Sales Forecasting for Pricing Strategies Based on Time Series and Learning Techniques

Jean-Christophe Ricklin[1,2], Ines Ben Amor[2], Raid Mansi[2], Vassilis Christophides[1][a] and Hajer Baazaoui[1][b]
[1]*ETIS UMR 8051, CY University, ENSEA, CNRS, Cergy, France*
[2]*BOOPER, 59 Boulevard Exelmans 75016 Paris, France*

Keywords: Sales Forecasting, Machine & Deep Learning, Time Series, Retail, Pricing Strategies.

Abstract: Time series exist in a wide variety of domains, such as market prices, healthcare and agriculture. Modelling time series data enables forecasting, anomaly detection, and data exploration. Few studies compare technologies and methodologies in the context of time series analysis, and existing tools are often limited in functionality. This paper focuses on the formulation and refinement of pricing strategies in mass retail, based on learning methods for sales forecasting and evaluation. The aim is to support BOOPER, a French startup specializing in pricing solutions for the retail sector. We focus on the strategy where each model is refined for a single product, studying both ensemble and parametric techniques as well as deep learning. To use these methods a hyperparameter setting is needed. The aim of this study is to provide an overview of the sensitivity of product sales to price fluctuations and promotions. The aim is also, to adapt existing methods using optimized machine and deep learning models, such as the Temporal Fusion Transformer (TFT) and the Temporal Convolutional Network (TCN), to capture the behaviour of each product. The idea is to improve their performance and adapt them to the specific requirements. We therefore provide an overview and experimental study of product learning models for each dataset, enabling informed decisions to be made about the most appropriate model and tool for each case.

1 INTRODUCTION

Time series analysis plays a key role in forecasting sales, particularly in the area of e-commerce, which is in full expansion. Interest in these methods is demonstrated by the participation of the world's largest retailer, Walmart, in Kaggle competitions. One of the most famous examples may be the M5 competition (Makridakis et al., 2022a), where it was observed that traditional methods used in retail, such as exponential smoothing or the Croston model, were outperformed by machine learning tools such as ensemble models like XGboost or LightGBM. In addition, there are many recent deep learning tools such as TCN (Temporal Convolutional Networks), TFT (Temporal Fusion Transformer) and DeepAR that further enhance the technical ability to extract temporal information. These techniques are often accompanied by AutoML (O'Leary et al., 2023) (Alsharef et al., 2022) tools (such as GluonTS (Alexandrov et al., 2020)) that provide them with a pre-built calibration solution. However, (Benidis et al., 2022) choose to train a local univariate model for each product. Based on our experiments, AutoML (Waring et al., 2020) combined within multiple deep learning models takes too long to go into production for local prediction when tasked with searching within a broad enough ensemble to have a varied interpretation of the context. Nevertheless, training a model for each product provides a local understanding of the product's true elasticity, which is particularly important when trying to provide pricing advice. The challenge is therefore to maintain accurate sales knowledge while having information on the local elasticity of the product.

Our idea is to review the literature on existing solutions, build a reasonable number of models based on sales characteristics, and then reduce the hyperparameter space.

The present work concerns real data from BOOPER a company providing inventory management and price advice software for the retail sector. Its strategy is to collect customer data in order to pre-

[a] https://orcid.org/0000-0002-2076-1881
[b] https://orcid.org/0000-0002-2151-7397

Ricklin, J., Ben Amor, I., Mansi, R., Christophides, V. and Baazaoui, H.
Sales Forecasting for Pricing Strategies Based on Time Series and Learning Techniques.
DOI: 10.5220/0012434300003636
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 3*, pages 1060-1067
ISBN: 978-989-758-680-4; ISSN: 2184-433X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

dict sales fluctuations and estimate them on the basis of historical data. The main data challenges revolve around two significant factors: the considerable variability in sales series within store products, and the decision to implement a single product forecasting approach.
The study presented in this paper is part of a research line with a wider scope, aiming to : define an accurate one-month estimates of shelf supply; understand product elasticity, which helps to identify the most popular products based on seasonal and annual events, especially for promotional periods; identify price plateaus, price changes likely to dissuade a customer; know the right historical duration to feed the models.

