïve is hard to beat, customer flow is highly seasonal in retail.

To the best of our knowledge, this is the first article addressing multi-step ahead hourly forecasts on intermittent time series with multiple seasonal periods in the retail sector. By intermittent we mean that, although there are customer entries when the store is open, there are no customer entries when the store is closed, thus being intermittent.

The outline of the paper is as follows. Section 2 reviews similar researches and introduces our contributions. Section 3 shortly describes several strategies to forecast in a multi-step ahead scenario, cutting-edge models to forecast in time series with complex patterns, and presents our innovations. Section 4 describes the data set and the suggested forecasting architecture, showcases results and discusses about the empirical evidence gathered in our research. Finally, Section 5 draws main conclusions, addresses shortcomings and also mentions future work directions.

2. Literature review

This research builds upon: (i) multi-step ahead forecasting strategies, (ii) models with multiple seasonal periods, (iii) scale-independent performance measures for intermittent time series, and (iv) customer flow forecasting.

2.1. Multi-step ahead forecasting strategies

Multi-step ahead forecasting is a challenging task and persists as an activate research topic. Unlike one-step ahead forecasting, multi-step ahead forecasting has to deal with accumulation of errors and growing uncertainty Sorjamaa, Hao, Reyhani, Ji & Lendasse (2007). The most basic and widely-used strategies to tackle multi-step ahead forecasting proposed in the literature are the recursive strategy, direct strategy, and MIMO (Multi-input Multi-output). Recursive is the most naïve strategy, it relies on a single-output model, where forecasts work as inputs recursively. For this reason, it suffers greatly from accumulation of errors. Hence, the direct strategy was devised in an attempt to circumvent accumulation of errors, however this strategy builds an independent model for each target, thus computational costs escalate rapidly and stochastic dependencies are lost. Several authors have been trying to combine these former two strategies, Sorjamaa & Lendasse (2006) showed that DirRec outperformed recursive and direct strategy when using a K-NN (K-Nearest Neighbors) model in two benchmark datasets. Later, Ben Taieb & Hyndman (2012) presented the rectify strategy, this strategy uses the recursive strategy to produce forecasts, subsequently forecast errors are modelled using a direct strategy. Their empirical study, evaluated in the data used for M3 and the NN5 competitions, demonstrated the superiority of rectify over direct and recursive strategy for a K-NN model. Despite these praiseworthy efforts, strategies reliant on multi-output models are generally superior Taieb, Bontempi, Atiya & Sorjamaa (2011); Xiong, Bao & Hu (2013). MIMO Bontempi (2008); Bontempi & Taieb (2011), previously known as joint strategy Kline (2004), effectively resolves the difficulties of single-output strategies, yet MIMO uses the same model structure to forecast each horizon values. This drawback led to the creation of DIRMO Taieb, Bontempi, Sorjamaa & Lendasse (2009). Nevertheless, there is no universal superior strategy. An & Anh (2015) study showed that DirRec was consistently better than other multi-output strategies, MIMO and DIRMO included, when using neural networks.

2.2. Models with multiple seasonal periods

SARIMA (Seasonal autoregressive integrated moving average) (Hyndman & Athanasopoulos, 2018, Chapter 8.9) and Holt-Winters Holt (2004); Winters (1960) are widely used for seasonal time series, yet they were designed to solely deal with a single seasonality. Taylor (2003) extended Holt-Winters exponential smoothing method to handle a second seasonal component. Despite these laudable improvements, several applications require models capable of handling multiple seasonal periods,

two different seasonal periods is not enough.

So far, TBATS, BATS and Prophet are the most notable models for multiple seasonality De Livera, Hyndman & Snyder (2011); Taylor & Letham (2018). TBATS is an acronym for key features of this model, namely trigonometric, Box-Cox, ARMA (Autoregressive integrated moving average), trend and seasonal components. BATS can be seen as a generalization of former exponential smoothing methods for multiple seasonality time series incorporating ARMA errors corrections and Box-Cox transformation. In spite of resolving multiple seasonality issue, BATS uniquely accepts integer seasonal periods. TBATS refines BATS by accepting non-integer seasonal periods. This is accomplished by means of trigonometric terms in a Fourier series manner. Likewise, Prophet utilizes Fourier series to manage non-integer seasonal periods, however Prophet has a component to model holidays or special events and behaves as a curve-fitting model rather than seeking to account for the temporal dependence structure in the data.

2.3. Scale-independent performance measures for intermittent time series

Hyndman (2006) published a thorough article about this subject.

Firstly, he addressed scale-dependent performance measures as MAE (Mean absolute error) or RMSE (Root mean squared error). These performance measures excel whenever comparing different models across a single time series, but they are hard to interpret and improper for multiple time series.

Secondly, percentage errors such as MAPE (Mean absolute percentage error) or sMAPE (Symmetric mean absolute percentage error), despite being scale-independent and intuitive, they either suffer from asymmetric behaviour or penalize over-forecasts differently than under-forecasts. In addition, MAPE and sMAPE are incompatible with intermittent time series as division by zero conducts to undefined or infinite values. MAAPE (Mean arctangent absolute percentage error) Kim & Kim (2016) is a percentage error performance measure cleverly tailored for intermittent time series, division by zero becomes no longer a problem, but previously discussed caveats from MAPE and sMAPE still apply.

Thirdly, Hyndman (2006) analyzed the usage of relative performance measures, i.e., the division between a scale-dependent performance measure for the testing model and a benchmark model. In case of a long forecast horizon relative performance measures offer barely no disadvantages,

since they are scale-independent, easy to interpret, symmetric, and penalize over-forecasts and under-forecasts equally. However, if the forecast horizon is narrow, relative measures can be misleading.

Finally, Hyndman (2006) advocated for the usage of MASE (Mean absolute scaled error) as the standard performance measure. MASE was inspired in relative performance measures, however MASE uses in-sample forecasts errors for the baseline model, making it an appropriate performance measure regardless of the forecast horizon dimension.

2.4. Customer flow forecasting

Thus far, few articles have been published regarding customer flow forecasting. Abrishami & Kumar (2018) used LSTM (Long short-term memory), RF (Random forest), MLP (Multilayer perceptron) and Prophet to forecast daily customer entries of over 65 business stores across the United States. They used a feature extraction technique described in Christ, Kempa-Liehr & Feindt (2016) that showed competitive results. Cortez, Matos, Pereira, Santos & Duque (2017) predicted daily customer flow for three classes (all faces, female and male) with historical data retrieved from a pilot project that used a digital camera and facial recognition system to detect foot traffic in a sports store. SVM (Support vector machines), Holt-Winters with a weekly seasonal period and an hybrid model combining both models were utilized to forecast those three classes. Ma & Fildes (2020) used third-party mobile payments to indirectly predict customer flow. Junior, Gusmão, Moreira & Tome (2021) used LSTM and Prophet to forecast hourly and half-hourly client entries for the next 28 days.

