MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: A planning solution for forecasting product sales with cannibalization

Author: Ferran Ibáñez Prat

Advisor: Sergi Zamora

Supervisor: Lesly Acosta

Department: Facultat de Matemàtiques i Estadística

University: Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

I would like to express my sincere gratitude to my mentor in Accenture, Sergi Zamora and to Joquin Gallego. Without their daily support this project would not have been possible.

Abstract

Predicting sales using cannibalization effects is a current major challenge. Several approaches deal with the estimation of cannibalization of new product launches or promotion effects but how to account for it when predicting future sales is still missing. For this reason, we aim to fill this gap by proposing a new framework based on time series causality as a method to identify potential candidates of causality. To attain such an objective, we use two state-of-art gradient boosting based algorithms, namely Extream gradient Boosting (XGBoost) and Light Gradient Boosting Machine (LGBM), as well as two Multi-step forecasting strategies. We show that the cannibalization approach together with Recursive forecasting provides more accurate forecasts respect to established benchmark models.

Keywords: Time Series, Cannibalization, XGBoost, LGBM, Sales Forecasting, Product Promotions, Machine Learning

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Nowadays product promotions represent a significant percentage of a firm's budget. Promotions are widely used to drive consumers' choice and there is strong evidence that they are an effective marketing strategy to increase short-term product sales (see Steenkamp et al., 2005 ; Ward and Davis, 1978), to sustain existing products in mature markets or to stimulate a new's product launching (Raju, 1995). In addition, they have become more and more important in the manufacturer and retailers' budget (Optimedia, 2017). Predicting future sales is a key point for any business budgeting and resource allocation, but the growing competition between products and brands and the increase of marketing strategies makes it a complex task. Thus, to capture the cross-category relationships between some products and the effect that they might have on each other's sales is a current major concern.

Another major challenge in demand forecasting, is taking into account the cannibalization effect. Any product manager needs to evaluate how much of the new demand of a product is due to cannibalizing the firm's other products rather than drawing from competitors or generating primary demand. Heskett (1976) defined product cannibalization as "the process by which a new product gains sales

by diverting them from an existing product". However, the impact that temporary product promotions have on the sales of other existing related products has also been referred as cannibalization (Heerde et al., 2004). In this study we will be exploring the existence of cannibalization due to temporary product promotion rather than the cannibalization due to the launch of new products.

Therefore, our objective is to forecast product sales considering the dynamic effects of cannibalization and we will do that by means of two gradient boosting techniques, Extreme Gradient Boosting and Light Gradient Boosting Machines. Gradient boosting techniques are know for its success in a wide range of practical applications (A. Bissacco and Soatto, 2007) and other works have already shown its great performance in the field of sales forecasting (see Pavlyshenko, 2019). Moreover, we will be exploring the application of two forecasting strategies that tackle the Multi-step ahead forecasting task, namely Recursive (or Iterative) Multi-step Forecasting and Direct (or Independent) Multi-step Forecasting.

This master thesis is organized as follows. We will begin with a little literature review to cover the existing research and to place in context our work. Then, Chapter 2 introduces the models used to generate sales forecasts. Chapter 3, provides a brief description of the data used, presents the methodology to identify and to account for cannibalization, and describes the different forecasting strategies as well as its implementation. Chapter 4 presents the results and discusses them. Finally, Chapter 5 concludes the work with a little summary and discussion.

1.1 Literature Review

Are product promotions successful? How do competitors react to price promotions and advertising? Actually, it has been shown that promotions are usually faced with more promotions and advertising is usually countered with more advertis-

ing instruments (Nijs et al., 2001). In addition, there exist several factors that significantly affect sales during promotions. We are talking about the size of the price cut (Christen et al., 1997); the frequency of the promotions (Christen et al., 1997); the promotion channels, e.g. TV, radio, etc. (Sethuraman and Tellis, 2002); the product category characteristics, e.g. if perishable or not (Baltas, 2005); and the competitors' promotions (Struse, 1987). In particular, this last one has to do with cannibalization and although we are all aware of this phenomenon it is typically less clear how to quantify the size of it. Lomax et al. (1997) proposed three different strategies to identify cannibalization effects and concluded that it is necessary to use multiple strategies to get a true picture. Aguilar-Palacios et al. (2021) they estimate cannibalization by means of time series causal Impact and in Nijs et al. (2001) it is used a Vector Autoregressive with Exogenous variables (VARX) approach with different promotion channels as exogenous variables. In the current literature, the use of multivariate time series models is more extended than the univariate ones because they can account for multivariate cannibalization within product categories or between product categories and, in particular, vector auto-regressive and state space models are the most widely used modeling approaches (see van Heerde et al., 2010).