The aim is to define the best model for each product, taking into account the sales context, such as price, seasonality, events, changes.

In this paper, we present the several conducted experiments to answer specific questions about model behaviour, such as to how does the size of lags influence the accuracy of Boosting models' forecasts? Are deep learning models relevant in the context of single-series analysis? Which model should be used? Is a combination of models of interest?

The rest of this paper is organised as follows. In Section 2, we present the preliminaries and related work. Then, in Section 3, we formalise the problem and detail the constraints. Section 4 is dedicated to the presentation of our methodology for studying pricing strategies, the numerical results and the interpretation of the obtained results. Finally, Section 5 concludes and presents our future work.

2 RELATED WORK

Most of the methods used in mass retailing are based on statistical estimation. In this paper, we will be mostly focused on the autoregressive approach. The principle is relatively straightforward. We want to construct a global prediction function f_h such that $z_{i,t+h} = f_h(z_{i,t}, \theta_{i,t})$.

2.1 Machine & Deep Learning Methods

The Seasonal AutoRegressive Integrated Moving Average -X (SARIMAX) model has historically been used by our company as a benchmark. It has the formula given by:

$$\begin{gathered}(1-\sum_{i=1}^{p}\phi_i L^i)(1-\sum_{i=1}^{P}\Phi_i L^{iS})(1-L)^d(1-L^S)^D y_t = \\ (1+\sum_{i=1}^{q}\theta_i L^i)(1+\sum_{i=1}^{Q}\Theta_i L^{iS})\varepsilon_t + \sum_{i=1}^{k}\beta_i x_{i,t}\end{gathered} \tag{1}$$

p and ϕ_i are the order and coefficient of the autoregressive (AR) part, d is the degree of first differencing involved (non-seasonal), q and θ_i are the order and coefficient of the moving average (MA) part, P and Φ_i are the order and coefficient of the seasonal AR part, D is the degree of seasonal differencing, Q and Θ_i are respectively the order and the coefficient of the seasonal MA part, S is the length of the seasonal cycle, L is the lag operator, y_t is the time series, ε_t is the error term, $x_{i,t}$ are the exogenous variables and the β_i are the coefficients associated with the exogenous variables. The ARMA part can be fixed and optimised using the Box-Jenkins method, as detailed in the (Hyndman and Athanasopoulos, 2018) book.

However, the parametric (S)ARIMA-X model is not the most appropriate as it can only identify linear relationships between variables and past values. In addition, the noise is rarely Gaussian when considering sales series. Then Croston method described in (Kaya et al., 2020) and some Poisson-based methods detailed in (Liboschik et al., 2017) are also interesting in this context. But these methods do not take into account the feature importance.

This transformation aims to treat past observations and other features derived from the time series as independent variables for predicting future values. This transformation is often performed using a technique called a ”rolling window”. For each data point, we create features from previous data points within a window of specified size. We can first introduce bagging techniques such as random forest (Breiman, 2001). The idea behind the bagging method is to put together many prediction trees to get a higher accuracy.

Then we need to look at some boosting methods (such as XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2018)). These have been particularly successful in competitions. XGBoost is the first boosting algorithm that gave as much choice as possible to deal with overfitting problems, which are common in boosting techniques. LightGBM is the faster algorithm in the boosting list, the key technics added in LightGBM are Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). CatBoost, which is slower, but can produce better results in some cases. Key techniques in CatBoost include the use of random permutations in training instances and symmetric trees, trees grown level by level extending all leaf nodes with the same split condition.

Significant improvements have also been made to deep learning methods, which are increasingly well suited to temporal tasks. DeepAR (Salinas et al., 2020) is designed to be autoregressive, meaning that it makes predictions by taking into account its own past predictions as part of the input data. TCN (Bai et al., 2018) extracts spatially invariant local relationships. Finally, TFT (Lim et al., 2021) attempts to combine the best of both approaches. Other architectures are designed to work with time series data, e.g. Nbeat, which has special processing for time series, and LSTM are built to find short and long dependencies.