2.5. Novel contributions

This research brings several innovations: (i) A new multi-step ahead forecasting strategy for single-output models, which we call seasonal recursive; (ii) A model-free online seasonal z-score outlier detection algorithm; (iii) A generalization of seasonal naïve to handle multiple seasonal periods by averaging; (iv) An out-of-the-box robust ensemble model to enhance forecast accuracy. Moreover, after reviewing akin literature, we concluded that there is little knowledge on how to achieve competitive forecast accuracy in time series with complex seasonal patterns and intermittent behaviour. In addition, the performance measures used by many practitioners demonstrate theoretical

and practical problems, conducting to misleading conclusions regarding model selection. This article aims to fill these gaps. To attain such an objective, we gathered knowledge across various articles and undertook a large-scale empirical study on real data to compare different strategies and models in a practical point of view.

Note that innovations (ii) and (iii) can be applied to time series with multiple seasonal periods as long as the greater seasonal periods are multiples of the minimum one. For instance, if a hourly time series has daily, weekly, monthly and yearly seasonality, we can model them as multiples of 24 as 24, 168, 672 and 8736, respectively. Note that it is of key importance to not add seasonal periods that conflict with lower ones. For instance, a monthly seasonal period of $30.5 \times 24 = 732$ would annihilate a weekly seasonal period of 168.

3. Forecasting

This section shortly describes the pre-processing steps, training strategies, models and performance measures we have used in the experimental part as well as reviewing akin literature whenever appropriate.

3.1. Pre-processing

Prior to choose which models are the most appropriate, it is a good practice to gather insights about the data and, if advantageous, clean and transform them. In retail, it is well-known that during holidays, massive promotion campaigns, and close by events, customer flow soars. Other features, such as weather and macroeconomic dynamics may also have its influence in customer flow. However, these features are not always registered, particularly in small businesses. In fact, adding new features does not necessarily ensure better forecasts because the complexity of the underlying structure grows. These unusual number of arrivals, either upper or lower, must not be ignored, because feeding models with odd data often lead to inadequate forecasts. As a consequence, outliers, once detected, should be replaced by a more convenient value, using interpolation, mean value, model prediction or any other meaningful strategy.

3.1.1. Online seasonal z-score outlier detection

To the best of our knowledge, though the following algorithm is simple, this is the first article proposing this approach. This algorithm is model-free and is extremely effective for detecting

outliers on seasonal time series. Although not mandatory, we recommend to remove the trend from the time series first. The steps of this algorithm are depicted bellow.

Algorithm 1 Online seasonal z-score outlier detection

Input: $\{x_1, x_2, ..., x_N\}$ (Time series), $k \in \mathbb{N}$ (Number of previous of seasonal elements used for comparison), $s \in \mathbb{N}$ (Seasonal cycle length), $\alpha \in \mathbb{R}$ (Confidence region control parameter)

Output: O (Outlier index set)

$O = \emptyset$
$i \leftarrow ks + 1$
while $i \leq N$ **do**
  $j \leftarrow 1$
  $S = \emptyset$
  while $j \leq k$ **do**
    $S = S \cup \{x_{i-js}\}$
    $j \leftarrow j + 1$
  end while
  $\mu = mean(S)$
  $\sigma = std(S)$
  $U = \mu + \alpha\sigma$
  $L = \mu - \alpha\sigma$
  if $x_i > U$ or $x_i < L$ **then**
    $O = O \cup \{i\}$
  end if
  $i \leftarrow i + 1$
end while
return O

Once we have the Outlier index set O, we need to discard the associated time series values in favor of more appropriate ones. If $j \in O$, then $x_j \in \{x_1, x_2, ..., x_N\}$ is an outlier that should be replaced by x_j^* given by

$$x_j^* = \frac{1}{k} \sum_{i=1}^{k} x_{j-is}, \tag{1}$$

where s is the seasonal cycle length and k is the number of previous of seasonal elements used to calculate the new value. Clearly, s and k should be coherent with the ones used in the detection phase. Rather than replace the outlier by the mean of its seasonal lags, we can use more sophisticated strategies, such as single or double exponential smoothing or autoregressive processes to predict the former outlier. Note that this strategy can be generalized to multiple seasonal periods $s_1, s_2, ..., s_n$ with $k_1, k_2, ..., k_n$ seasonal lags, respectively. However, if our greater seasonal periods are multiples of the minimum one, a single seasonal period is enough provided that we set a big enough k. In the general case, μ and σ are replaced by

$$\mu^* = \frac{1}{n}\sum_{i=1}^{n}\mu_i, \tag{2}$$

$$\sigma^* = \frac{1}{n}\sum_{i=1}^{n}\sigma_i, \tag{3}$$

where μ_i and σ_i are the mean and standard deviation of the S set for s_i and k_i.

3.1.2. Min-max normalization

We easily understand that algorithms as K-means and K-NN require data normalization, otherwise a single feature could entirely dominate the optimization process when calculating the euclidean distance. Nevertheless, it is harder to understand why is it so important in models heavily reliant on derivative optimization such as neural networks. It stems from fundamentally empirical reasons as it ensures a more stable and faster convergence with higher probability of escaping local minimums. We present the so-called min-max normalization, which maps each element into the range [0,1] and is given by

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{4}$$

3.2. Multi-step ahead forecasting strategies

Several forecasting experts aim their research to the so-called model-centric approach, anyhow one shall not neglect the potential of an adequate training strategy. Recalling our intent, we have to produce hourly forecasts for a whole month, i.e., we are before a multi-step ahead forecasting problem. We have to predict the next H values $[x_N+1, ..., x_N+H]$ based on past data $[x_1, x_2, ..., x_N]$. There are several ways to achieve this purpose based on different methods of converting a univariate time series into a supervised problem suitable for machine learning models Bontempi, Ben Taieb & Le Borgne (2013); Taieb et al. (2011).

3.2.1. Recursive strategy

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a single-output model whose output corresponds to the next hourly forecast based on d preceding values, also known as lags. Considering a forecast horizon of dimension H, the forecasts are given by

$$\hat{x}_{N+h} = \begin{cases} f(x_N, ...x_{N-d+1}) & \text{for } h = 1, \\ f(\hat{x}_{N+h-1}, ..., \hat{x}_{N+1}, x_N, ..., x_{N-d+h}) & \text{for } h \in \{2, ..., d\}, \\ f(\hat{x}_{N+h-1}, ...\hat{x}_{N+h-d}) & \text{for } h \in \{d+1, ..., H\}. \end{cases}$$

Looking to the previous equations, we grasp how this strategy works. We get the first forecast, $\hat{x}_{N+1} = f(x_N, ...x_{N-d+1})$, then we use it as an input to get the second forecast, $\hat{x}_{N+2} = f(\hat{x}_{N+1}, ...x_{N-d+2})$, and so on. We repeat this process until we have forecasts for the entire horizon. Although very intuitive, this strategy pitfall is obvious as it includes previous forecasts, instead of actual values, as inputs for the following model prediction. Consequently, due to forecast uncertainty we are passing past forecast errors to the next forecast, accumulating errors as we forecast. Note that when $h > d$ we no longer have any actual value as input, only forecasts, hence this strategy might not be worthy either in the case of a large H or small d.

