forecasting

MDPI

Article

Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices

Lorenzo Menculini [1,*], Andrea Marini [1], Massimiliano Proietti [1], Alberto Garinei [1,2], Alessio Bozza [3], Cecilia Moretti [4] and Marcello Marconi [1,2]

1 Idea-re S.r.l., 06128 Perugia, Italy; amarini@idea-re.eu (A.M.); mproietti@idea-re.eu (M.P.)
2 Department of Engineering Sciences, Guglielmo Marconi University, 00193 Rome, Italy; a.garinei@unimarconi.it (A.G.); m.marconi@unimarconi.it (M.M.)
3 Cancelloni Food Service S.p.A., 06063 Magione, Italy; a.bozza@cancelloni.it
4 Independent Researcher, Via Parco 6, 06073 Corciano, Italy; cecilia.moretti1@gmail.com
* Correspondence: lmenculini@idea-re.eu

Abstract: Setting sale prices correctly is of great importance for firms, and the study and forecast of prices time series is therefore a relevant topic not only from a data science perspective but also from an economic and applicative one. In this paper, we examine different techniques to forecast sale prices applied by an Italian food wholesaler, as a step towards the automation of pricing tasks usually taken care by human workforce. We consider ARIMA models and compare them to Prophet, a scalable forecasting tool by Facebook based on a generalized additive model, and to deep learning models exploiting Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs). ARIMA models are frequently used in econometric analyses, providing a good benchmark for the problem under study. Our results indicate that ARIMA models and LSTM neural networks perform similarly for the forecasting task under consideration, while the combination of CNNs and LSTMs attains the best overall accuracy, but requires more time to be tuned. On the contrary, Prophet is quick and easy to use, but considerably less accurate.

Keywords: time-series; forecasting; deep learning; ARIMA; prophet; prices; sales

check for updates

Citation: Menculini, L.; Marini, A.; Proietti, M.; Garinei, A.; Bozza, A.; Moretti, C.; Marconi, M. Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices. *Forecasting* **2021**, *3*, 644–662. https://doi.org/10.3390/forecast3030040

Academic Editor: Sonia Leva

Received: 16 August 2021
Accepted: 9 September 2021
Published: 15 September 2021

1. Introduction

The main aim of firms is profit maximization. To achieve this goal, the constant updating and forecasting of selling prices is of fundamental importance for every company. Although digital transformation is a phenomenon that involves all companies, from small to large, many of them still update prices by hand through logics that are not always clear nor objective and transparent, but rather based on the experience and expertise of those in charge of updating the price list. On the other hand, the automation of price prediction and update can provide a strong productivity boost by freeing up human resources, which can thus be allocated to more creative and less repetitive tasks. This also increases the morale and commitment of employees; it also speeds up the achievement of goals, and improves accuracy by minimizing human errors. Subjectivity is also reduced: once the operating criteria have been established, forecast algorithms will keep behaving consistently. This in turn means an improvement in compliance.

Besides the automation of price updates, the prediction of the sales prices charged to customers in the short term also holds great value. In general, organizations across all sectors of industry must undertake business capacity planning to be efficient and competitive. Predicting the prices of products is tightly connected to demand forecasting and therefore allows for a better management of warehouse stocks. The current economic crisis caused by COVID-19 has highlighted the value of such management optimization, stressing the importance of the companies' ability to minimize inventory reserves and just-in-time production models. Forecast models considered in this paper can contribute to keeping the difference between wholesale purchase prices and company's sales prices under control, in view of maximizing the gross operating income. They can therefore help companies avoid the risk of incalculable losses and ultimately improve their contractual capacity.

The present work proposes to deal with these topics by investigating and comparing different price forecasting models. The specific task we consider is that of predicting the prices of three food products sold by a medium/large-size local wholesaler based in central Italy. In such way, we investigate the predictability of wholesale prices, comparing the performance of traditional econometrics time-series forecasting models with Facebook's Prophet and machine learning (ML) models. The main goal of this paper is therefore to develop a forecasting model that could represent a first step towards the automation of the price-setting process, thus effectively aiding the work of company employees. Indeed, a forecasting tool that can reliably predict how product prices should be updated, effectively reproducing the logic adopted by company experts in doing so, aims at being of practical use in automating the maintenance and management of price lists. Scalability and flexibility of the models presented in this paper are also an important point: for the sake of simplicity, we have applied the models to three different products, but we underline that the same models and algorithms can be easily applied to any product.

Time-series forecasting has always been a major topic in data science with plenty of applications. For a general review of some of the most used tools, see for example [1]. See also [2] for a very recent and comprehensive review with special consideration on the topics touched upon in this work. Well-known traditional econometric methods are not always appropriate to study and forecast big and noisy time-series data. This has generated particular interest in machine learning methods, bolstering data driven approaches that include a wide range of methods that have the advantage of not relying on prior assumptions and knowledge of data. See for example [3–6] for reviews focusing on the application of deep learning [7] to time series. Long short–term memory (LSTM) networks [8] and convolutional neural networks (CNNs) [9] are almost ubiquitous in time-series forecasting with machine learning. CNNs are even more commonly used for image recognition and feature extraction. However, the forecasting accuracy of standalone CNNs can be relatively low [10].

The literature concerning economic time-series prediction employing various methos—from classical to artificial intelligence ones—is very rich. Nevertheless, although we believe automatic updating mechanisms and forecasting of sale prices are of uttermost relevance, the literature on these topics is not as developed as one would expect. Most studies focus primarily on the implementation of models for the analysis and forecasting of general price levels (inflation) or commodity and stock market prices, or on demand management and forecasting (see e.g., [11–13]). Improving sales forecasting in business organizations is becoming more and more important, as witnessed for example by the M5 competition [14] announced by MOFC and devoted to the forecasting of sales data from three different US states, made available by Walmart. The focus of the competition was put identifying the most appropriate prediction methods for different types of situations, comparing the accuracy and uncertainty of ML and deep learning methods vis-à-vis those of standard statistical ones, with the overarching aim of improving the use of forecasting models by businesses and non-profit organizations.

The forecasting of food prices in China was considered by the authors of [15,16]. In particular, Zou et al. [16] compared the performances of ARIMA, neural networks (NNs) and a combination of the two to forecast wheat prices in the Chinese market. Their findings showed that overall, NNs perform best at the task. Neural networks were also employed in [17] to forecast monthly wholesale prices of two agricultural products. Ahumada and Cornejo [18] considered a similar problem, also taking into account possible cross-dependencies of different product prices. In [19] the author focused on sales forecasting using machine learning models, a topic similar to the one considered in the present paper. For more recent work on forecasting commodities prices see [20], where the authors forecasted gold prices, and [21] where the Levenberg-Marquardt Backpropagation (LM-BP) algorithm was applied to stock prices prediction. Other authors used machine learning methods for inflation forecasting [22,23], also in comparison with more classical econometric models [24]. Xue et al. [25] recently presented a high-precision short-term forecasting model for financial market time series employing deep LSTM neural networks, comparing them with other NN models. Their results showed that LSTM deep neural networks have high forecasting accuracy for stock market time series. In 2020, Kamalov [26] evaluated multilayer perceptrons, CNNs and LSTM neural networks to forecast significant changes in stock prices for four major US public companies, showing that these three methods yield better results when compared to similar studies that forecast the direction of price change. For models similar to the ones considered in this work and applied again to stock indexes forecasting, see [27]. Hybrid ARIMA/neural network models where instead studied by the authors of [28]. Stock prices have also been forecasted using LSTMs in conjunction with the attention mechanism [29]. Machine learning models using LSTMs and CNNs are of widespread use in time-series forecasting, well beyond the financial and economic realm. For recent work on time-series forecasting using machine learning outside the economic and financial area see [30], an application to COVID-19 spreading forecasting, and [31] for an application of deep learning to influenza prevalence forecasting.

In this paper, we compare the performance of standard Autoregressive Integrated Moving Average (ARIMA) models [32], which we take as a benchmark, to Prophet—a forecasting tool developed by Facebook and based on a Generative Additive Model (GAM) [33]—and machine learning models exploiting LSTMs, both on their own and in combination with CNNs. ARIMA univariate models are considered a standard reference model in econometrics. The compared models are rather different, as are the datasets that they accept in input, making the comparison interesting. On one hand, Prophet's driving principles are simplicity and scalability; it is specifically tailored for business forecasting problems and handles missing data very well by construction. On the other, the NN models we construct directly lend themselves to carrying out a multivariate regression, fully exploiting all the collected data; however, they also require some data pre-processing, as does ARIMA. Prophet has been compared to ARIMA models for the prediction of stock prices [34] and bitcoin [35].