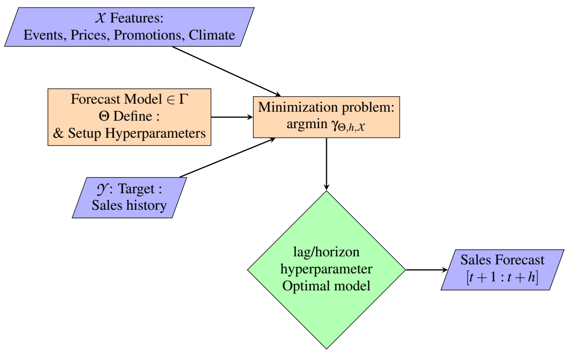
2.2 Improvement to Realize in Retail Sales Forecasting

There is a notable gap in the literature in the area of retail forecasting, especially when adopting a strategy that involves learning individual models for each product. There are few papers in the literature that propose a clear methodology on the conditions that lead to the use of a particular model. Indeed, when we apply the models in the literature to a diverse set of randomly selected products, it's difficult to find one model that always performs best. Even when tracking a single product, we find that the best model can change over the course of the product's lifecycle. Furthermore, in our optimisation processes, it is essential to prioritise models that have the greatest impact in relation to the context. Finally, it is essential to determine the appropriate training period for the models in order to know when to retrain them, and we don't find any advice on this in the literature.

In our research, we used machine learning and deep learning algorithms to predict sales series. This approach posed problems. When exploring extended hyperparameters using advanced optimisation tools, we were faced with time constraints. This problem can become more difficult, particularly with deep learning.

3 PROBLEM STATEMENT

To provide a clearer visualization and understanding of our problem, we will present the formalization of the process used to train and select the model within a function set, gamma, that we want to reduce. An overview is given in figure 1. First, we consider some models parameterized with an initial set of hyperparameters. We compare them to a target series with an initial horizon, related to a product sales series. The contextual features requested by customers, such as climate, prices (of the product and competitors), promotions and events, as well as some temporal features, are given with an initial lag. After a minimization on the convex lost function, we obtain a prediction with optimal training. It's easy to understand that the optimal training depends on the definition of gamma and the theta we have to reduce the space Γ (cf. figure 1).



$\mathcal{X}$ Features: Events, Prices, Promotions, Climate
Forecast Model $\in \Gamma$ Θ Define : & Setup Hyperparameters
Minimization problem: argmin $\gamma_{\Theta,h,\mathcal{X}}$
$\mathcal{Y}$: Target : Sales history
lag/horizon hyperparameter Optimal model
Sales Forecast $[t+1:t+h]$

Figure 1: An overview of the model selection process by product.

3.1 Problem Definition

Let $\mathcal{Y}$ be the set of time series of sales that exist in a store containing $i \in I \subset \mathbb{N}$ products.

Then we construct the application of $\Lambda : I \times T \longrightarrow \mathbb{R}$ such that $i,t \longrightarrow y_{i,t}$.

Let $\mathcal{X}$ be a set of features x_i that customers want to see in the model for their price management.

The set Γ of machine & deep learning functions in which we can find the best function $\gamma_{i,\theta,t}$, where θ is a vector of hyperparameters that predict the sales quantity $y_{i,t}$ of a product i at a given time t.

We also define the following minimization problem:

$$\text{argmin}_{\gamma_{\Theta,\mathcal{X}}} \mathbb{E}\left(\mathcal{L}(\gamma_{\theta_t}(\mathbf{x}_{i,[t+h:t-l]}, \mathbf{y}_{i,[t-1:t-l]}), y_{i,t})\right) \quad (2)$$

Where l is the number of lags and h is the horizon. $\mathcal{L}$ is a convex metric that will allow us to evaluate the performance of the models to be defined.

We rely on the notations given in the state of the art (Benidis et al., 2022) and position the current problem of the startup as a local univariate one.

Indeed, we only predict a single series of sales per model, associated with a unique product contained in a single store. We point out that the main need of clients is to obtain elasticity indicators.