3.2.2. Seasonal recursive strategy

We can make a slight modification on the previous training strategy to make the model reliable for seasonal time series by including k seasonal lags , i.e. $\hat{x}_{N+1} = f(x_N, ...x_{N-d+1}, x_{N-ks+1},$ $x_{N-(k-1)s+1}, ..., x_{N-s+1})$, with s being the seasonal cycle length. The way seasonal recursive works, is just as recursive, each forecast incorporates the input for the next one. The main difference is the addition of non-consecutive lags with a regular space between them of s periods. Since for most applications $ks \gg H$, there are many actual values for nearly every $h \in \{1, 2, ..., H\}$, in consequence, this strategy is not only suitable for seasonal time series, but also accumulation errors are reduced, which imply lower forecast uncertainty. Note that, in the same way as our proposed outlier detection algorithm, this strategy can be generalized to multiple seasonal periods $s_1, s_2, ..., s_n$ with $k_1, k_2, ..., k_n$ seasonal lags, respectively. In addition, if our greater seasonal periods are multiples of the minimum one, a single seasonal period is enough for a big enough k.

3.2.3. MIMO strategy

MIMO stands for Multi-input Multi-output, therefore $F : \mathbb{R}^d \rightarrow \mathbb{R}^H$. This way, there are no accumulation errors and the forecasts are returned in a single step as $[\hat{x}_{N+H}, ..., \hat{x}_{N+1}] = F(x_N, ..., x_{N-d+1})$. The major drawback of this approach is the fact that most machine learning models do not allow multi-output naturally. The turnaround is to train H single-output models, one for each target, via direct strategy Ben Taieb & Hyndman (2012). Nevertheless, in the direct strategy the stochastic dependence among forecasted values is not preserved and training H models might be computationally onerous.

3.3. Models

Choosing the correct model is another important step along with the right training strategy. In this subsection, we briefly review a multitude of models for time series forecasting, including ones with multiple seasonal periods, tree-based and deep learning models.

3.3.1. Seasonal naïve

Seasonal naïve is a simple forecast method for seasonal time series, we simply set the forecast to be equal to the last seasonal observed value. Formally, for a horizon H, forecasts are given by

$$\hat{x}_{N+h} = \begin{cases} x_{N+h-s} & \text{for } h \in \{1, 2, ..., s\}, \\ \hat{x}_{N+h-s} & \text{for } h \in \{s+1, ..., H\}. \end{cases}$$

In spite of being naïve, this method can outperform other complex methods, especially in highly seasonal time series with no trend.

3.3.2. Multiple seasonal naïve

We can extend seasonal naïve to handle multiple seasonalities by applying seasonal naïve multiple times to different seasonal cycle lengths and average each forecast.

3.3.3. Multilayer perceptron

MLP is an extremely powerful model, since it is an universal approximator Hornik, Stinchcombe & White (1989). Moreover, MLP is suitable for MIMO strategy, i.e., approximate $F : \mathbb{R}^d \rightarrow \mathbb{R}^H$. $\hat{F}$, the approximation of F, is a composition of functions given by

$$\hat{F}(X) = f^{L+1}(f^L(...(f^2(f^1(W_1 X + \boldsymbol{b_1})W_2 + \boldsymbol{b_2}))...)W_{L+1} + \boldsymbol{b_{L+1}}), \tag{5}$$

where $X = \left[x_N, ..., x_{N-d+1}\right]^T \in \mathbb{R}^d$ is the input data, $f^i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is the vector activation function which gives the ith layer output, n_i being the number of neurons of the ith layer and $\boldsymbol{b_i} \in \mathbb{R}^{n_i}$ is a bias vector between layer i-1 and i. Finally, $W_i \in \mathbb{R}^{n_{i-1} \times n_i}$ is the weight matrix between layer i-1 and i and L is the number of hidden layers. Note that the vector activation function is composed by non-linear single-output activation functions as $f^i(a_1^{(i)}, a_2^{(i)}, ..., a_{n_i}^{(i)}) = (f_1^i(a_1^{(i)}), f_2^i(a_2^{(i)}), ..., f_{n_i}^i(a_{n_i}^{(i)}))$. MLP network is trained with an optimization algorithm such as Adam Kingma & Ba (2014), combined with backpropagation Rojas (1996) to compute the error gradients efficiently of a certain cost function. For further details we refer to (Goodfellow, Bengio & Courville, 2016, Chapter 6). The whole horizon H is forecasted in a single step as $[\hat{x}_{N+H}, ..., \hat{x}_{N+1}] = \hat{F}(x_N, ..., x_{N-d+1})$.

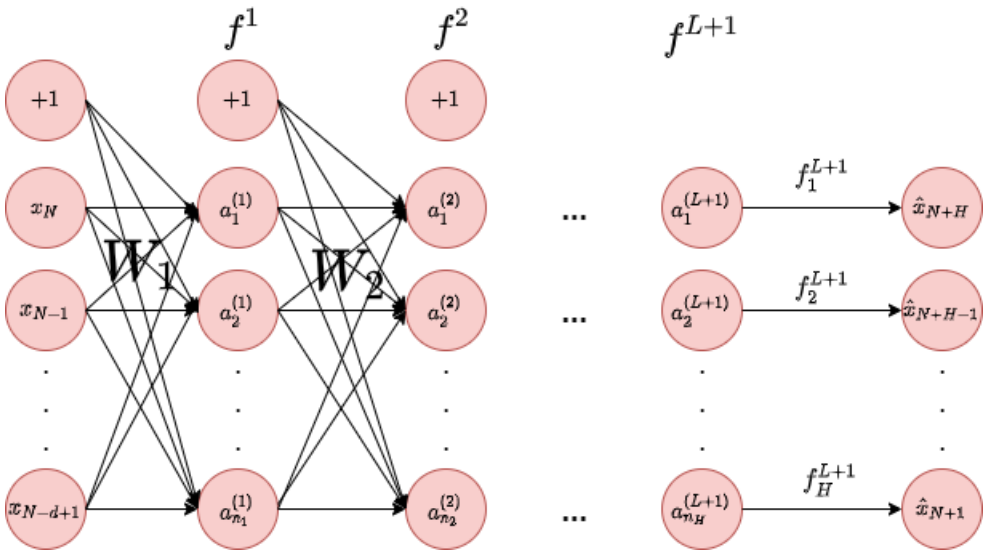

Figure 1: MLP architecture.

3.3.4. Long short-term memory

Hochreiter & Schmidhuber (1997) developed a new deep learning architecture named LSTM, in 1997, capable of capturing long time dependencies and mitigate vanishing gradient problem. Two common issues among preceding RNNs (Recursive neural networks). This architecture is suitable for MIMO strategy just as MLP, however LSTM was specifically designed for sequence data. In other words, LSTM is able to capture dependencies across samples, whereas MLP considers they are independent. Nonetheless, in univariate time series, the benefits of LSTM over MLP are

overshadowed.