Our results indicate that the combination of CNNs and LSTMs yields the most accurate results for all the three products, but require the longest and computationally more expensive tuning. On the contrary, Prophet performances were not brilliant, but model tuning and data preparation were particularly quick. ARIMA and LSTM-only neural networks showed good performance both in terms of accuracy and time required for model selection and training.

The rest of the paper proceeds as follows. Section 2 introduces the dataset features, discussing its properties and some pre-processing steps that were taken on it; it also briefly presents the three models under consideration, their set-up and tuning. In Section 3 we give the results obtained with the three approaches and compare them. We conclude in Section 4 by discussing the results of this study and providing an outlook on future perspectives in the light of the paper findings.

2. Materials and Methods

2.1. Dataset Description and Preparation

For this study we had access to a dataset comprising a total of approximately 260,000 food order records, reporting the following information: date of order , order number, unit sale price,article code, sold quantity, customer code, offer (if present) and offer type, unitary product cost. The records were collected by the wholesaler in a period ranging from year 2013 to 2021. For the study conducted in this paper, we decided to focus on the three products with the most records (after resampling the dataset with weekly frequency, the number of available records for most of the products was relatively low (about 100 or less). In fact, only the three selected products had records dating back to 2013, resulting in weekly datasets with more than 200 records), namely Carnaroli rice 1 kg $\times$ 10 (henceforth product 1), Gorgonzola cheese 1/8 of wheel 1.5 kg (product 2) and Cured aged ham 6.5 kg (product 3). The forecasting task considered in this work was to predict the average selling price for the following week, for each of the selected products.

As a first thing, we chose to leave out all data following the outbreak of the COVID-19 pandemic. This was motivated by the huge impact that the lockdowns and restrictions imposed by the authorities had on the food and catering sector, introducing a major shock in sales trends at all scales. Therefore, we excluded all records dated later than 9 March 2020 (last day before the first national lockdown in Italy).

A preliminary data analysis revealed that the dataset contained a good number of outliers: for some of them, it appeared evident that this was due to incorrect typing of the product sale price. To improve the quality of the dataset, we calculated the *z-score* of each record based on its price as $z = (p - \bar{p}^{(w)})/\sigma^{(w)}$, where p is the unit sale price and $\bar{p}^{(w)}$ and $\sigma^{(w)}$ are the mean and standard deviation for the selected product, weighted by the quantity sold in each order. Then, we filtered out all records with $|z| > 4$. Figure 1 shows the price distribution for product 2, after the filtering.

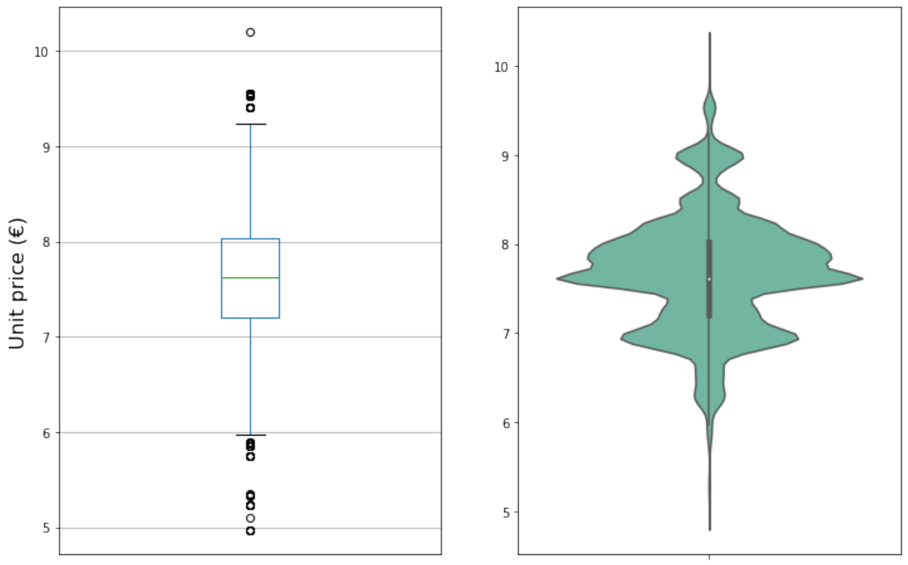
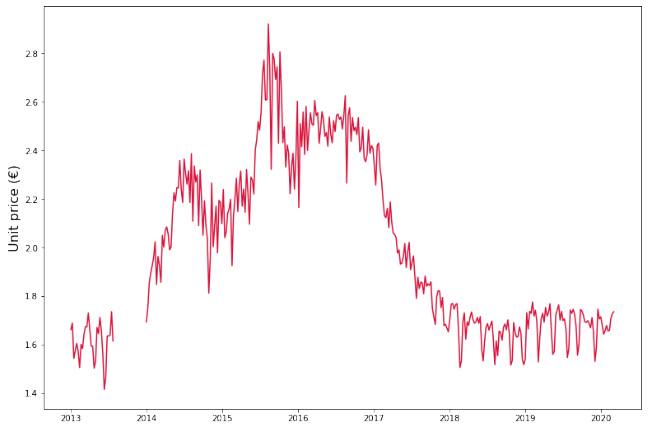


Figure 1. Price distribution (after *z-score* filtering) for product 2: to the left, boxplot, and to the right, violin plot.

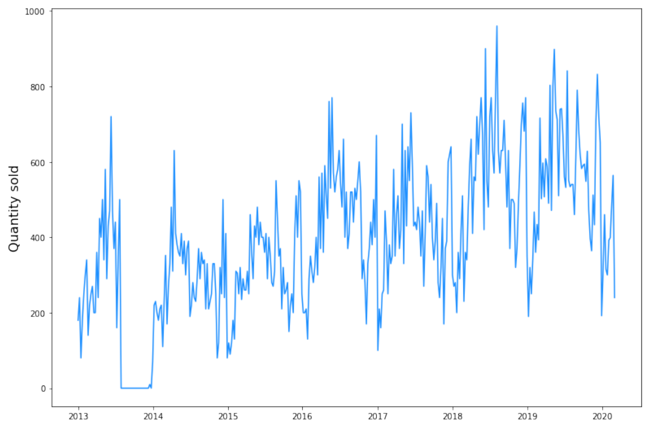
In view of the subsequent time series forecasting and as a further step in dealing with data inaccuracies, we decided to resample the dataset with weekly frequency. This was done by cumulating the number of orders in each window and calculating the average sale price for each week. For later use in neural network models, when resampling data we also kept track of the following fields in the dataset: number of served customers, number of orders, number of orders on sale, (weighted) average product cost, and (weighted) price standard deviation. Table 1 summarizes the main features of the resampled price time series for each of the products. In Figures 2–4 we display the time series of sale prices and sold quantities after resampling. All prices, here and everywhere in the paper, are intended in euros (€).

Table 1. Mean and standard deviation of the weekly prices time series.