3.2 Dataset Statistical Constraints

When studying sales time series, we often encounter periods with a high number of zeros and fluctuations in variance and trend. In order to identify the differences between products, it's essential to establish specific statistical indicators such as the ADI (Average Demand Interval) and the CV (Coefficient of Variation). Other treatments, such as outlier detection and change point analysis, can have a significant impact on the forecast. Another finding is that the time series may not be autoregressive.

The ADI is defined as follows:

$$ADI = \frac{\sum_{i=1}^{N} \tau_i}{N} \tag{3}$$

Where N is the number of sales days over the time interval (i.e. from the first day of sale to the last) $\tau_i = t_1 - t_0$ i.e. the distance between the first sale and the next sale, which in fine gives the full number of days.

The CV is defined as :

$$\varepsilon = \frac{\sum_{i=1}^{N} \varepsilon_i}{N} \tag{4}$$

ε_i is a number of sales. ε is the average number of sales on days when there were sales.

$$CV = \frac{\sqrt{\frac{\sum_{i=1}^{N} (\varepsilon_i - \varepsilon)^2}{N}}}{\varepsilon} \tag{5}$$

CV is therefore the variance of sales observed on sales days. In addition, by these two estimators, we can construct the (Syntetos and Boylan, 2005) Classification (SBC).

- intermittent series if CV < 0.49 and ADI > 1.32
- smooth series if CV < 0.49 and ADI < 1.32
- lumpy series if CV > 0.49 and ADI > 1.32
- erratic series if CV > 0.49 and ADI < 1.32

Table 1: Datasets presentation.

	Data Source		M5 Competition
Characteristics	Shop full time series data	Selected products	(Values)
Number of days	1296	1296	1913
Sales days	5694.6 have more than 360	All of them have more than 360	18598 have more than 360
Number of Series	37964	183	30 490
Number of Features	38	38	
Stationarity		65	
Change Point	Median = 2, Mean = 64	Median = 2, Mean = 2	
Outlier	Median = 1, Mean = 2.67	Median = 17, Mean = 17.6	
Seasonality	8732	47	
Autocorrelated	6454	89	
Intermittent	13890	50	26 956
Smooth	430	33	2 062
Lumpy	17412	50	6 416
Erratic	1466	50	1 791

We point out that, to test for stationarity, we use the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, which has the null hypothesis of series stationarity. For extreme values, we use the z-statistic, and for breakpoints, we use the function defined in (Killick et al., 2012). Booper has data regarding during the days on which the products recorded sales (sales geater than zero), for the missing days, we are unable at this time to determine whether it is a zero sales day or an error in the data recording of sales. So, we have to take zero for both. Another potential source of dataset constraint due to the absence of certain products in the store for a certain period of time, which could have a significant impact on the quality of the forecast.

We have also chosen not to delete days with very high product sales data (outliers), as they do not represent specific anomalies linked to higher hierarchical levels.

4 EXPERIMENTAL STUDY, RESULTS AND INTERPRETATION

This section presents our experimental study on product sales time series. First, we present our dataset and its preparation required for our experiments. Then, we detail the proposed machine learning and deep learning models and their hyperparameters. Finally, we present and discuss the results.

4.1 Experimental Datasets

Several datasets have been used during this study. The first was selected directly from a customer context. We then refined the choice of products to better understand the behavior of the model. We started with large-scale tests under commercial conditions. In the first experiment, we included 450816 products. These products were selected in 23 stores, so we made no selection on time-series behavior. The study was carried out in a store that handles general data, i.e, it sells a wide range of product types. These types are described in the section 3.2. This has enabled us to collect series with different behaviours. We randomly selected two hundred series of each product type. The types were classified according to the interval between two sales and the variance of average product sales. Then, for the first selection, the model was trained on a large number of stores. This enabled us to understand what happens in the real world and to generate hypotheses to check whether we wanted to adapt reality to the smaller test. We then carried out detailed experiments on a representative set of randomly selected products. This was done to understand how some criteria might affect the model's predictive performance. This study was carried out on a general data store, which means that it sells a variety of prod-