Each LSTM cell has four inner components: input gate, forget gate, output gate, and memory cell, denoted by $i, f, o, \widetilde{c}$, respectively, depicted in Fig.(2). These inner components learn what to keep from previous LSTM cells and what to forget. They are updated with the current time step $\boldsymbol{x_t}$ and last hidden state $\boldsymbol{h_{t-1}}$ as follows

$$\boldsymbol{f_t} = \sigma(W_f \boldsymbol{x_t} + U_f \boldsymbol{h_{t-1}} + \boldsymbol{b_f}), \tag{6}$$

$$\boldsymbol{i_t} = \sigma(W_i \boldsymbol{x_t} + U_i \boldsymbol{h_{t-1}} + \boldsymbol{b_i}), \tag{7}$$

$$\boldsymbol{o_t} = \sigma(W_o \boldsymbol{x_t} + U_o \boldsymbol{h_{t-1}} + \boldsymbol{b_o}), \tag{8}$$

$$\widetilde{\boldsymbol{c}}_{\boldsymbol{t}} = tanh(W_c \boldsymbol{x_t} + U_c \boldsymbol{h_{t-1}} + \boldsymbol{b_c}), \tag{9}$$

where W_f, W_i, W_o, W_c are weight matrices between the current time step input and forget gate, input gate, output gate and memory cell, respectively. Likewise, U connects the previous hidden state with the gates and memory cell. $\boldsymbol{b_f}, \boldsymbol{b_i}, \boldsymbol{b_o}, \boldsymbol{b_c}$ are bias vectors w.r.t to gates and memory cell. Thereafter, we can calculate the next cell state, $\boldsymbol{c_t}$, and hidden state, $\boldsymbol{h_t}$, with the following equations

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \widetilde{c}_t, \tag{10}$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot tanh(\boldsymbol{c_t}), \tag{11}$$

where $\odot$ is the element-wise product. $\boldsymbol{h_t}$ captures short-time dependencies, whereas $\boldsymbol{c_t}$ is responsible for long-time dependencies. LSTM network is trained just as MLP yet it resorts to BPTT (Backpropagation through time) Werbos (1990) to compute the error gradients.

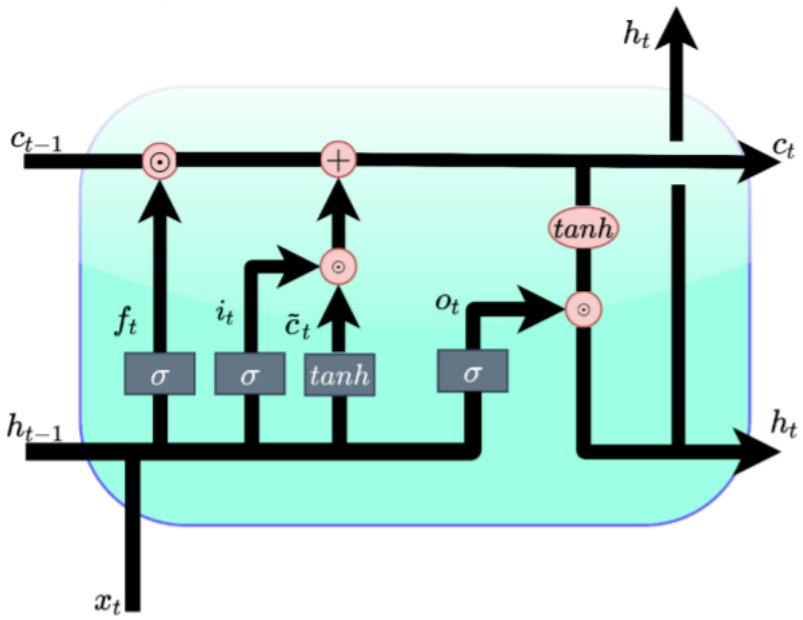

Figure 2: Recurrent processes in a LSTM cell. Adapted from Junior et al. (2021).

Dimensions.

- $\boldsymbol{x_t} \in \mathbb{R}^d$
- $\boldsymbol{f_t}, \boldsymbol{i_t}, \boldsymbol{o_t} \in (0,1)^h$
- $\boldsymbol{h_t}, \widetilde{\boldsymbol{c_t}} \in (-1,1)^h$
- $\boldsymbol{c_t} \in \mathbb{R}^h$
- $W \in \mathbb{R}^{h \times d}$
- $U \in \mathbb{R}^{h \times h}$
- $\boldsymbol{b} \in \mathbb{R}^h$

T, d and h refer to the number of time steps, the number of vector input features and the number of hidden state units, respectively. In the case of univariate time series, d=1. Note that a LSTM model can have multiple LSTM layers, with each LSTM layer containing T LSTM cells.

3.3.5. Histogram-based gradient boosting trees

Boosting is an ensemble strategy which combines a set of weak learners into a strong learner to minimize training errors. Gradient boosting trees, as the name suggests, mixes gradient descent and boosting together, utilizing decision trees Suthaharan (2016) as weak learners.

Algorithm 2 Gradient boosting tree training

Input: $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$ (Dataset), $J(y_i, \hat{y}_i)$ (Cost function), K (number of decision trees), α (Shrinkage learning rate).

Begin with a naïve base learner such as $F_0(x) = \frac{1}{N}\sum_{i=1}^{N} y_i$

$j \leftarrow 1$

while $j \leq K$ **do**

$g(\boldsymbol{x_i}) = -\frac{\partial J(y_i, F_{j-1}(x))}{\partial F_{j-1}(x)}|_{x=\boldsymbol{x_i}}$ for all $\boldsymbol{x_i} \in \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_N}\}$

Fit a decision tree h_j in residuals data $\{(\boldsymbol{x_i}, g(\boldsymbol{x_i}))\}_{i=1}^{N}$

$\gamma_j = \arg_\gamma \min \left(\sum_{i=1}^{N} J(y_i, F_{j-1}(\boldsymbol{x_i}) + \gamma h_j(\boldsymbol{x_i}))\right)$

Update the model: $F_j(x) := F_{j-1}(x) + \gamma_j \alpha h_j(x)$

$j \leftarrow j + 1$

end while

return F_j

Gradient boosting is known for its extremely efficiency among tabular data, however, training multiple decision trees is computationally expensive, particularly for big data sets with tens of thousands of samples and hundreds of features. In a dataset with N samples and D features each decision tree has to sort all columns, which has a time-complexity of $O(DN\log(N))$, then, while building a tree, each feature has to be tested whenever a new branch is added, and each feature has at most N-1 split points. According to former calculations, it is reasonable to accept that as data set size grows gradient boosting rapidly becomes unfeasible to train. In consequence, it is common to accelerate the training by discretizing the input features to $B \ll N$ bins. This strategy for gradient boosting, known as histogram-based gradient boosting, vastly reduces the training time, since there are only B split points per feature and computing a histogram of B bins for N examples has a time-complexity of O(N+B). Histogram-based gradient boosting (HistGB) is not suitable for MIMO strategy, this means that we are left to chose between a recursive or direct

strategy whenever using this model for multi-step ahead forecasting.

3.3.6. Prophet

Prophet is an open source software released by Facebook's core data science team Taylor & Letham (2018). This model is based on an additive form given by

$$x(t) = g(t) + s(t) + h(t) + \epsilon_t, \tag{12}$$

where g(t) represents the trend function, s(t) the seasonality function, h(t) the holidays function and ϵ_t an error term representing the deviations not accommodated by the former components. The trend component can be modelled by a saturating growth model or a piecewise linear model, both having changepoints to alter the growth change. The seasonality function is modelled by a Fourier series and the holidays function is grounded on a matrix of regressors that learns the effect of each holiday. Prophet uses L-BFGS to find maximum a posteriori estimations. The prior is a normal distributions for $s(t)$ and $h(t)$ and Laplace distribution for changepoints. In a practical point of view, Prophet has several advantages: (i) seasonality of multiple non-integer periods, (ii) suitable for irregularly spaced time series, (iii) fast to train, (iv) confidence intervals for forecasts, and (v) ease of use for non-experts.