Product	1	2	3
Mean (€)	1.99	7.64	7.27
Std (€)	0.36	0.40	0.26

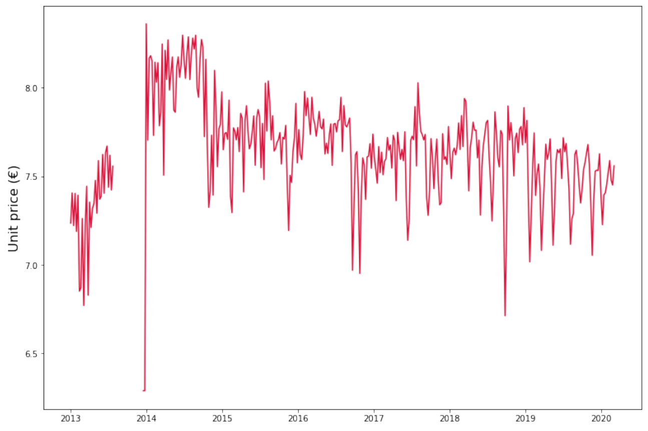


(**a**)

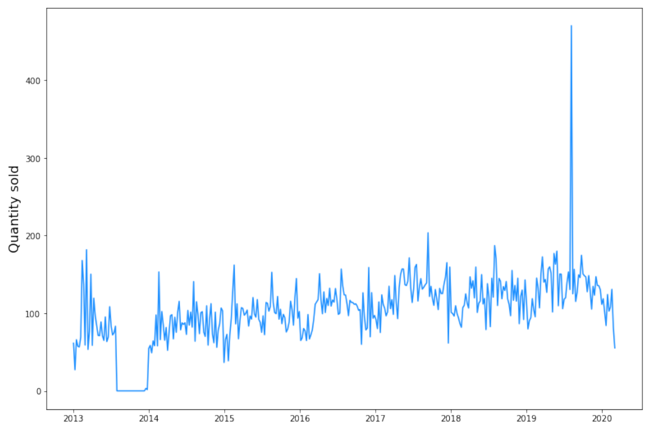


(**b**)

Figure 2. (**a**) Unit price and (**b**) sold quantity time series for product 1 after resampling with weekly frequency.

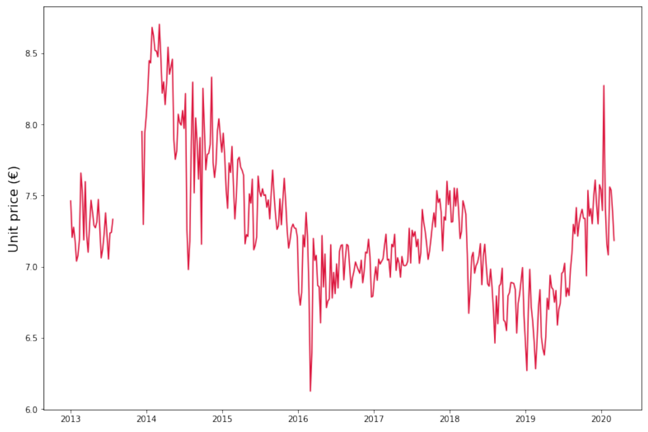


(**a**)

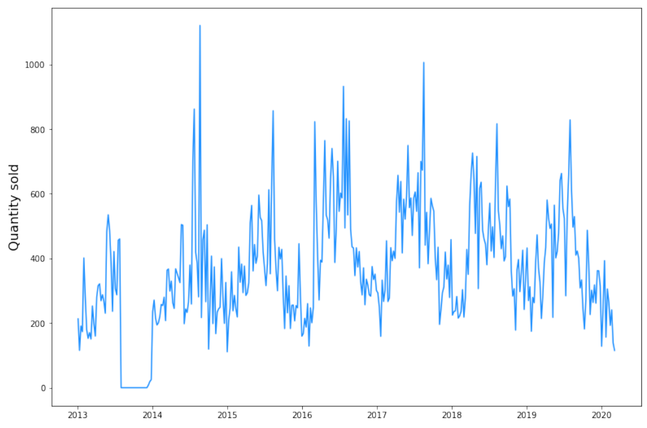


(**b**)

Figure 3. (**a**) Unit price and (**b**) sold quantity time series for product 2 after resampling with weekly frequency.



(**a**)



(**b**)

Figure 4. (**a**) Unit price and (**b**) sold quantity time series for product 3 after resampling with weekly frequency.

We then split the weekly dataset in the following way:

- Training dataset: years 2013–2017;
- Validation dataset: year 2018;
- Test dataset: years 2019–March 2020.

This choice was made to make the training set as large as possible, while also having validation and test sets that cover at least one year of data.

As can be seen in Figures 2–4, even after resampling the price time series have missing periods. Therefore, they are not suited for being given as input to ARIMA and neural network forecasting models, as they require evenly spaced data. To overcome this problem, we adopt the following strategy for all products:

1. Since the time series for all products have a long window with no data in the second half of year 2013, we do not consider this empty period and start right after it;
2. When occasional weeks with no data occur, we take the average of the preceding and following week prices and interpolate.

In this way, we were able to fill in all empty weeks—in fact, after resampling the missing records were very sparse. Please note that the above procedure is only necessary for preparing the dataset for ARIMA and NN models, as Prophet has no problems in handling missing datapoints. The size of the datasets for each product, both before and after removal of empty periods, is summarized in Table 2.

Table 2. Dataset size (# of datapoints) for each product and forecasting model.

Product	Model	Train	Valid	Test
1	Prophet	240	52	62
	ARIMA & NN	211		
2	Prophet	241	52	62
	ARIMA & NN	211		
3	Prophet	242	52	62
	ARIMA & NN	212		

2.2. ARIMA Models

ARIMA models [32] are among the simplest and most used econometric approaches to univariate time-series modeling. In this work, we implemented non-seasonal ARIMA models, neglecting the modulation effects of holidays and using therefore pure trend lines.

In econometrics, it is quite customary when dealing with price variables to transform prices through a logarithmic map, since this generally leads to better results. We decided to follow this approach when using the ARIMA modeling, thus working with $\log(p_t)$ in the model fitting. As a first step we checked the stationarity properties of the time series. We performed the *Augmented Dikey–Fuller* (ADF) unit root test using the built-in method in the *statsmodels* Python package. The results we obtained are qualitatively similar for all the three products we considered: for the $\log(p_t)$ time series one cannot reject the null hypothesis of the presence of a unit root, signaling the non-stationarity of the series. First differencing the series, i.e., considering $\Delta\log(p_t) = \log(p_t) - \log(p_{t-1})$, makes it stationary. Thus, the $\log(p_t)$ series are integrated of order one, and accordingly the models we considered are ARIMA$(p, 1, q)$. Table 3 shows the full results of the ADF test.

Table 3. Results of the augmented Dikey–Fuller tests.

Product	Series	ADF Test Statistics	*p*-Value	Lags Used
1	$\log(p_t)$	-1.10	0.713	4
	$\Delta\log(p_t)$	-12.3	6.63×10^{-23}	3
2	$\log(p_t)$	-2.84	0.0530	7
	$\Delta\log(p_t)$	-11.5	4.03×10^{-21}	6
3	$\log(p_t)$	-2.29	0.175	7
	$\Delta\log(p_t)$	-11.5	4.12×10^{-21}	6

To have a rough indication on the AR orders, p's, and on the MA orders, q's, we computed the sample autocorrelation function (ACF) and the partial autocorrelation function (PACF) for $\Delta \log(p_t)$. Recall that:

- for an exact MA(q), ACF is zero for lags larger than q;
- for an exact AR(p), PACF is zero for lags larger than p.

As an example, we show the plots of these functions for product 2 in Figure 5.

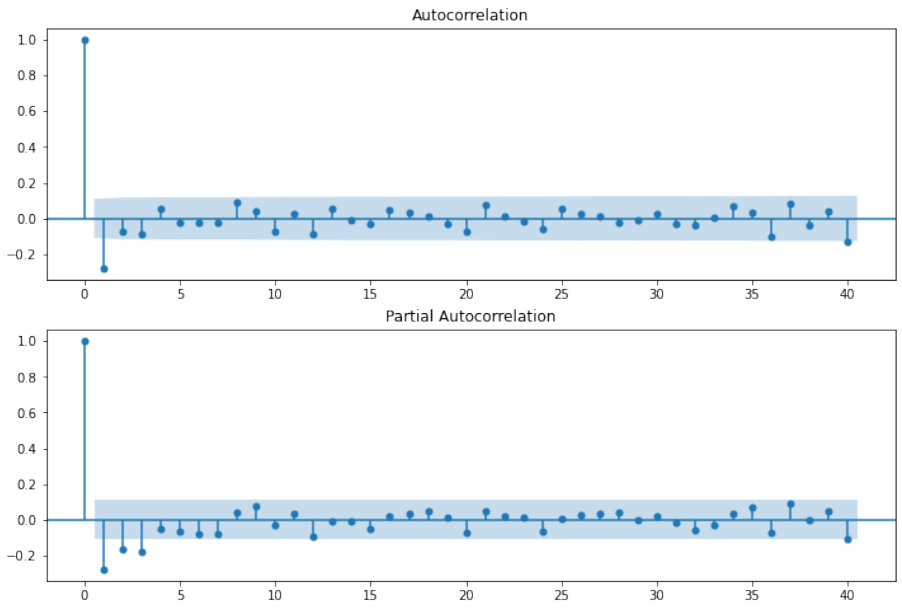


Figure 5. Autocorrelation function and partial autocorrelation function for $\Delta \log(p_t)$ of product 2.