On the other hand, when it comes to forecasting sales taking into account the cannibalization effects in a dynamic way the literature is almost negligible. Most of the current work is on the line of predicting future sales without putting attention on the effects of promotions (Pillo et al., 2016). For instance, Pavlyshenko (2019) estimated various Machine Learning models to forecast product sales and found that stacking outperformed several traditional and Machine Learning approaches, or in Ansuj et al. (1996), it is found that the forecasts obtained using Artificial Neural Networks (ANN) were more accurate than those of Autoregressive Integrated Moving Average (ARIMA) model with interventions. To our knowledge,

there exists few research about promotions in product sales forecasting. Pillo et al. (2016) adress this problem by estimating Support Vector Machines (SVM) under promotion impacts. However, they did not icluded promotions of other products to account for cross-brand or cross-category effects. Ali et al. (2009) showed that decision trees improved forecasting accuracy of simple time series techniques when feature combinations constructed from sales and promotional variables are used but, again, they did not account for cross-brand or cross-category effects.

Scope of the work

Previous research has shown that Machine Learning techniques are a serious contender to traditional statistical methods in demand forecasting (Gilliland, 2020. Machine Learning itself is capable of modelling complex dynamics and capturing a large number of features. However, the impact of cannibalization in product demand is not part of the feature set and they fail to capture these interactions between different products. This is why in this study we want to extend the existing research by creating a Machine Learning framework that takes into account the demand cannibalization due to product promotions. To do so, we will use the promotion effects as a proxy to identify the cannibalization effects and the cross correlation function (CCF) to identify the potenital cannibalization candidates. Then, for each Stock Keeping Unit (SKU), we will use a feature set that includes promotional and sales variables of other SKU which will allow us to account for cross-brand and cross-category promotional effects. Will compare the forecast accuracy of the Machine Learning models that include the cannibalization with the ones without the cannibalization together. Finally, we will test if recursive Multi-step forecasting outperforms direct Multi-step-forecasting in presence of cannibalization effects and without them.

CHAPTER 2

THEORETICAL FRAMEWORK

2.1 Models

2.1.1 Regression Trees

Tree based methods are non-parametric statistical learning methods used for classification or regression. The idea behind them is that they split the feature space into a set of partitions and then set a label to predict to each one. Decision Trees are based on a set of nodes and edges that are organized in an hierarchical form. They have a root node, internal nodes and terminal nodes and in root and internal nodes we test a boolean function applied to the features. Those features are not predefined but they are automatically developed by the algorithm.

If we have a learning set with p inputs and n osbervations $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{X}_i = (x_{i1}, x_{i2}, ..., x_{ip})$, with a partition into K distinct and non-overlapping regions, $R_1, R_2, ..., R_k$ and we model the response as a constant c_k in each region, the corresponding regression model can be seen as a piecewise constant approximation (Hastie et al., 2001):