3.3.7. TBATS

TBATS is a powerful model as it allows multiple seasonal non-integer periods, trends, Box-Cox transformation to induce non-linearity and ARMA error correction. TBATS can be seen as a generalization of the former state space models, with Holt-Winters being the most widely used. In fact, ARMA error correction is of utmost importance. Previous exponential smoothing models assumed that the error stochastic process $\{d_t\}$ is serially uncorrelated, however Chatfield (1978), in an emprical study, demonstrated that the errors of a Holt-Winters method could be modelled by a AR(1) process. Modelling the errors by a ARMA process can then result in an improvement of forecast accuracy.

Before introducing TBATS equations, is is necessary to understand how BATS works. Note that BATS is just as TBATS but only allows multiple seasonal integer periods, i.e., there are no trigono-

metric terms. BATS can be formulated as

$$x_t^{(\omega)} = \begin{cases} \frac{x_t^{\omega-1}}{\omega}, & \omega \neq 0, \\ \log(x_t), & \omega = 0, \end{cases} \tag{13}$$

$$x_t^{(\omega)} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t, \tag{14}$$

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t, \tag{15}$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t, \tag{16}$$

$$s_t^{(i)} = s_{t-m_i}^{(i)} + \gamma_i d_t, \tag{17}$$

$$d_t = \sum_{i=1}^{p} \varphi d_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \tag{18}$$

where T is the number of seasonal cycles, $m_1, m_2, ..., m_T$ denote the seasonal periods, α, β and γ_i for $i \in \{1, 2, ..., T\}$ are smoothing parameters, l_t is the time series level at time step t, b is the long-run trend, b_t the short-run trend at time step t, $s_t^{(i)}$ the ith seasonal component at time step t, ϕ is a parameter for the damped trend, ϵ_t is a Gaussian white noise, $x_t^{(\omega)}$ denotes a Box-Cox transformation using parameter ω, and d_t, the error term at time step t, follows a ARMA(p,q) process. Note that if we represent the BATS model as BATS$(\omega, \phi, p, q, m_1, m_2, ...m_T)$, then BATS$(1, 1, 0, 0, m_1)$ denotes the well-known Holt-Winters.

As previously stated, BATS is only suitable for integer seasonality and has undesirably many states. TBATS overcomes this by resorting to a trigonometric representation of seasonal components based on Fourier series, see Eq.(19-21). For a thoroughly explanation about TBATS, including parameter estimation, we refer to De Livera et al. (2011).

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}, \tag{19}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos(\lambda_j^{(i)}) + s_{j,t-1}^{*(i)} \sin(\lambda_j^{(i)}) + \gamma_1^{(i)} d_t, \tag{20}$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \sin(\lambda_j^{(i)}) + s_{j,t-1}^{*(i)} \cos(\lambda_j^{(i)}) + \gamma_2^{(i)} d_t, \tag{21}$$

where $\gamma_1^{(i)}$ and $\gamma_2^{(i)}$ are smoothing parameters and $\lambda_j^{(i)} = \frac{2\pi j}{m_i}$.

3.3.8. Weighted average ensemble

Stacking Wolpert (1992) is a clever way to combine multiple models by means of a new model that taking model predictions as inputs and knowing the actual values learns how to combine the models to improve forecast accuracy. Nevertheless, stacking can lead to overfitting and, to capture long seasonalities, many predictions are required, inevitable conducting to a shortage of samples for training the models. In turn, we rather suggest combining the models with a weighted average where the contribution of each model to the final forecast is weighted by its performance as follows

$$\hat{x}_h = \sum_{i=1}^{K} w_i \hat{x}_{i,h} \quad h \in \{1, 2, ..., H\}, \tag{22}$$

where K is the number of models to combine, $\hat{x}_{i,h}$ denotes the h-step ahead forecast of the ith model, H is the forecast horizon and $w_i = \frac{ForecastAccuracy(Modeli)}{\sum_{j=1}^{K} ForecastAccuracy(Modelj)}$.

It is worth noting that weighted average ensemble (WAE) is not only used to boost forecast accuracy, but also because relying on a single model is dangerous. It can become inconsistent quickly. We intend to have a robust model to manage trend changes, seasonality changes, schedule changes and any other type of unexpected changes, WAE attempts to overcome these handicaps.

3.4. Performance measures

This section reviews common performance measures in time series forecasting and addresses its pros and coins. Recall that, in mathematics, a metric is a function that must hold identity of indiscernibles, symmetry and triangle inequality in its domain. Despite most of the performance measures used to assess time series forecasting models do not verify these axioms, many researchers roughly name them metrics anyway. We rather prefer to call it as performance measures.

RMSE and MAE are by far the most used performance indicators in regression tasks and are given as follows

$$RMSE = \sqrt{\frac{1}{H} \sum_{i=1}^{H} (x_i - \hat{x}_i)^2}, \tag{23}$$

$$MAE = \frac{1}{H} \sum_{i=1}^{H} |x_i - \hat{x}_i|. \tag{24}$$

RMSE can be seen as a normalized version of the 2-norm and as a standard deviation estimator for the error distribution. Likewise, MAE is a normalized version of the 1-norm. These two

performance indicators are easy to understand and consistent to compare different models in a single time series. However, since they are scale-dependent, they are improper to test models across multiple time series. Moreover, since they are unbounded, they give no clue about the effectiveness of a forecasting model.

Percentage measure errors such as MAPE are famous in businesses and organizations due to its scale-independent nature and interpretability. MAPE is expressed as

$$MAPE = \frac{100}{H} \sum_{i=1}^{H} \left| \frac{x_i - \hat{x}_i}{x_i} \right|. \tag{25}$$

Several authors have been discussing the adequacy of MAPE de Myttenaere, Golden, Grand & Rossi (2016); Tofallis (2015); Hyndman (2006). Looking at its formula, Eq.(25), the division by the actual value arises issues for intermittent series due to division by zero, also MAPE is lower bounded by 0, but has no upper bound. Further, this measure is not symmetric, for an actual value of 50 and forecast value of 30 $MAPE = 100 \times |\frac{50-30}{50}| = 40\%$, while for an actual value of 30 and forecast value of 50 $MAPE = 100 \times |\frac{30-50}{30}| = 66.(6)\%$. This example demonstrates how unreliable MAPE can be when comparing across multiple time series even when the same absolute error is kept.

Kim & Kim (2016) devised MAAPE, a good alternative measure performance of MAPE for intermittent time series, upper bounded by $\frac{100\pi}{2}$. MAAPE has some advantages over MAPE as it solves the undefined/infinity issue when we divide close to zero. In addition, MAAPE is more balanced to positive and negative errors. Nonetheless, MAAPE is less intuitive, still asymmetric, and penalizes less as absolute error grows.

Makridakis (1993) defined SMAPE as

$$SMAPE = \frac{200}{H} \sum_{i=1}^{H} \frac{|x_i - \hat{x}_i|}{|x_i + \hat{x}_i|}, \tag{26}$$

a modified version of the original ajusted MAPE Schnaars (1986). Indeed, this alternative is symmetric and makes division by zero less likely, also if actual values and forecast values are positive, 200 is the upper bound. In spite of being symmetric, SMAPE induces a new asymmetry, favoring models that over-forecast. Furthermore, SMAPE behaves poorly if either the actual value

or forecast value is 0 since SMAPE becomes 200 for that observation, thus not the best choice for intermittent time series.