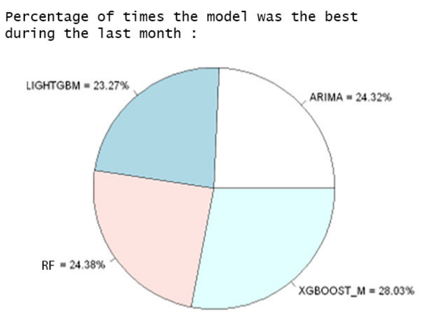


Figure 2: Graphs obtained with R libraries, using all company data, the best performance models.

uct types, which are described in the section 3.2. This allowed us to collect series with different behaviours.

To prepare the data and implement a sliding window strategy, we used the Darts library (Herzen et al., 2022). It makes it easy to determine the sliding window intervals. For the normalization, we use Sklearn's MinMax method, which is the most widely used, especially for the application of deep learning models. The normalization, step is essential to ensure convergence of the gradient descent. We also try the Yeo-Johnson function, which is interesting because it allows us to obtain a stationary time series, but it is less stable, so we don't select it in our experiments. We don't use normalization step for Boosting models.
To avoid data leakage, we evaluate model performance on an unknown test dataset corresponding to two last months of historical data (60 days). For deep learning models, we choose to divide the remaining data following the exclusion of the test dataset. into a training set corresponding to 865 days and a validation set consisting of 371 days. The first partition reduces overfitting, while the second is used for backpropagation.

The data, encompassing a broad range of product types, facilitated the collection of series exhibiting varied behaviors. We classified these types based on sales intervals and variance, randomly selecting 183 series per type for detailed analysis (cf. table 1). Utilizing the Darts library (Herzen et al., 2022), we implemented a sliding window strategy for data preparation. For normalization, Sklearn's MinMax method was employed, crucial for deep learning models' gradient descent convergence. However, we excluded the normalization step for Boosting models.

4.2 Experiments

Addressing the introductory questions, our two-step approach first hypothesized about boosting and parametric performance, focusing on a small set of hyperparameters for scalability. We then integrated deep learning models, selecting products with specific characteristics (intermittent, smooth, erratic, lumpy) to mimic real-world shop scenarios, as outlined in table 1. Products with minimal historical sales were excluded, especially when assessing time influence in boosting models. Our evaluation metrics included MAE, RMSE, MASE, and a monthly MAE reflecting aggregated monthly sales, aligning with large retailer assessments. These metrics, though not flawless, were applied to the experimental dataset described in section 4.1, aiming to identify the most suitable model for each product type.

4.3 Time Series Prediction and Experimental Results

Using the autoRank library (Herbold, 2020), we analyzed results statistically and graphically. This provided the necessary statistical justification for our results. In addition, we have mentioned the hyperparameter settings used for each experiment. The other tables show the mean rank (MR), the median difference from the best model (MED), the median absolute difference (MAD), the 95% confidence interval (CI), the effect size (γ) and the magnitude of the effect size for each model.

During the first experiment, we were using the following couples between models and hyperparameters:

Table 2: Setting Hyperparameters for the First Experiment.

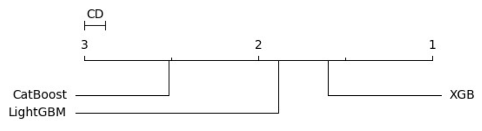
Model	**Parameters**
XGBoost	booster='gbtree', eta=0.4, gamma=0, max_depth=4, early_stopping=10
Random Forest	ntree=1000
(S)ARIMA-X	An automated version were (p,d,q)(P,D,Q) are found with the (Hyndman and Athanasopoulos, 2018) algorithm
LightGBM	nrounds=1000, metric='mae', objective='regression', boosting_type='gbdt'

Table 3: Analysis of Boosting Models' Performance.