In the ARIMA model selection and fitting procedures, we used a different dataset splitting scheme with respect to the one described above, in that the *training* set comprised years 2013–2018 (i.e., the union of the former training and validation sets). This is since we decided not to use the validation set to select the hyperparameters p and q, instead exploiting the *Bayesian Information Criterion* (BIC) as a metric for model comparison [36]. Hence, we took into account different combinations of p and q around the values suggested by the ACF and PACF plots, and eventually we selected the model with least BIC.

2.3. Prophet

Prophet is an open-source tool provided by Facebook Inc., available both in Python and R. For the current analysis, the Python package (with Python 3.9) was used. As explained by the authors [37], the idea leading to Prophet was to develop a flexible forecasting tool which is easy to both use and tune. The underlying model features a decomposable time series with three components: growth (or trend) $g(t)$, seasonality $s(t)$ and holidays $h(t)$ (if present). In the present case, there are no obvious holidays to consider, as the wholesaler's customers are mainly restaurants and hotels, which tend to stay open during holidays. The time series is therefore decomposed as

$$y(t) = g(t) + s(t) + \epsilon_t \,, \tag{1}$$

where ϵ_t encodes variations that are not taken into account by the model, and which are assumed to be normally distributed [37]. The Prophet model can be seen as a Generative Additive Model (GAM) [33]. In this framework, forecasting is phrased as a curve-fitting task, with time as the only regressor, so the model is univariate.

The trend function adopted for the problem under study is a piecewise linear function written as

$$g(t) = \big(k + \sum_{i:t>s_i} \delta_i\big)t + \big(m + \sum_{j:t>s_j} \gamma_j\big)\,, \tag{2}$$

where k is a scalar coefficient, s_i are the *trend changepoints*—i.e., S times s_1, s_2, ... , s_S at which the angular coefficient of the trend is allowed to change—δ_i are the rate adjustments, and $\gamma_j = -s_j\delta_j$ are parameters used to make the function continuous. The algorithm starts with a number $S = 25$ of potential changepoints, placed in the first 80% of the time series to avoid responding to fluctuations in the last part of the series. Then, the actual changepoints are selected by putting a sparse prior of the kind $\delta_j \sim \text{Laplace}(0, \tau)$, with τ (tunable hyperparameter) regulating the magnitudes rate adjustments. Note that using a Laplace prior is equivalent to L1-regularization. A larger τ means the model has more power to fit trend changes.

As for seasonality, Prophet accounts for it using Fourier series, namely

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right)\right). \tag{3}$$

Since we considered weekly data with no other obvious expected seasonality effects but yearly ones, we had $P = 365.25$d. For weekly seasonality, the truncation parameter is set to $N = 10$ by the authors of [37] when modeling yearly seasonality, and we follow this specification. When performing the fit, a smoothing prior $\beta \sim N(0, \sigma^2)$ is imposed on the $2N$ components $\beta = (a_1, \ldots, a_n, b_1, \ldots, b_n)^T$, with σ a second hyperparameter (essentially acting as an L2-regularization parameter).

Prophet fits its GAM using the L-BFGS quasi-Newton optimization method of [38] in a Bayesian setting, finding a maximum *a posteriori* estimate.

2.4. Neural Networks

The advent of artificial intelligence, in particular machine learning, has led to the development of a set of techniques that have proved to be very useful in many different areas. One breakthrough has certainly been deep learning [7], which has revolutionized our way of handling and exploiting information contained in data. Deep learning can effectively detect and model hidden complexity in data, automatically extracting features that should otherwise be extracted manually by dataset inspection.

A standard choice when facing problems involving time series is that of using LSTM neural networks, a kind of recurrent neural networks (RNNs) devised by Hochreiter and Schmidhuber in 1997 [8]. Like all RNNs, they can by construction handle data endowed with temporal structure, while also providing a way to deal with the vanishing gradient problem [39]. Here we will describe the application of LSTM NNs to the problem under study, both on their own and in combination with CNNs. Indeed, standard LSTM NNs for time-series forecasting can be enriched with one-dimensional convolutional layers that sequentially apply a unidimensional filter to the time-series. Convolutions can be seen as non-linear transformations on the time-series data. This enhances the model's capability to learn discriminative features which are useful for the forecasting and that can be fed to the LSTM layers that follow. The models developed in this work were trained and tested using Python 3.9 and TensorFlow 2.5.

Unlike in the ARIMA and Prophet case, where the problem was kept univariate, with NNs we decided to exploit a larger fraction of the information available in the dataset by setting up a multivariate regression. By NN models' construction, this can be done without additional effort with respect to univariate models. Note however that in Appendix A we also provide results obtained by univariate deep learning models. Since calendar dates cannot be used as input variables to a NN, we made an addition to the fields listed in Section 2.1, performing a time embedding to provide information about seasonality. We did this by adding the columns

$$\text{w\_cos} = \cos(2\pi w/52.1429)\,, \tag{4a}$$

$$\text{w\_sin} = \sin(2\pi w/52.1429)\,, \tag{4b}$$

where w is the week number ($w = 0, 1, \ldots, 52$). Therefore, we had a total of 9 input columns that were passed to the NN models. A sample of the input dataset for one product is shown in Table 4.

Table 4. A slice of the dataset used to generate input data for the NN models, for a given product. The columns contain the following information: *quant* is the number of units sold, *customers* is the number of different customers served, *orders* is the number of orders, *on sale* indicates how many orders had a price discount, *avg_cost* was the average cost of the product for the wholesaler, *w _cos* and *w_sin* are as defined in Equation (4), *p_std* is the standard deviation of the product prices, and *price_avg* is the weighted average of the product sale price.

Quant	Customers	Orders	On Sale	cost_avg	w_cos	w_sin	p_std	price_avg
10	1	1	0	1.40	0.990	−0.141	0	1.41
0	0	0	0	1.40	1.000	−0.0214	0	1.55
70	6	6	0	1.40	0.993	0.120	0.0690	1.69
220	17	18	0	1.40	0.971	0.239	0.0580	1.75
230	14	15	0	1.39	0.935	0.353	0.0685	1.86

To construct the actual training dataset (made of a tensor $\mathbf{x_t}$ and a scalar y_t, for each time t) that could be fed to LSTM neural networks, we then performed the following steps:

- reshape data so that at each time t, $\mathbf{x_t}$ is a $n \times 9$ tensor containing the n last values of each time series in Table 4;
- set $y_t = \Delta_{t+1} = p_{t+1} - p_t$ as the variable to be used in the cost function.

In this way, the model learns to predict y (price variation) at each time based on information about the last n timesteps. Predicting the increment of the quantity of interest instead of the quantity itself is a well-known way to improve performance when training machine learning models. Moreover, as already explained above (see Table 3), we checked through the ADF test that the Δ_t time series was stationary. The number n of timesteps used depends on the model and will be specified later.

The NN models tried in this paper for predicting product prices fall in two classes: those using only LSTM layers and those with CNN layers before the LSTM. We denoted these classes A and B, respectively.

3. Results

In this section, we report results obtained with the three different approaches to forecasting—namely ARIMA, Prophet and deep learning—studied in this work.

3.1. ARIMA Results

We considered first ARIMA models, to provide a standard performance benchmark with which to compare the other models developed in the rest of this work.

The best-performing ARIMA models for the three products are given in the first column of Table 5. As outlined in Section 2.2, they were selected by considering the series of the price logarithms $\log(p_t)$ and using a least BIC criterion [36]. For the sake of comparison with the other models, we transformed back the $\log(p_t)$ series to the p_t series to compute the root mean squared error (RMSE) between the predicted and observed increments

$$\hat{\Delta}(t) = \hat{p}_t - p_{t-1}\,, \qquad \Delta(t) = p_t - p_{t-1}\,, \tag{5}$$

$\hat{p}_t$ being the predicted price at time t.