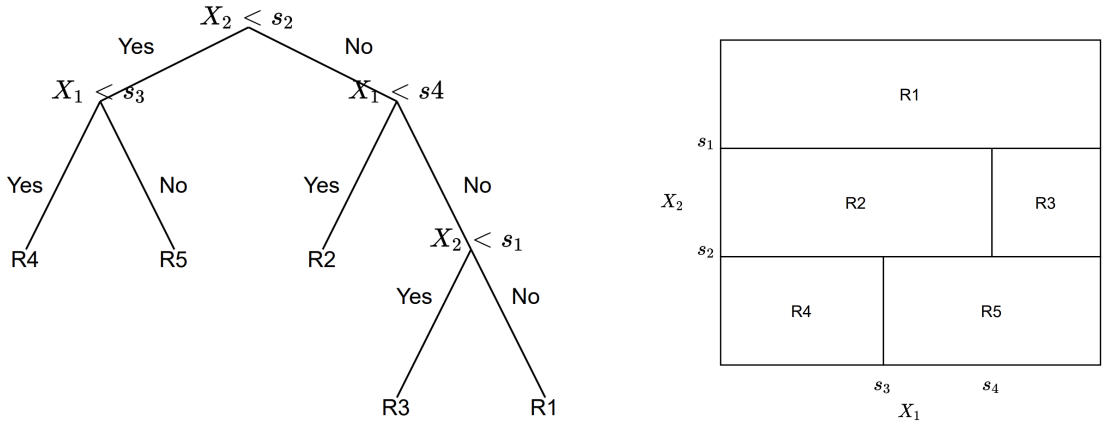

Figure 2.1: Decision Trees scheme

$$f(x) = \sum_{m=1}^{M} c_k I(x \in R_k)$$

For every observation that falls into R_k, the prediction is simply the mean of the response values for the training observations in R_k, i.e. $\hat{c}_m = average(y_i|x_i \in R_k)$. To find the best partitions $R_1, R_2, ..., R_k$, a top down binary approach known as binary splitting, starts at the root node and successively splits the predictor space into regions in such a way that it leads to the greatest possible reduction in Residual Sum of Squares, given by $\sum_{k=1}^{K} \sum_{i \in R_k} (\hat{y}_i - _{R_j})^2$, where $\hat{y}_{R_j}$ is the mean response within the kth region. In greater detail, if we consider the splitting variable j and split point s to define the pair of half-planes:

$$R_1(k, s) = \{x|x_j \leq s\} \quad \text{and} \quad R_2(k, s) = \{x|x_j > s\}$$

The best pair (j,s) is the one that solve:

$$\min_{j,s} \left[\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 . \right]$$

Once the best split has been found, the data is partitioned into two resulting

regions and the process is repeated until a stopping criterion is found. The stopping criteria has to do with the dimensionality of a tree and since it is an hyperparameter we will be tuning it to find the most optimal one. The key point here is how large should be a tree to optimally capture the important structure of the data without over-fitting it, so a smaller tree may work better. The preferred strategy for tuning the parameter consists on stopping the splitting process until some minimum node size is reached, and then, prune the tree by using the *cost-complexity pruning*, which consists on growing a large tree T_0, and then prune it to obtain a sub-tree. Basically, for each α it corresponds a sub-tree $T \in T_0$ that minimizes the penalized function:

$$C_\alpha(T) = \sum_{k=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{c}_k)^2 + \alpha|T|$$

where $|T|$ is the number of terminal nodes in T, R_k represents the region corresponding to the terminal node k. The tuning parameter $\alpha \geq 0$ controls the trade-off between a tree's complexity and and it's adequacy to the training data. With $alpha = 0$ the sub-tree T simply equals T_o, and large values of α will result in smaller subtrees. This can bee seen as a price to pay for trees with large numbers of terminal nodes. In addition, it can bee shown that for every α it exists a unique subtree T_o that minimizes $C_\alpha(T)$ (see Breiman et al., 1984). The optimal value of α can be approximated by cross-validation or by using a test sample in case our data set is very large.

2.1.2 Ensemble learning

Ensemble learning is a statistical learning methodology that combines several independent weak [1] predictors in order to improve the predictive performance. The principal types of ensemble methodologies are, **B**ootstrap **AGG**regating (Bagging), basically estimates multiple decision trees based on bootstrapped samples and then aggregates the results to obtain the optimal predictor; **Stacking**, typically uses different models fitted on the same train set, then each model generates a prediction that will be used as a feature for a second level model which is used to make predictions on a test set; and **Boosting**, which consists on adding sequentially new models to the ensemble, and at each iteration, a new weak based-learner model is trained based on the features of the whole ensemble learnt so far.

2.1.3 Gradient Boosting Algorithm

In practice, to obtain a solution of the parameter estimates given some arbitrarily cost function $C = (y, f)$ and the so-called base-learner $h(x, \Theta)$, can be so complex. Thus, Friedman (2001) proposed a new methodology that modifies the steepest descent algorithm by fitting a weak base-learner $h(x, \Theta)$ that is the most correlated to the negative gradient of the cost function $C(y, f) = \sum_{i=1}^{n} L(y_i, f(x_i)$, where y_i is our target and $f(x_i)$ the function that is supposed to model y_i given a learning set $S = \{(x_i, y_i)\}_{i=1}^{n}$, and being $(x_i \in R^k)$.

To have a better understanding of the algorithm, we can select the classic squared-error loss as a cost function: $C(y, f) = \sum_{i=1}^{n} (y_i - f(x_i))^2$. Then, the way to proceed in an iterative fashion is the following:

[1] According to Rokach(2010) , "A weak learner produces a classifier which is only slightly more accurate than random classification".