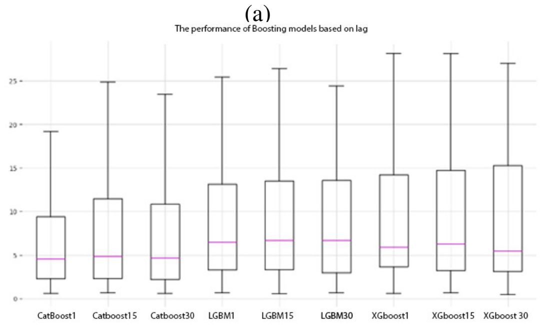
	MR	MED	MAD	CI	γ	Magnitude
CatBoost	2.516	3.221	1.845	[2.795, 3.852]	0.000	negligible
LightGBM	1.886	4.120	2.395	[3.579, 4.726]	-0.284	small
XGB	1.598	4.704	3.006	[4.032, 5.699]	-0.401	small

Figure 2 displays the distribution of top-performing models over the last month, indicating a balanced model rank variation. The pie chart shows no significant MAE differences across product types, using hyperparameters from Table 2.

Then we create model_Comp, with hyperparameters described in Table 4, selects the model with the closest absolute value prediction to the sales achieved in the previous month, based on three different algorithms. As in the first experiment, this model can vary. Extending our study, present findings on 183



(a)



(b)

Figure 3: (a) The average rank of the models, higher rank correlates with better performance. (b) The distribution of errors across models associated with the lag size.

randomly selected products, using Darts library defaults for boosting models. Lag size variation showed negligible impact on boosting models' MAE performance, confirmed by the Nemenyi test (cf. Figure 3 (b)). CatBoost, however, significantly outperformed others in RMSE. Figure 3 (a) evaluates errors in MAE across lags 1 to 30, with CatBoost leading in RMSE performance for the hyperparameter given in the Table 3. The table shows model performance dispersion, indicating varying effectiveness.

Then we compare the combination of the model in Table 4 with the machine learning model given in the Table 5. In the comparison, we obtained the result given in the Figure 4, using varied error metrics and sliding windows (nb_in = 30, nb_out = 15).

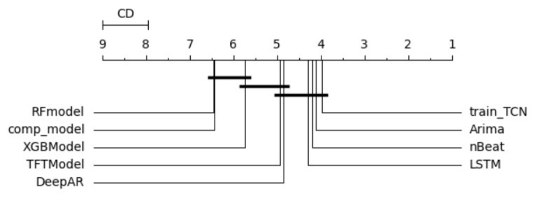
Table 4: Hyperparameters of Combined Models Used in the Company.

Model	Hyperparameters
XGBoost	objective='reg:squarederror', learning_rate={0.03, 0.05, 0.07}, reg_lambda={1}, n_estimators={1000}, max_depth={3,5,7}, min_child_weight={6,7,8,9}, gamma={0}, colsample_bytree={0.8}, eval_metric='mae'
Random Forest	n_estimators={1000}, max_features={8,10}, min_samples_leaf={4,6,8}, min_samples_split={8,10,12}, bootstrap={True, False}, max_depth={3,5,7}
(S)ARIMA-X	An automated version were (p,d,q)(P,D,Q) are found with the (Hyndman and Athanasopoulos, 2018) algorithm

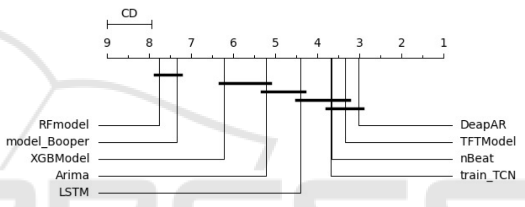
In terms of RMSE with the Nemenyi test highlighting key findings: (1) Negligible differences between RF and model_Comp, (2) Similar performance levels for XGBoost and (S)ARIMA-X, (3) No significant differences among Arima, LSTM, train_TCN, nBeat, and TFTModel groups. LSTM's lower MAD than ARIMA and DeepAR's lower CI upper bound than standard LSTM suggest varying model order

Table 5: Hyperparameter, Used to Obtain the Ranking.