In all cases, we checked also that the Ljung–Box statistics [40,41] for 1-, 6- and 12-lag residual autocorrelations do not reject the null hypothesis, so the residuals can be considered approximately white noise.

The results obtained on the test set by the selected ARIMA models are summarized in Table 5. We also report values for the MAE (mean absolute error) and MAPE (mean absolute percent error) and ME (mean error).

Table 5. Selected ARIMA models for the three products, associated RMSE on the entire training+validation set, and performances on the test set (RMSE, MAE and MAPE). Please note that MAPE is computed for price time series p_t, not for the $\Delta(t)$ time series.

Product	Model	Train+Val RMSE	Test RMSE	Test MAE	Test MAPE	Test ME
1	ARIMA (2, 1, 0)	0.097	0.0758	0.173	0.215	0.00521
2	ARIMA (0, 1, 2)	0.232	0.0581	0.132	0.159	0.0140
3	ARIMA (3, 1, 1)	0.211	0.0348	0.0178	0.0222	0.0507

3.2. *Prophet Results*

3.2.1. Prophet Grid Search

As suggested in the Prophet documentation and reviewed in Section 2.3, one can tune the τ (trend changepoints prior scale) and σ (seasonality prior scale) hyperparameters so that the model fits data as well as possible. We did so by performing a grid search over τ and σ in the following way: for each combination of $\tau \in \{0.005, 0.01, 0.05, 0.1, 0.5\}$ and $\sigma \in \{0.01, 0.05, 0.1, 0.5, 1, 2\}$, we started by fitting the model over the training dataset, and predicted the price for the following week (first datapoint in the validation set). We calculated the squared error between predicted and observed price. Then, we moved on to the second datapoint in the validation set, performed a new fit using also the first validation set datapoint, and predicted the price for the following week. The whole process was repeated until the validation dataset was exhausted, and the RMSE was calculated. In other words, this is nothing but a cross validation procedure that starts by fitting the entire training dataset—predicting the values of the series for the following timestep—and ends when the entire training plus validation dataset is fitted. For each product, the configuration with least RMSE was selected, yielding the results contained in the first four columns of Table 6.

Table 6. Prophet results: RMSE on the validation set, RMSE, MAE, MAPE and ME calculated on the test set. The metrics were calculated as described in the main text. Please note that the MAPE is computed for the price times series p_t, not for the $\Delta(t)$ time series.

Product	τ	σ	Valid RMSE	Test RMSE	Test MAE	Test MAPE	Test ME
1	0.5	0.01	0.0831	0.00812	0.0694	0.0414	−0.0152
2	0.1	0.01	0.293	0.220	0.165	0.0224	0.0456
3	0.5	1.0	0.215	0.350	0.301	0.0424	−0.217

3.2.2. Prophet Forecasting

After selecting the best values of the parameters for each product, we employed them to specify the Prophet model in the test phase. Testing took place in the following way: we started by fitting the model over the entire training plus validation dataset, predicting the first data entry in the test dataset and calculating the squared error. We repeated the procedure for all entries in the test dataset, each time using all previous history, and calculated the achieved RMSE (as well as the other metrics) at the end of the process. Please note that by employing this procedure, also the test dataset is progressively used to fit the model. We plot in Figure 6 the result of the fit over the entire dataset for product 1, i.e., the function that would be used to predict the unit price for the first week following the end of the test dataset.

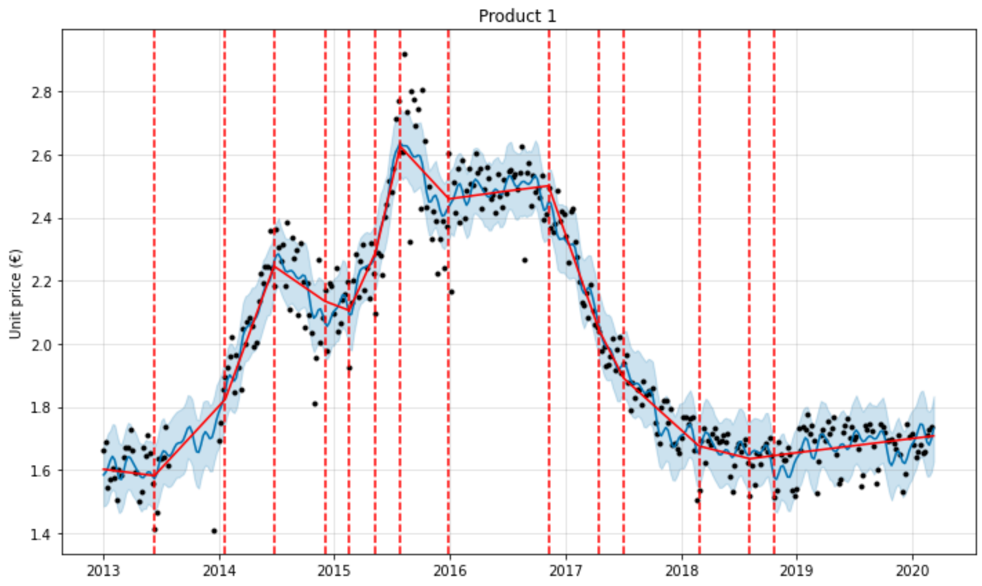


Figure 6. Prophet fit over the entire product 1 dataset. The actual fit is shown in light blue, the red line shows the trend function, while trend changepoints are indicated by red dashed vertical lines.

Figure 7 shows instead the plot of the predicted and observed increments—$\hat{\Delta}(t)$ and $\Delta(t)$ as defined in Equation (5)—in the case of product 2. The performance of Prophet in forecasting the weekly price time series is summarized in Table 6. As done for ARIMA models, besides the RMSE parameter used in the fine tuning, we also report values for the MAE and MAPE.

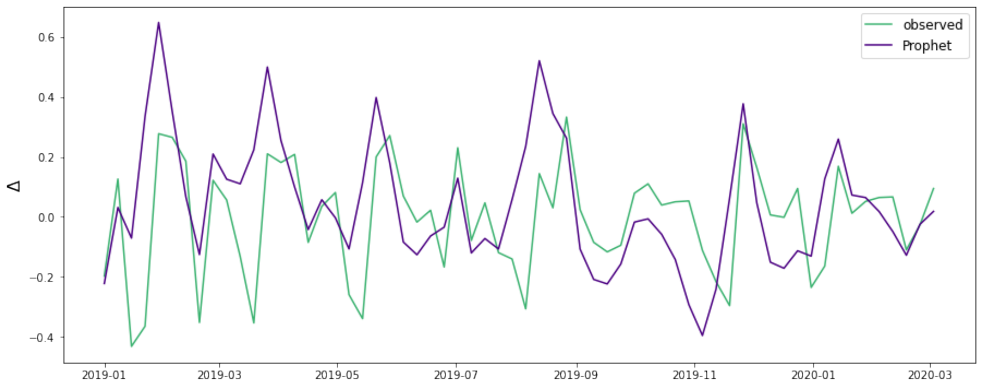


Figure 7. Comparison of Prophet forecasts and observed data for price variations in the test set, for product 2. The corresponding RMSE is 0.220.

3.3. Neural Networks Results

3.3.1. NN Grid Search

As a third forecasting tool, we studied deep neural networks. Two different classes of models—class A and B as introduced in Section 2.4—were analyzed. Data were standardized using a *z-score* normalization (specifically, the *MinMaxScaler* function of the *scikit-learn* Python package was employed), and we used a batch size of 32. We performed two different grid searches to select the best model in each class, as we now briefly describe.

For class A, we trained NN models with the following architecture and hyperparameters:

- a number $l \in \{1, 2, 3\}$ of LSTM layers;
- a number $n_u \in \{32, 64, 96\}$ of LSTM neurons in every layer, with *normal Glorot* weight initialization [42]. Each layer has the same number of units;

- following each LSTM layer, a dropout layer with dropout rate $r \in \{0.1, 0.2, 0.3\}$. The dropout rate is taken to be equal for all layers;
- an output layer consisting of a single neuron with linear activation function and normal Glorot initialization;
- an MSE cost function and *Adam* optimization algorithm [43], with learning rate (or *stepsize*) $\alpha \in \{0.0005, 0.001\}$;
- an early stopping procedure, monitoring the cost function on the validation set with a patience of 5 epochs, while also having an upper bound of 150 training epochs.