1. Initialize our estimate $\hat{f}_0$ with a constant.

2. For $m = 1$ to M, we do:

   (a) Compute the negative gradient vector

   $$g_m(x) = -\frac{\partial C(y, f)}{\partial f(x)} \quad at \quad f(x) = f_{m-1}(x)$$

   (b) Fit a new base learner function to the negative gradient

   $$h_m(g_m, x) = -\rho_m g_m(x)$$

   where the *step length* ρ_m is the solution to

   $$p_m = arg \min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}_{m-1} + \rho_m h(x; \Theta_m)$$

   (c) Update the estimate: $\hat{f}_m(x) = f_{m-1}(x) + \rho_m h_m(g_m, x)$;

   (d) End for

3. The estimate obtained is:

   $$\hat{y}_i = f_M(x) = \sum_{m=1}^{M} h_m(g_m, x) \qquad h_m \in H$$

where H represents the space of weak learners. Basically, the Gradient Boosting predicts the output $\hat{y}_i$ by fixing what we have learned at iteration m and adding a new weak-learner in each additional iteration. In the case of Gradient Boosting Decision Trees (GBDT), it will be used m additive functions to predict the output where H would be the space of regression trees. The main problem of GBDT is that the computation complexity increases proportionally to the number of features and instances, so it makes this implementation very time consuming if we handle a large feature set.

2.1.4 XGBoost

Extreme Gradient Boosting (XGBoost) is a gradient boosting algorithm that uses regression trees as weak learners and it is known for its good performance compared to other traditional methods. This algorithm was proposed by Chen and Guestrin (2016) and it is based on Friedman's Gradient Boosting algorithm but with a particular difference: it introduces a regularization term in the objective function that penalizes the model complexity to prevent overfiting. Thus, the regularized objective function is the following:

$$C = \sum_{i=1}^{n} L\left(y_i, f(x_i)\right) + \sum_{k=1}^{m} \Omega(h_k) \qquad where \qquad \Omega\left(h_k\right) = \alpha H + \frac{1}{2}\lambda \sum_{j=1}^{H} w_j^2$$

where L is a convex loss function that measures the difference between the prediction and the target; Ω is the regularization term that penalizes model complexity; H is the number of leaves in the tree. Each h_k corresponds to an independent tree structure j with weights.

Since the cost function introduces functions as parameters, it cannot be optimized by regular techniques, so the model must be trained in an additive manner, where $f(x_i)^m$ is the prediction of the i-th instance at the m-th iteration:

$$C^{(m)} = \sum_{i=1}^{n} L\left(y_i, f(x_i)^{(m-1)} + h_m(x_i)\right) + \Omega(h_m)$$

Moreover, in contrast to the use of the first order derivative in the Friedman's algorithm, a second-order Taylor series of objective function is adopted to quickly minimize the cost function:

$$C^{(m)} \simeq \sum_{i=1}^{n} L\left(y_i, f(x_i)^{(m-1)} + g_i h_m(x_i) + \frac{1}{2} k_i h_m^2(x_i)\right) + \Omega(h_m)$$

where g_i and k_i are the respective first and second derivative of the loss function.

Finally, if we remove the constants, we end up with the following simplified objective function which helps to quickly optimize the cost function in a general setting:

$$C^{(m)} \simeq \sum_{i=1}^{n} L\left(g_i h_m(x_i) + \frac{1}{2} k_i h_m^2(x_i)\right) + \Omega(h_m)$$

2.1.5 LGBM algorithm

The Light Gradient Boosting Machine (LGBM) algorithm was developed by Microsoft in 2016, (see Ke et al., 2017). It is also based on the Gradient Boosting Decision Tree (GBDT) Machine Learning algorithm. In contrast to XGBoost , LGBM tackles the problem of efficiency and scalability of GBDT by two novel techniques that work together, the so-called called *Gradient-based One-side Sampling* (GOOS) and *Exclusive Feature Bundling* (EFB). In fact, the training process of the LGBM is 20 times faster than a conventional GBDT while achieving almost identical prediction performances. GOOS overcomes the problem of data instances by only keeping the ones with greater gradients, that is those that contribute more to the information gain. Then, the information gain $\hat{V}_j(d)$ over the subset $A \cup B$ of splitting feature j at node d is:

$$\hat{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right)$$

where $A_l = x_i \in A : x_{ij} \leq d$, $A_r = x_i \in A : x_{ij} > d$, $B_l = x_i \in B : x_{ij} \leq d$, $B_r = x_i \in B : x_{ij} > d$ and $\frac{1-a}{b}$ is used to normalize the sum of the gradients overr B back to the size of A^c (Ke et al., 2017).