Model	Parameters
TFT	hidden_size=64, lstm_layers=1, num_attention_heads=4, dropout=0.1, batch_size=16, n_epochs=50, add_encoders=None
DeepAR	model_RNN = RNNModel(model="LSTM", hidden_dim=(value needed), n_rnn_layers=2, dropout=0.2, batch_size=16, n_epochs=50, optimizer_kwargs="lr": 1e-3, random_state=0
LSTM	model_RNN = RNNModel(model="LSTM"), batch_size=16, n_epochs=50, optimizer_kwargs="lr": 1e-3, log_tensorboard=True, random_state=42, training_length=20,
N-BEATS	generic_architecture=False, num_blocks=3, num_layers=4, layer_widths=200, n_epochs=50, batch_size=9
TCN	n_epochs=50, dropout=0.1, dilation_base=2, weight_norm=True, kernel_size=7, num_filters=33, nr_epochs_val_period=1



(a)



(b)

Figure 4: (a) Model Ranking Based on the Company's Metric (b) Model Ranking According to RMSE.

(cf. Table 6 summarised in Figure 4(b)) Table 7 and Figure 4(a) underscore the significant impact that a change in metric can have on model rankings. when we consider the measure given by the company, we have the following results. The Friedman test again noted variations in model outcomes. Post-hoc analysis using Nemenyi test findings include: (1) No significant differences between RF model, model_Comp, and XGB Model, (2) No significant differences among TCN, ARIMA, N-BEATS, LSTM, DeepAR, and TFT, (3) Significant differences in other model pairings.
This implies that when other metrics are considered, the hierarchy of model performance may shift considerably.

4.4 Discussion

We are exploring ML/DL tools for time series analysis in sales forecasting, focusing on understanding the impact of pricing and promotion strategies. Our approach, tested on a comprehensive dataset encompassing various shops and products, revealed that

Table 6: Obtained results given in terms of RMSE.

	MR	MED	MAD	CI	γ	Magnitude
RFmodel	7.754	0.772	0.624	[0.426, 1.625]	0.000	negligible
model_Comp	7.345	0.921	-	[0.727, 5.211]	-	large
XGBModel	6.212	0.915	0.711	[0.500, 1.681]	-0.144	negligible
Arima	5.215	1.108	0.883	[0.601, 2.348]	-0.296	small
LSTM	4.392	1.237	0.816	[0.748, 2.508]	-0.431	small
train_TCN	3.677	1.246	0.876	[0.763, 2.545]	-0.420	small
nBeat	3.654	1.223	0.935	[0.726, 2.557]	-0.383	small
TFTModel	3.346	1.295	0.929	[0.866, 2.352]	-0.446	small
DeepAR	3.023	1.478	1.033	[0.825, 2.486]	-0.558	medium

Table 7: Obtained results (metric fixed by BOOPER).

	MR	MED	MAD	CI	γ	Magnitude
RFmodel	6.446	5.753	4.320	[3.401, 12.290]	0.000	negligible
model_Comp	6.435	5.770	4.688	[3.066, 14.634]	-0.003	negligible
XGBModel	5.742	6.304	4.936	[3.936, 17.543]	-0.080	negligible
TFTModel	4.938	12.476	11.068	[3.956, 25.123]	-0.540	medium
DeepAR	4.862	11.065	10.077	[4.733, 28.030]	-0.462	small
LSTM	4.300	13.246	10.792	[6.666, 26.225]	-0.615	medium
nBeat	4.192	12.132	10.550	[5.449, 28.506]	-0.534	medium
Arima	4.108	16.408	13.818	[7.041, 32.255]	-0.702	medium
train_TCN	3.977	13.906	11.024	[8.233, 26.195]	-0.657	medium