For this class of models, we used several timesteps $n = 4$. The grid search over the hyperparameters l, n_u, r, α was performed by initializing and training each model ten times, monitoring the cost function on the validation set and recording the best result obtained for every configuration. The best-performing models for each product are reported in Table 7.

Table 7. Results of the grid search on class A (LSTM-only).

Product	l	n_u	r	α	**Train RMSE**	**Valid RMSE**
1	3	32	0.1	0.001	0.0964	0.0612
2	3	32	0.1	0.001	0.157	0.176
3	1	64	0.1	0.001	0.143	0.148

The second class of models, class B, consisted of a combination of CNN layers and LSTM layers, as done for example in [20,27]. We trained models with the following specifications:

- two one-dimensional convolutional layers (conv1D) with $f \in \{10, 20, 30\}$ *output filters* each, *kernel size* $k_s \in \{2, 4\}$ and *relu* (rectified linear unit) activation function. The hyperparameters f, k_s were taken to be the same for both layers;
- in each of the conv1D layers, padding was set to *same*, *causal*, or no padding at all. The corresponding hyperparameters are dubbed pad_1, pad_2;
- an *average pooling* layer in between the convolutional layers, with *pool size* equal to 2 and no padding;
- following the above layers, LSTM layers are added following the same structure as for the first class of models.

The reason behind adding CNN layers is that they can help make better use of the data history, improving the ability of LSTM networks to learn from long series of past data. The grid search was indeed performed with varying numbers of timesteps $n \in 4, 8, 12$. As for the previous class, each model was initialized and trained ten times. Results for the grid search over the hyperparameters $l, n_u, r, \alpha, f, k_s, n$ are shown in Table 8. Figure 8 shows the trajectory of training and validation cost functions for the best-performing model in the case of product 1.

Table 8. Results of the grid search on class B (CNN + LSTM).

Product	l	n_u	r	α	f	k_s	pad_1	pad_2	n	**Train RMSE**	**Valid RMSE**
1	1	64	0.3	0.0005	20	2	causal	causal	8	0.0770	0.0553
2	3	64	0.1	0.0005	20	2	no	same	12	0.175	0.165
3	1	32	0.2	0.001	20	2	same	same	8	0.148	0.132

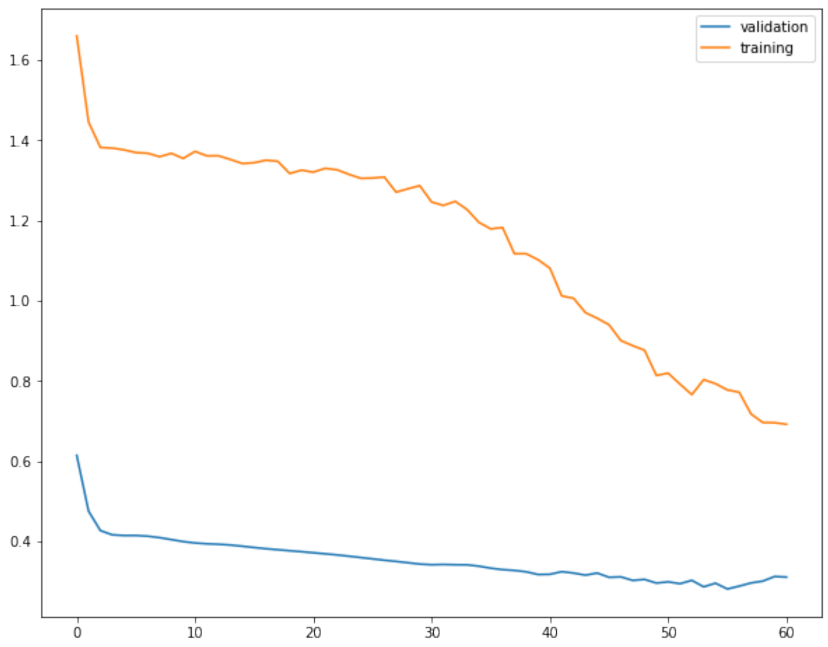


Figure 8. Trajectory of training and validation losses, as a function of the training epoch, for product 1. Please note that the cost functions are calculated on rescaled data, therefore they cannot be directly compared to the values appearing on the first line of Table 8.

3.4. NN Forecasting

We then used the models specified by the hyperparameter choices in Tables 7 and 8 to forecast price variations on the test set. We proceeded by training each selected model on the entire training *and* validation set, up to the epoch at which the training had stopped during the grid search. Table 9 reports the test set performances of NN models obtained in this way. Although they played no role in model training and validation, here we also report the values of the MAE and MAPE metrics obtained on the test set. We already observe that class B models always outperform class A models in forecasting prices for all the three products. Figure 9 displays the forecasts made by the best NN model and compares it to the actual price variations, in the case of product 3.

Appendix A provides results obtained with univariate deep learning models instead, enabling us to assess whether the use of more input data in deep learning models provides an advantage as compared to the other univariate models. As the results of Table 9 are comparable with those of Table A3, we conclude that the advantage of a multivariate versus univariate analysis is not significant in this study, and it does not alter the conclusions drawn by considering the results of the multivariate analysis directly. Therefore, in the following we will only refer to the results of Table 9.

Table 9. Performance of the fine-tuned deep learning models on the test dataset.

Product	**1**		**2**		**3**	
class	A	B	A	B	A	B
RMSE	0.0617	0.0613	0.179	0.162	0.219	0.200
MAE	0.0498	0.0511	0.130	0.126	0.157	0.150
MAPE	0.0299	0.0305	0.0174	0.0168	0.0221	0.0212
ME	0.0138	0.00138	−0.0767	−0.0321	−0.0260	0.0262

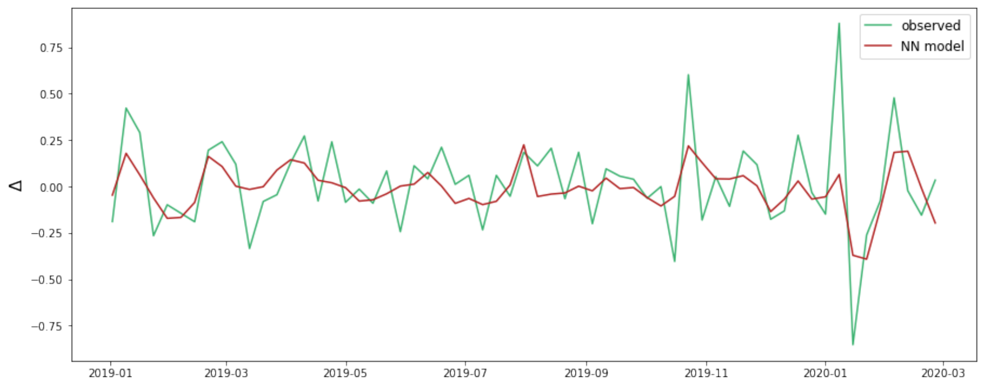


Figure 9. Comparison of NN forecasts and observed data for price variations in the test set, for product 3. The corresponding RMSE is 0.200.

3.5. Result Comparison

We can now compare the results obtained with the various models that have been tried. The comparison can be made by putting together the results contained in Tables 5, 6 and 9. For convenience, Table 10 summarizes the findings; for assessment purposes, we also include results of a *no-change* forecast model, in which the value of the price time series at the step ahead is predicted to be equal to the present value. In the same table we also report the average MAPE obtained by each model on the three products. Average MAPE can indeed be taken as an aggregate metric measuring model performance for the forecasting task considered in this work.

Table 10. Comparison of model performances on the test dataset. NN-A and NN-B indicate class A and class B neural network models, respectively, while *no-change* denotes the model where the price is forecasted to be equal to the latest value of the series.