Another performance measure is R-squared, initially conceived as a goodness-of-fit measure for linear regression models and interpreted as the percentage of variance in the dependent variable that the independent variables explain collectively. Still, its use as a symmetric and balanced scale-independent performance measure is growing and, according to some studies Chicco, Warrens & Jurman (2021), can be more informative than the foregoing performance measures. R-squared is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{H}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{H}(x_i - \mu)^2}, \tag{27}$$

where μ is the training sample mean. This measure is upper bounded by 1, but not lower bounded. The best case is when $R^2 = 1$. When $R^2 = 0$ it means that the tested model operates as good as a horizontal line equal to the mean, whereas $R^2 < 0$ means that a horizontal line performs better than the tested model. As a rule of thumb, a good model must have $R^2 > 0.6$.

Despite being more informative according to Chicco et al. (2021), we do not recommend R-squared as a general performance measure for time series forecasting, because time series are often non-stationary, conducting to a biased R^2 value, close to 1, due to a large variance, not necessarily as a result of a low total squared error.

Hyndman & Koehler (2006) recommended the usage of relative measures when there are multiple out-of-sample forecasts. Relative measures are scale-independent, symmetric, easily interpretable and rarely undefined or infinity even for intermittent time series. The RelMAE (Relative mean absolute error) is given by

$$RelMAE = \frac{MAE}{MAE_b}, \tag{28}$$

where MAE_b denotes the MAE of a benchmark model. Its interpretation is simple, $RelMAE > 1$ when the benchmark model is superior, on average, than the model being assessed and $RelMAE < 1$ when the model performs better, on average, than the benchmark model. Moreover, RelMAE is symmetric and penalizes positive and negative forecast errors equally. All these qualities convey us to strongly advocate for the general usage of RelMAE over other performance measures of

this section, whether it is across single or multiple time series. Additionally, we suggest naïve or seasonal naïve as benchmark models.

4. Results and discussion

This experimental study comprises 98 univariate time series of customer arrivals aggregated in timestamps of 15 minutes from January 2015 to November 2019. Fig.(3) encompasses each step from raw data to customer flow forecasting. Whenever the store is closed, we consider that there are 0 client entries in order to make our time series regularly spaced. November 2019 data was not used to train the models and so is used as a test set. November 2019 has 30 days, thus 720 hours to forecast, H=720. The experiments were conducted in Python. Table (1) displays each model and the respective Python library. Fig.(4) demonstrates the effectiveness of the novel online seasonal z-score outlier detection algorithm in our case study, with k=53, s=168 and $\alpha = 2$. It definitely helped to attain better forecast accuracy by capturing, on average, 2.25% observations as outliers. Each model was gauged across multiple time series using the following performance measures

$$WinRatio = \frac{\#victories}{98}, \tag{29}$$

$$RelMAE^* = \frac{1}{98}\sum_{i=1}^{98} RelMAE, \tag{30}$$

where week seasonal naïve, s=168, is used as benchmark for RelMAE.

Table (2) compares different seasonal naïve models. Results show that, as expected, multiple seasonal naïve yielded better results than any individual seasonal naïve. Regarding multi-step ahead forecasting strategies, results demonstrate that single-output strategies conveyed unsurprisingly unsatisfactory results, though our proposed seasonal recursive strategy outperformed, on average, the recursive strategy, see Table (3). MIMO under MLP architecture was the best model in terms of $RelMAE^*$ with an impressive score of 0.818, winning across 40 time series. On the contrary, LSTM proved itself unable to learn the underlying multi-output function with a $RelMAE^*$ almost as high as 3, see Table (4). Table (5) displays multiple seasonality models such as Prophet and TBATS performed reasonably, with a $RelMAE^*$ of roughly 1, which implies that they performed as good as week seasonal naïve. These results of multiple seasonality models might be surprising, however since we are using a relative measure, it does not necessarily mean that multiple seasonality

models perform disastrously. In fact, retail time series are known to be highly seasonal, especially weekly, thus week seasonal naïve already achieves extraordinarily good performance, making it an hard benchmark to overcome. Finally, just as expected, when we introduced WAE for the best 5 models it attained a $RelMAE^*$ of 0.742 and won across 55 time series, see Table (6). As a result, we advocate for the usage of ensemble models, instead of trying to find the best model and its hyper-parameters, not only it generally yields better forecast accuracy, but also makes the model more robust to sudden pattern changes. Note that non-specified model hyper-parameters indicate that default ones were used.

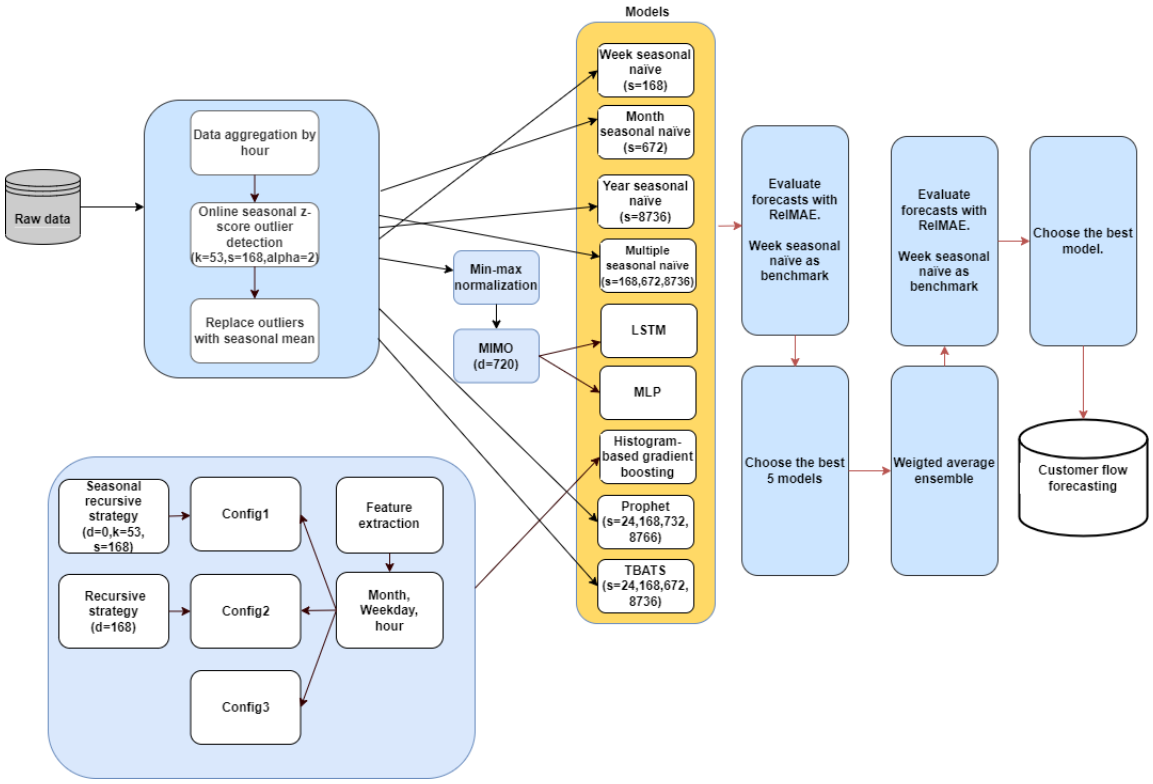

Figure 3: Flowchart for customer flow forecasting.