Simultaneously, EFB overcomes the model complexity by bundling features that rarely take nonzero values into a single feature (see Ke et al.,2017).

CHAPTER 3
METHODOLOGY

3.1 Data set description

Our study is based on a Kaggle data set (available here) of a large Ecuadorian-based grocery retailer called Corporación Favorita, which operates hundreds of supermarkets, with over 200,000 different products on their shelves. The purpose of Corporación Favorita was to challenge the kaggle community to build a model that accurately forecasts product sales so they could better ensure they please customers by having just enough of the right products at the right time. The data was initially structured in different files:

- **Train data:** inculdes the variables unit_sales, date, store_nbr, item_nbr and onpromotion which tells whether an item_nbr was on promotion for a specified date and store_nbr. It is important to underline the fact that we do not know the typology of the promotions, so these incentives can be either monetary or non-monetary.
- **Master data:** includes information about the type and if perishable for

each SKU . There are a total of 33 different types of items (Beauty, Grocery, Seafood, Eggs, etc.).

3.1.1 Data cleansing

Initially, the training data contained observations from 2013-01-01 to 2017-08-15 of 4036 different products shared across 54 different stores but due to the data set dimensionality, we shrank our scope to only those observations that belonged to store number 54. Basically, this store was the the one that contained a higher number of available promotions per product and, as we will see in section 3.2, promotions are a key point to account for cannibalization. Thus, we reduced the total number of rows from 125.597.040 to 1.648.867, the total number of SKUs from 4036 to 2600, and the total number of types from 33 to 30. Moreover, Grocery products are those that predominate in store 54, as can be seen in figure 3.1.1.

Missing values Treatment

Due to algorithm constraints, each product must have a non-intermittent time series free of missing observations. However, missing values in the sample appeared when we created a common time span. Basically, the dataset didn't included rows for those items that had zero unit sales for a particular data. Therefore, when we created a common time span for all of them, a considerable amount of missing values appeared. In particular, the full date span consisted of 1679 periods of information with 80% of the SKU sales series containing more than 500 missing values, i.e. the 80% of the SKU series contained $\geq 20\%$ of missing values. This can be observe in figure 3.1.1 which displays the cumulative distribution of the total number of missing values per SKU . At this point, we considered convenient to only keep those series that contained less than 3% of NAs in its sample. Consequently, we reduced the total number of SKU sales series to only 106. Furthermore, we