Product	Metric	ARIMA	Prophet	NN-A	NN-B	No-Change
1	RMSE	0.0758	0.0812	0.0617	0.0613	0.0972
	MAE	0.0581	0.0694	0.0498	0.0511	0.0715
	MAPE	0.0348	0.0414	0.0299	0.0305	0.0429
	ME	0.00521	−0.0152	0.0138	0.00138	−0.0066
2	RMSE	0.173	0.220	0.179	0.162	0.268
	MAE	0.132	0.165	0.135	0.126	0.204
	MAPE	0.0178	0.0224	0.0181	0.0168	0.0276
	ME	0.0140	0.0456	−0.0767	−0.0321	0.00811
3	RMSE	0.215	0.350	0.219	0.200	0.354
	MAE	0.159	0.391	0.157	0.150	0.376
	MAPE	0.0222	0.0424	0.0221	0.0212	0.0425
	ME	0.0507	−0.217	−0.0260	0.0262	−0.0211
	avg MAPE	0.0249	0.0354	0.0234	0.0228	0.0377

We begin by noting that Prophet performances are considerably poorer than the ARIMA benchmark and neural networks. This is readily seen by looking at the last row of Table 10. From the same table we also learn that Prophet did not provide a very solid overall improvement with respect to the no-change forecasting model; in the case of product 3, performances were even comparable, signaling that the Prophet model could not model the given time series very well. On one hand, the fact that Prophet yielded the poorest forecasts could be expected as Prophet is based on a model with relatively low complexity that prioritizes ease of use and of tuning. On the other, as mentioned in Section 3.2.2, Prophet was progressively fitted also on the test set, effectively using more data than the other models for the fit, so its performances have arguably benefited from that. Nevertheless, one point that needs to be stressed is that we were able to provide Prophet with the entire weekly price time series comprising all of 2013 data, with no need for particular data pre-processing. Therefore, although Prophet's performances were not brilliant for the problem studied in this paper, it could still be useful in certain contexts where quick, preliminary forecasts are needed.

Turning to the other models, we observe that those featuring both CNN and LSTM layers (class B) yielded the most accurate forecasts for all the three products. At the same time, tuning them required by far the longest time (approximately 20 hrs for each product using a NVIDIA RTX 3060 GPU), with moderate improvement on purely LSTM (class A) models: considering the RMSE metric, the improvement amounts to about 9% for both product 2 and 3, while for product 1 the two classes yield similar results (class B has lower RMSE but higher MAE and MAPE). On the other hand, class A models required a considerably lower computational effort (of the order of 30 min to select the model hyperparameters for each product). Notice that once the grid search was concluded, training the best model on the entire training plus validation dataset required only a few minutes, both for class A and B models.

ARIMA models performed well if we take into account both the achieved values of the metrics and time necessary for tuning. They were less accurate than class B models, yielding RMSEs that were 23% higher for product 1, 7% higher for product 2, and 8% higher for product 3. Similar considerations apply if we instead look at MAE and MAPE metrics. However, ARIMA required about the same tuning time as class A models, and performed better than them for products 2 and 3: the RMSE obtained by ARIMA models was 23% larger for product 1, but 2% smaller for products 2 and 3.

4. Discussion

In this paper, we have discussed the application of different methods to the forecast of wholesale prices. We put a standard econometric model (ARIMA) side by side with two different approaches to time-series forecasting. These were rather diverse both in kind and complexity, going from a simple additive model using a piecewise linear trend and Fourier series seasonality (Prophet) to deep learning models featuring both convolutional and LSTM layers. The findings showed that while Prophet was quick to set-up and tune, requiring no data pre-processing, it was not able to come close to the performance of the other, well-established time-series forecasting models. Thus, it should be only preferred when simplicity and rapidity of the forecast are key aspects. Instead, we found that the best deep learning models performed better than ARIMA, but also required much longer times for the hyperparameter tuning. More precisely, the combination of LSTM and CNN layers obtained the best results, but did so at the expense of training time and computational resources; therefore, when needing reasonably accurate forecasts without taking times that can be of the order of one day, one could resort to pure-LSTM deep learning models, which offer a good trade-off and did better overall with respect to ARIMA models for the present task. We remark that ARIMA—like Prophet—is univariate, while a multivariate input dataset was used to train and test deep learning models. It would be possible to consider vector autoregressive (VAR) models [44,45] instead, as a multivariate generalization of ARIMA; however, our findings with the deep learning models revealed that the multivariate dataset is probably not so informative for the problem under study. We also point out that machine learning models were tuned over a larger parameter space than the others: the search grids were made of 54 and 8978 hyperparameter configurations for class A and class B NN models, respectively, versus a maximum of 5 ARIMA configurations and the 25 of Prophet. Note that even factoring out the three different choices of the timesteps number n, which are not really part of the NN models but refer to the way data are prepared, we are left with 2916 configurations for class B models. Another important aspect to consider in comparing the models is that dataset size strongly affects the performance of machine learning models. To this regard, we note that the sizes of the datasets were not very large (just over 200 for each product, as seen in Table 2), hence one could expect especially the NN model performance to further improve when more historical data are made available.

The work done in this paper can be extended in many directions. First, it would be interesting to carry out a similar analysis also for the forecasting of sales, and to consider data with daily frequency instead of weekly. Sales forecasting with higher frequency can indeed be relevant for wholesalers and retailers. Second, a more refined version of the study would distinguish between different customers, as the selling strategies adopted by the wholesaler do certainly differ when dealing with customers of various kinds and sizes. Customer profiling is an extremely interesting and active avenue of applied research, which can clearly enhance companies' returns. Therefore, in the near future we plan to carry on the analysis by combining customer classification algorithms with time-series forecasting: understanding how and when each customer buys, how specific products are treated differently by different customers (identifying *complementary* and *substitute* goods for each of them), as well as relating the price elasticity of demand for different product/customers, are all aspects that could lead to economic benefits for the wholesaler. To this end, one would want to compare further machine learning models to dynamic panel data where prices for each product and customer and price demand elasticity are considered. The approach could lead to an essential gain in the accuracy of the forecast, and would be an important contribution to sales prediction analysis, which is becoming an increasingly important part of modern business intelligence [46,47]. Another aspect of sure relevance would be to evaluate algorithms and models on data including the effects of the COVID-19 outbreak, to both learn how the market has transformed and help modifying selling strategies in adapting to the current rapidly changing situation.

The limits encountered in the application of the forecasting tools examined in this work encourage the evaluation of further models that could bring together the advantages of each approach. Finally, it would be relevant to apply one of the most exciting recent advances in machine learning, namely the attention mechanism [48], to the forecasting task we have considered. Work on this particular topic is definitely attractive and recently made its appearance in this field [49].

Author Contributions: Conceptualization, A.G. and M.M.; methodology, L.M., A.M., M.P. and C.M.; formal analysis, L.M., A.M. and M.P.; investigation, L.M.; data curation, A.B. and L.M.; writing—original draft preparation, L.M.; writing—review and editing, L.M., A.M. and C.M.; supervision A.G. and M.M.; project administration, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The paper is an output of the project "Cancelloni Big Data Analytics (BDA)" at Cancelloni Food Service S.p.A.

Data Availability Statement: Restrictions apply to the availability of these data. Data used in this work are property of Cancelloni Food Service S.p.A. and cannot be disclosed.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NN	Neural Network
LSTM	Long Short–Term Memory
NN	Neural Network
CNN	Convolutional Neural Network
GAM	Generalized Additive Model
ARIMA	Autoregressive Integrated Moving Average
ADF	Augmented Dikey–Fuller
BIC	Bayesian Information Criterion
ACF	Autocorrelation Function

PACF	Partial Autocorrelation Function
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percent Error
ME	Mean Error

Appendix A. Univariate Deep Learning Models

In this appendix we synthetically report the results of grid searching and forecasting with univariate deep learning models. As mentioned in the main text, this is done to provide an unbiased comparison with univariate ARIMA and Prophet models

Tables A1 and A2 contain the outcomes of the grid search, which was executed exactly as outlined in Section 3.3.1 for the multivariate models. Table A3 instead contains the performances of the selected models on the test set.

Table A1. Results of the grid search on class A (LSTM-only) univariate models.

Product	l	n_u	r	α	**Train RMSE**	**Valid RMSE**
1	2	96	0.2	0.001	0.116	0.0674
2	2	96	0.3	0.0005	0.195	0.183
3	2	64	0.3	0.001	0.224	0.162

Table A2. Results of the grid search on class B (CNN + LSTM) univariate models.

Product	l	n_u	r	α	f	k_s	**pad$_1$**	**pad$_2$**	n	**Train RMSE**	**Valid RMSE**
1	2	96	0.3	0.0005	30	2	valid	same	8	0.112	0.0600
2	2	96	0.1	0.001	10	2	causal	causal	12	0.188	0.179
3	1	64	0.2	0.001	30	2	same	causal	8	0.229	0.154

Table A3. Performance of the fine-tuned univariate deep learning models on the test dataset.