Model	Library	Documentation link
TBATS	tbats	tbats
Prophet	prophet	prophet
MLP	keras	Dense
LSTM	keras	LSTM
HistGB	sklearn	HistGB

Table 1: Models and their respective Python libraries.

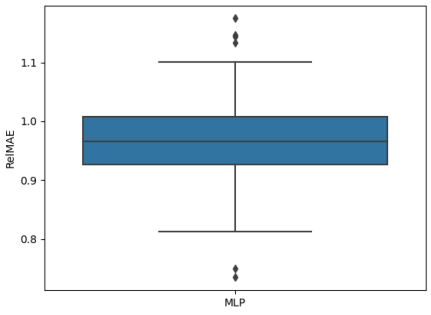

Figure 4: RelMAE of MIMO MLP with outlier treatment processed by online seasonal z-score outlier detection algorithm vs MIMO MLP with raw data.

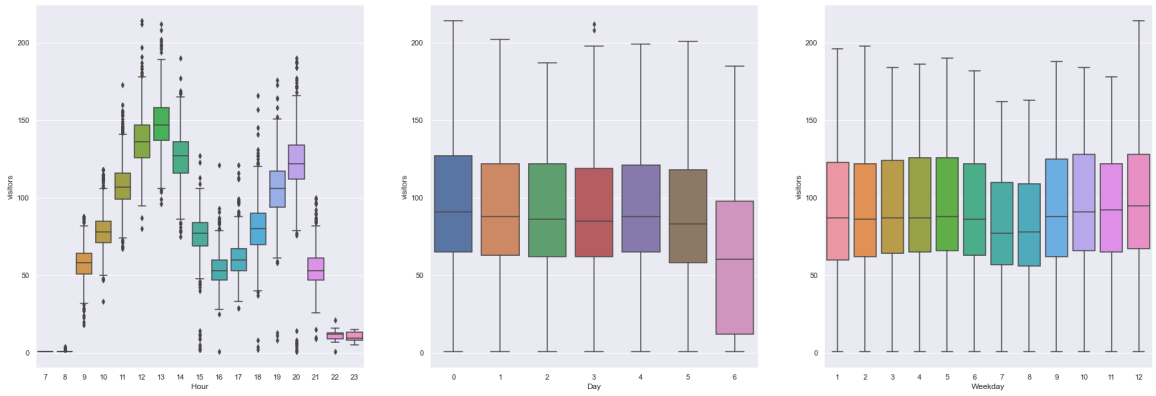

Figure 5: Customer flow by hour (panel left), weekday (panel middle) and month (panel right) for a store.

[1]The ith element represents the number of neurons in the ith hidden layer.

[2]Prophet API time unit is in days, instead of hours. Therefore, seasonal periods presented here are multiplied by 24.

Model	s	$RelMAE^*$	WinRatio
Week seasonal naïve	168	1	0
Month seasonal naïve	672	1.002	0
Year seasonal naïve	8736	1.129	0
Multiple seasonal naïve	(168,672,8736)	0.857	11/98

Table 2: Seasonal naïve models results.

Model	Features	Strategy	s	k	d	$RelMAE^*$	WinRatio
HistGB (1)	Day,Weekday,Month	N.A.	N.A.	N.A.	N.A.	0.891	45/98
HistGB (2)	Day,Weekday,Month	Recursive	N.A.	N.A.	168	3.461	0
HistGB (3)	Day,Weekday,Month	Seasonal recursive	168	53	0	2.012	0

Table 3: Recursive and Seasonal recursive strategy results.

Model	Hidden layers	Optimizer	Batch size	Epochs	Activation	Loss function	$RelMAE^*$	WinRatio
MLP	(1080,720,360) [1]	Adam	100	100	ReLU	MSE	0.818	40/98
LSTM	(100,100)	Adam	100	100	ReLU	MSE	2.966	0

Table 4: MIMO strategy models results.

Model	seasonal periods (s)	$RelMAE^*$	WinRatio
TBATS	(24,168,672,8736)	1.081	0
Prophet [2]	(24,168,732,8766)	0.995	2/98

Table 5: Multiple seasonality models results.

The previous table results did not include WAE, we consider the forecast accuracy of a model as $\frac{1}{RelMAE}$.

Model	$RelMAE^*$	WinRatio
WAE	0.742	55/98
TBATS	1.081	0
Prophet	0.995	0
MLP	0.818	15/98
LSTM	2.906	0
HistGB (1)	0.891	28/95
HistGB (2)	3.461	0
HistGB (3)	2.012	0
Week seasonal naïve	1	0
Month seasonal naïve	1.002	0
Year seasonal naïve	1.129	0
Multiple seasonal naïve	0.857	0

Table 6: Results with WAE.

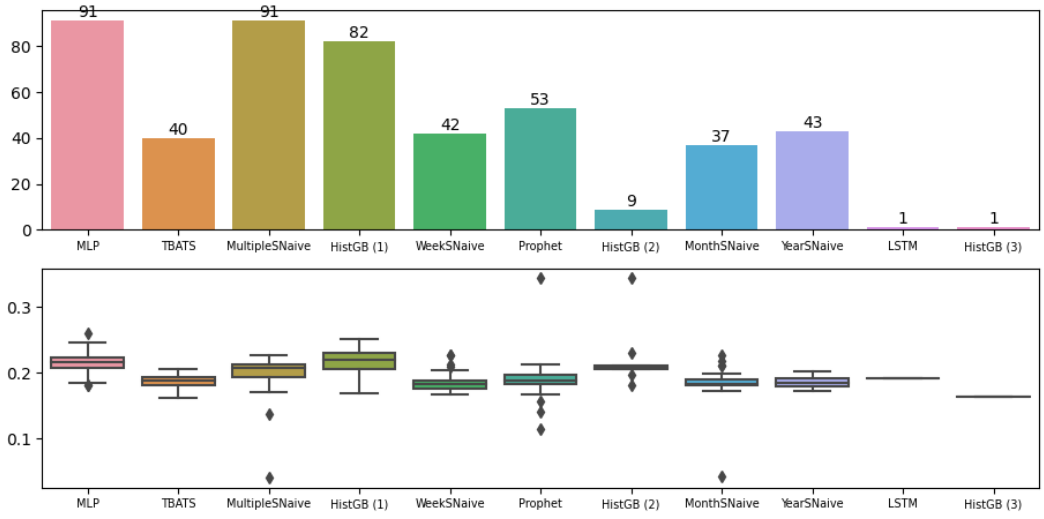

Figure 6: Number of times each model has been selected for WAE (upper panel). Boxplot displaying coefficients distribution w_i for each model in WAE (lower panel).

5. Conclusions

In this paper, we have conducted a large-scale study across 98 retail time series. We carried our research towards multi-step ahead forecasting strategies, multiple seasonality models, outlier detection and treatment, scale-independent performance measures for intermittent time series, and ensemble models in order to achieve high-quality hourly forecasts.

Regarding our contributions: (i) Seasonal recursive outperformed recursive strategy, though not as good as MIMO; (ii) Online seasonal z-score outlier detection clearly detected an huge amount of outliers, reducing the noise we fed to models, leading to lower forecast errors; (iii) Multiple seasonal naïve outperformed any individual seasonal naïve; (iv) WAE topped every single model.