deep learning models require fine-tuning for optimal performance.In contrast, tree-based models like Catboost, LightGBM, and XGBoost showed promising results using Darts' default settings, with Catboost leading in accuracy (cf. Figure 3). We compared a hybrid ARIMA, Random Forest, and XGboost model with deep learning models. The results indicated that without meticulous hyperparameter tuning, deep learning models couldn't surpass ensemble methods, though they still performed well (cf. Tables 7 & 6). For instance, XGBoost ranked similarly to TFT and DeepAR in company metrics (cf. figure 4). However, in RMSE evaluation, custom deep learning models lagged behind ensemble models but outperformed (S)ARIMA-X (cf. table 6). This suggests two insights: (1) Optimal hyperparameter tuning is essential for deep learning models, as evidenced by TCN's good ranking and TFT's notable upper bound CI in tables 6 and 7. We observed instances where deep learning surpassed other methods. (2) Using a separate model for each product might not always yield favorable results with deep learning tools, suggesting a potential reduction in model variety (cf. figure 4). Our method focuses on minimizing loss on the training set, avoiding combined minimization on training and test sets to prevent data leakage. Future explorations could include recent Boosting models with contextual foundations and experiments with tweedie loss, especially for products with many zero values, as seen in the M5 competition (Makridakis et al., 2022b). Our next steps involve identifying specific hyperparameter configurations for individual products and narrowing the hyperparameter optimization scope. Understanding parameter influence is key to finding optimal settings.

A significant finding from our study is the profound impact of the metric chosen for evaluation. Specifically, outcomes can vary notably when assessed using MASE; this is especially evident for products with intermittent sales. Conversely, when employing the Comp-approved metric, there is an emphasis on products with high amount of sales. This discrepancy explains the notable difference in rankings produced by the two metrics. Therefore, selecting an appropriate metric is vital. In our setting, the ideal metric would be one that effectively highlights feature variations. A constructive direction for our research might be to devise a metric that considers this aspect.

5 CONCLUSION AND FUTURE WORK

In this study, we explored various machine learning tools for forecasting sales time series, aiming to identify models that are not only accurate but also sensitive to changes in pricing and promotions. Our methodology involved testing a range of models on real-life data, ensuring the generality of our results. We observed that boosting algorithms, particularly Catboost, performed well even with small lags. However, without hyperparameter optimization, deep learning models were not the most effective for sales forecasting. Our experiments indicated that the best model for the last month often matched the overall best model, suggesting stability in product-specific model adaptation. The next steps in our research include fine-tuning hyperparameters for individual products and narrowing the optimization scope. This approach will help determine if deep learning methods can meet the goal of "one model per product" and how they compare with autoTS algorithms. While a multivariate approach offers more data for potentially more accurate forecasts, it may reduce interpretability, which is crucial for advising clients on variable influences. Therefore, a high-quality, single-product approach ensures in-depth product understanding. Familiarity with established models from the literature allows us to use them efficiently, balancing accuracy with computational resource considerations.

ACKNOWLEDGEMENTS

This work has benefited from a CIFRE grant managed by the Association Nationale de la Recherche et de la Technologie, France (ANRT, n°2022/0326) and BOOPER.

REFERENCES

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., et al. (2020). Gluonts: Probabilistic and neural time series modeling in python. *The Journal of Machine Learning Research*, 21(1):4629–4634.

Alsharef, A., Aggarwal, K., Sonia, Kumar, M., and Mishra, A. (2022). Review of ml and automl solutions to forecast time-series data. *Archives of Computational Methods in Engineering*, 29(7):5297–5311.

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., et al. (2022). Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Herbold, S. (2020). Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173.

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., et al. (2022). Darts: User-friendly modern machine learning for time series. *The Journal of Machine Learning Research*, 23(1):5442–5447.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Kaya, G. O., Sahin, M., and Demirel, O. F. (2020). Intermittent demand forecasting: A guideline for method selection. *Sādhanā*, 45:1–7.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Liboschik, T., Fokianos, K., and Fried, R. (2017). tscount: An r package for analysis of count time series following generalized linear models. *Journal of Statistical Software*, 82:1–51.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2022a). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2022b). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364.

O'Leary, C., Toosi, F. G., and Lynch, C. (2023). A review of automl software tools for time series forecasting and anomaly detection. In *ICAART (3)*, pages 421–433.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

Syntetos, A. A. and Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of forecasting*, 21(2):303–314.

Waring, J., Lindvall, C., and Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial intelligence in medicine*, 104:101822.