Product	**1**		**2**		**3**	
class	A	B	A	B	A	B
RMSE	0.0630	0.0591	0.186	0.166	0.226	0.207
MAE	0.0468	0.0449	0.131	0.122	0.155	0.158
MAPE	0.0281	0.0257	0.0175	0.0163	0.0221	0.0220
ME	0.00605	0.00747	−0.0981	−0.0654	0.0530	0.0262

References

1. Adhikari, R.; Agrawal, R.K. An Introductory Study on Time Series Modeling and Forecasting. *arXiv* **2013**, arXiv:1302.6613.
2. Petropoulos, F.; Apiletti, D.; Assimakopoulos, V.; Babai, M.Z.; Barrow, D.K.; Taieb, S.B.; Bergmeir, C.; Bessa, R.J.; Bijak, J.; Boylan, J.E.; et al. Forecasting: Theory and practice. *arXiv* **2021**, arXiv:2012.03854.
3. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
4. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181. [CrossRef]
5. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. A* **2021**, *379*, 20200209. [CrossRef]
6. Lara-Benítez, P.; Carranza-García, M.; Riquelme, J.C. An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *Int. J. Neural Syst.* **2021**, *31*, 2130001,
7. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
8. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
9. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw.* **1995**, *3361*, 1995.

10. Alibašić, E.; Fažo, B.; Petrović, I. A new approach to calculating electrical energy losses on power lines with a new improved three-mode method. *Teh. Vjesn.* **2019**, *26*, 405–411.
11. Kolassa, S.; Siemsen, E. *Demand Forecasting for Managers*; Business Expert Press: New York, NY, USA, 2016.
12. Taylor, J.W. Forecasting daily supermarket sales using exponentially weighted quantile regression. *Eur. J. Oper. Res.* **2007**, *178*, 154–167. [CrossRef]
13. Hofmann, E.; Rutschmann, E. Big data analytics and demand forecasting in supply chains: A conceptual analysis. *Int. J. Logist. Manag.* **2018**, *29*, 739–766. [CrossRef]
14. MOFC. The M5 Competition. Available online: https://mofc.unic.ac.cy/m5-competition/ (accessed on 2 September 2021).
15. Haofei, Z.; Guoping, X.; Fangting, Y.; Han, Y. A neural network model based on the multi-stage optimization approach for short-term food price forecasting in China. *Expert Syst. Appl.* **2007**, *33*, 347–356. [CrossRef]
16. Zou, H.; Xia, G.; Yang, F.; Wang, H. An investigation and comparison of artificial neural network and time series models for Chinese food grain price forecasting. *Neurocomputing* **2007**, *70*, 2913–2923. [CrossRef]
17. Jha, G.K.; Sinha, K. Agricultural price forecasting using neural network model: An innovative information delivery system. *Agric. Econ. Res. Rev.* **2013**, *26*, 229–239. [CrossRef]
18. Ahumada, H.; Cornejo, M. Forecasting food prices: The case of corn, soybeans and wheat. *Int. J. Forecast.* **2016**, *32*, 838–848. [CrossRef]
19. Pavlyshenko, B.M. Machine-Learning Models for Sales Time Series Forecasting. *Data* **2019**, *4*, 15. [CrossRef]
20. Livieris, I.E.; Pintelas, E.; Pintelas, P. A CNN–LSTM model for gold price time-series forecasting. *Neural Comput. Appl.* **2020**, *32*, 17351–17360. [CrossRef]
21. Zhang, L.; Wang, F.; Xu, B.; Chi, W.; Wang, Q.; Sun, T. Prediction of stock prices based on LM-BP neural network and the estimation of overfitting point by RDCI. *Neural Comput. Appl.* **2018**, *30*, 1425–1444. [CrossRef]
22. Thakur, G.S.M.; Bhattacharyya, R.; Mondal, S.S. Artificial Neural Network Based Model for Forecasting of Inflation in India. *Fuzzy Inf. Eng.* **2016**, *8*, 87–100. [CrossRef]
23. Paranhos, L. Predicting Inflation with Neural Networks. *arXiv* **2021**, arXiv:2104.03757.
24. Araujo, G.S.; Gaglianone, W.P. Machine Learning Methods for Inflation Forecasting in Brazil: New Contenders Versus Classical Models. Technical Report, Mimeo, 2020. Available online: https://www.cemla.org/actividades/2020-final/2020-10-xxv-meeting-cbrn/Session%202/3.%20Machine_Learning...%20Wagner%20Piazza.pdf (accessed on 10 August 2021).
25. Yan, X.; Weihan, W.; Chang, M. Research on financial assets transaction prediction model based on LSTM neural network. *Neural Comput. Appl.* **2021**, *33*, 257–270. [CrossRef]
26. Kamalov, F. Forecasting significant stock price changes using neural networks. *Neural Comput. Appl.* **2020**, *32*, 17655–17667. [CrossRef]
27. Hao, Y.; Gao, Q. Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. *Appl. Sci.* **2020**, *10*, 3961. [CrossRef]
28. Xiao, Y.; Xiao, J.; Wang, S. A hybrid model for time series forecasting. *Hum. Syst. Manag.* **2012**, *31*, 133–143. [CrossRef]
29. Qiu, J.; Wang, B.; Zhou, C. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLoS ONE* **2020**, *15*, e0227222. [CrossRef]
30. Chimmula, V.K.R.; Zhang, L. Time series forecasting of COVID-19 transmission in Canada using LSTM networks. *Chaos Solitons Fractals* **2020**, *135*, 109864. [CrossRef]
31. Wu, N.; Green, B.; Ben, X.; O'Banion, S. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *arXiv* **2020**, arXiv:2001.08317
32. Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time Series Analysis: Forecasting and Control*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
33. Hastie, T.; Tibshirani, R. Generalized Additive Models. *Stat. Sci.* **1986**, *1*, 297–310. [CrossRef]
34. Chan, W.N. Time Series Data Mining: Comparative Study of ARIMA and Prophet Methods for Forecasting Closing Prices of Myanmar Stock Exchange. *J. Comput. Appl. Res.* **2020**, *1*, 75–80.
35. Yenidoğan, I.; Çayir, A.; Kozan, O.; Dağ, T.; Arslan, Ç. Bitcoin forecasting using ARIMA and prophet. In Proceedings of the 2018 3rd International Conference on Computer Science and Engineering (UBMK), Sarajevo, Bosnia and Herzegovina, 20–23 September 2018; pp. 621–624. [CrossRef]
36. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 112. [CrossRef]
37. Taylor, S.J.; Letham, B. Forecasting at Scale. *Am. Stat.* **2018**, *72*, 37–45. [CrossRef]
38. Byrd, R.H.; Lu, P.; Nocedal, J.; Zhu, C. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.* **1995**, *16*, 1190–1208. [CrossRef]
39. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. 1310–1318.
40. Box, G.E.; Pierce, D.A. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J. Am. Stat. Assoc.* **1970**, *65*, 1509–1526. [CrossRef]
41. Ljung, G.M.; Box, G.E. On a measure of lack of fit in time series models. *Biometrika* **1978**, *65*, 297–303. [CrossRef]

42. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; Volume 9, pp. 249–256.
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Sims, C.A. Macroeconomics and reality. *Econom. J. Econom. Soc.* **1980**, *48*, 1–48. [CrossRef]
45. Hamilton, J.D. *Time Series Analysis*; Princeton University Press: Princeton, NJ, USA, 2020.
46. Mentzer, J.T.; Moon, M.A. *Sales Forecasting Management: A Demand Management Approach*; Sage Publications: Thousand Oaks, CA, USA, 2004.
47. Zhang, G.P. *Neural Networks in Business Forecasting*; IGI Global: Hershey, PA, USA, 2004.
48. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA; 2017; Volume 30, pp. 5998–6008.
49. Ekambaram, V.; Manglik, K.; Mukherjee, S.; Sajja, S.S.K.; Dwivedi, S.; Raykar, V. Attention based Multi-Modal New Product Sales Time-series Forecasting. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, 6–10 July 2020; pp. 3110–3118. [CrossRef]