In summary, we achieved the proposed goal, which was to devise a model capable of defeating our benchmark, week seasonal naïve, an already good performing model for this case study. Moreover, our proposed ensemble model, WAE, proved itself extremely successful, yielding an absolute error 1.35 times inferior, on average, compared to the benchmark. Furthermore, as seen in the previous section, multi-output strategies tend to perform better for multi-step ahead forecasting. Even though our innovative seasonal recursive strategy did not deliver top quality forecasts, in a different case study, it might thrive.

As a prospect, we plan to compare additional multi-output training strategies for the MLP architecture presented in the literature.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by COMPETE: POCI-01-0247-FEDER-039719 and FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UIDB/00127/2020.

References

Abrishami, S., & Kumar, P. (2018). Using real-world store data for foot traffic forecasting. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 1885–1890). doi:10.1109/BigData.2018.8622551.

An, N. H., & Anh, D. T. (2015). Comparison of strategies for multi-step-ahead prediction of time series using neural network. In *2015 International Conference on Advanced Computing and Applications (ACOMP)* (pp. 142–149). doi:10.1109/ACOMP.2015.24.

Ben Taieb, S., & Hyndman, R. (2012). *Recursive and direct multi-step forecasting: the best of both worlds*. Monash Econometrics and Business Statistics Working Papers 19/12 Monash University, Department of Econometrics and Business Statistics.

Bontempi, G. (2008). Long term time series prediction with multi-input multi-output local learning. *Proc. 2nd ESTSP*, (pp. 145–154).

Bontempi, G., Ben Taieb, S., & Le Borgne, Y.-A. (2013). Machine learning strategies for time series forecasting. volume 138. doi:10.1007/978-3-642-36318-4_3.

Bontempi, G., & Taieb, S. B. (2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International journal of forecasting*, *27*, 689–699.

Chatfield, C. (1978). The holt-winters forecasting procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *27*, 264–279. URL: http://www.jstor.org/stable/2347162.

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, *7*, e623.

Christ, M., Kempa-Liehr, A., & Feindt, M. (2016). Distributed and parallel time series feature extraction for industrial big data applications, .

Cortez, P., Matos, L. M., Pereira, P. J., Santos, N., & Duque, D. (2017). Forecasting store foot traffic using facial recognition, time series and support vector machines. In M. Graña, J. M. López-Guede, O. Etxaniz, Á. Herrero, H. Quintián, & E. Corchado (Eds.), *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16* (pp. 267–276). Cham: Springer International Publishing.

De Livera, A., Hyndman, R., & Snyder, R. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*, 1513 – 1527. doi:10.1198/jasa.2011.tm09771.

Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA, USA: MIT Press. http://www.deeplearningbook.org.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–80. doi:10.1162/neco.1997.9.8.1735.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, *20*, 5–10. URL: https://www.sciencedirect.com/science/article/pii/S0169207003001134. doi:https://doi.org/10.1016/j.ijforecast.2003.09.015.

Hornik, K., Stinchcombe, M. B., & White, H. L. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Hyndman, R. (2006). Another look at forecast accuracy metrics for intermittent demand. *Foresight: The International Journal of Applied Forecasting*, *4*, 43–46.

Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. (2nd ed.). Australia: OTexts.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*, 679–688. URL: https://www.sciencedirect.com/science/article/pii/S0169207006000239. doi:https://doi.org/10.1016/j.ijforecast.2006.03.001.

Junior, C., Gusmão, P., Moreira, J., & Tome, A. M. M. (2021). Time series forecasting in retail sales using lstm and prophet. In *Handbook of Research on Applied Data Science and Artificial Intelligence in Business and Industry* (pp. 241–262). IGI Global.

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*, 669–679. URL: https://www.sciencedirect.com/science/article/pii/S0169207016000121. doi:https://doi.org/10.1016/j.ijforecast.2015.12.003.

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

Kline, D. (2004). Methods for multi-step time series forecasting with neural networks. (pp. 226–250). doi:10.4018/978-1-59140-176-6.ch012.

Ma, S., & Fildes, R. (2020). Forecasting third-party mobile payments with implications for customer flow prediction. *International Journal of Forecasting*, *36*, 739–760. URL: https://www.sciencedirect.com/science/article/pii/S0169207019302365. doi:https://doi.org/10.1016/j.ijforecast.2019.08.012.

Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, *9*, 527–529. URL: https://www.sciencedirect.com/science/article/pii/0169207093900793. doi:https://doi.org/10.1016/0169-2070(93)90079-3.

de Myttenaere, A., Golden, B., Grand, B. L., & Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing*, *192*, 38–48. URL: https://doi.org/10.1016%2Fj.neucom.2015.12.114. doi:10.1016/j.neucom.2015.12.114.

Rojas, R. (1996). The backpropagation algorithm. In *Neural Networks: A Systematic Introduction* (pp. 149–182). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-61068-4_7. doi:10.1007/978-3-642-61068-4_7.

Schnaars, S. P. (1986). Long-range forecasting: From crystal ball to computer: J. scott armstrong, 2nd ed. (wiley, new york, 1985) [uk pound]22.95 (paper), pp. 689. *International Journal of Forecasting*, *2*, 387–390. URL: https://EconPapers.repec.org/RePEc:eee:intfor:v:2:y:1986:i:3:p:387-390.

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., & Lendasse, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing*, *70*, 2861–2869. URL: https://www.sciencedirect.com/science/article/pii/S0925231207001610. doi:https://doi.org/10.1016/j.neucom.2006.06.015. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005).

Sorjamaa, A., & Lendasse, A. (2006). Time series prediction using dirrec strategy. (pp. 143–148). volume 6.

Suthaharan, S. (2016). Decision tree learning. In *Machine Learning Models and Algorithms for Big Data Classification* (pp. 237–269). Springer.

Taieb, S. B., Bontempi, G., Atiya, A., & Sorjamaa, A. (2011). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. URL: https://arxiv.org/abs/1108.3259. doi:10.48550/ARXIV.1108.3259.

Taieb, S. B., Bontempi, G., Sorjamaa, A., & Lendasse, A. (2009). Long-term prediction of time series by combining direct and mimo strategies. In *2009 International Joint Conference on Neural Networks* (pp. 3054–3061). IEEE.

Taylor, J. W. (2003). Short-term electricity demand forecasting using double seasonal exponential smoothing. *The Journal of the Operational Research Society*, *54*, 799–805. URL: http://www.jstor.org/stable/4101650.

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, *72*, 37–45. URL: https://ideas.repec.org/a/taf/amstat/v72y2018i1p37-45.html. doi:10.1080/00031305.2017.138.

Tofallis, C. (2015). A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society*, *66*, 1352–1362. URL: https://doi.org/10.1057%2Fjors.2014.103. doi:10.1057/jors.2014.103.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*, 1550–1560. doi:10.1109/5.58337.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, *6*, 324–342. URL: http://www.jstor.org/stable/2627346.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800231. doi:https://doi.org/10.1016/S0893-6080(05)80023-1.

Xiong, T., Bao, Y., & Hu, Z. (2013). Beyond one-step-ahead forecasting: Evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Economics*, *40*, 405–415. URL: https://www.sciencedirect.com/science/article/pii/S0140988313001746. doi:https://doi.org/10.1016/j.eneco.2013.07.028.