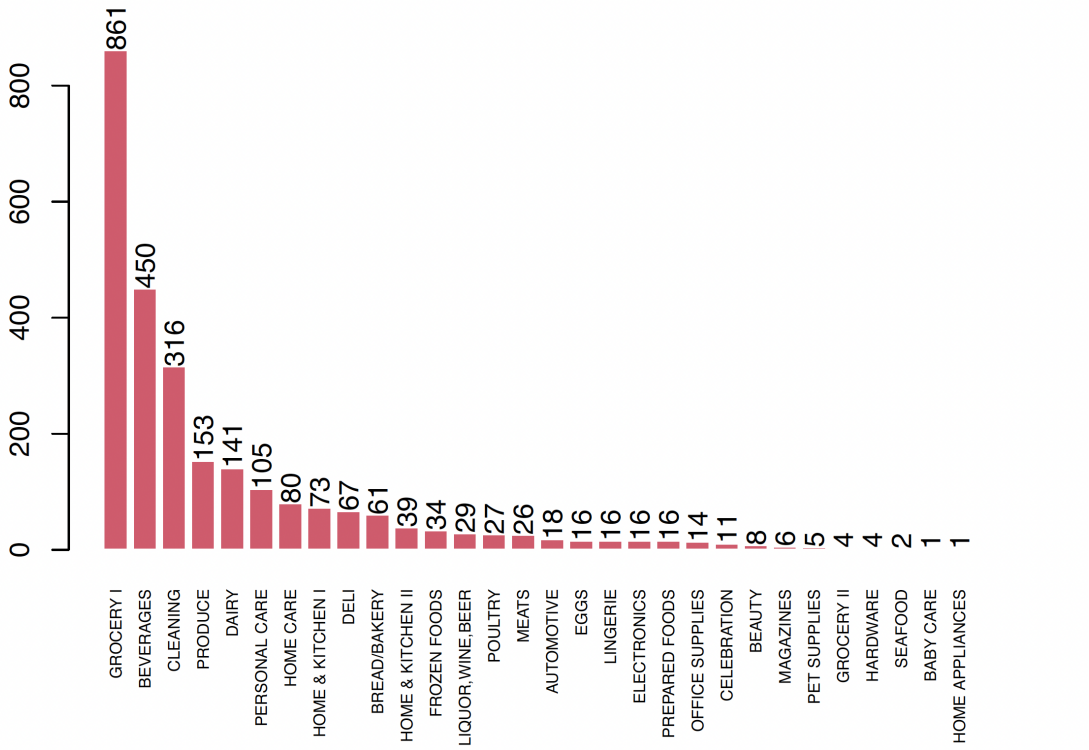

Figure 3.1: Total number of SKU per class in store 54

imputed zeros to all those missings.

Handling Outliers

Several SKU sales series presented some data points that were suspicious to be outliers, that is, they lie far from the rest of data points. It is crucial to handle these observations because they can affect negatively to the training process of Machine Learning algorithms. Our identification strategy consisted on finding a threshold for each SKU and, then, to all observations with values above the threshold, we assigned the threshold value. Ideally, each SKU series would need a particular strategy but considering that we had a portfolio of multiple SKUs we decided to apply a common approach to all. For each serie, the threshold

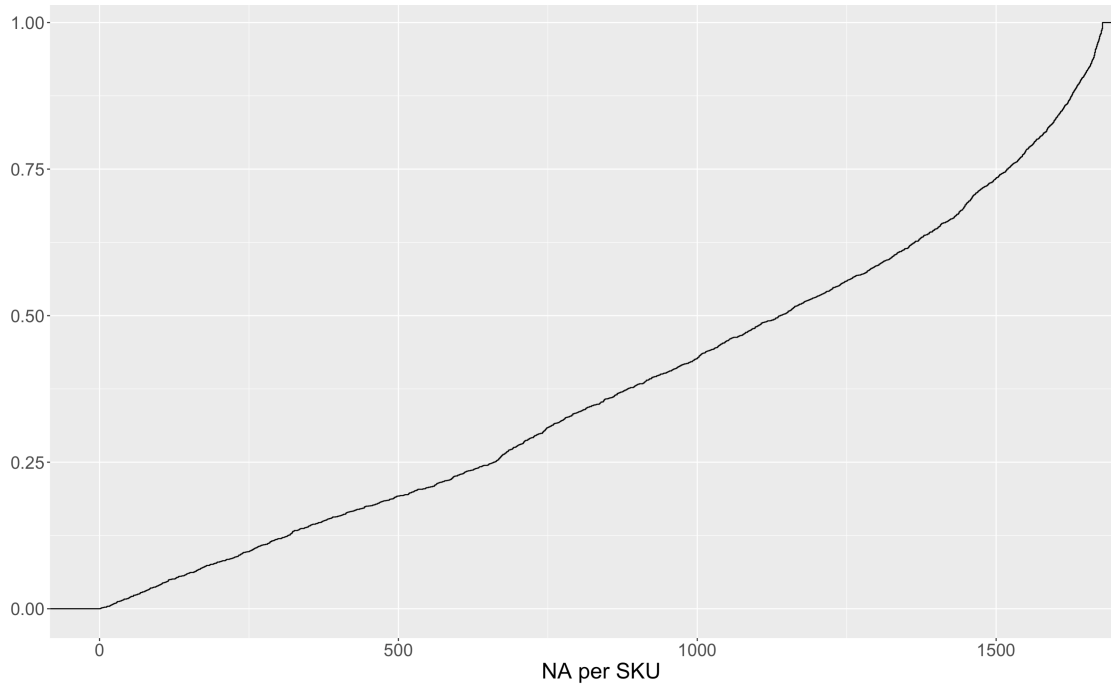

Figure 3.2: Cumulative distribution of the total number of missing values per SKU

was defined as $\bar{X} \pm 2.5S$, where $\bar{X}$ and & S are the respective sample mean and standard deviation of each SKU serie. Figure 3.3 displays an illustration using a specific SKU sales series (SKU 1036689) where the dashed line represents the current threshold for that SKU and the blue segments represent the data points where the observations were above the threshold.

Stationarity

A major concern when working with time series is stationarity. In strict sense, stationarity means that the joint probability distribution does not change over time, but if only the mean and variance remain constant over time, we talk about weak stationarity. To get an intuition of it, stationary ensures that the future is similar to the past and, hence, any prediction we make based on past infor-

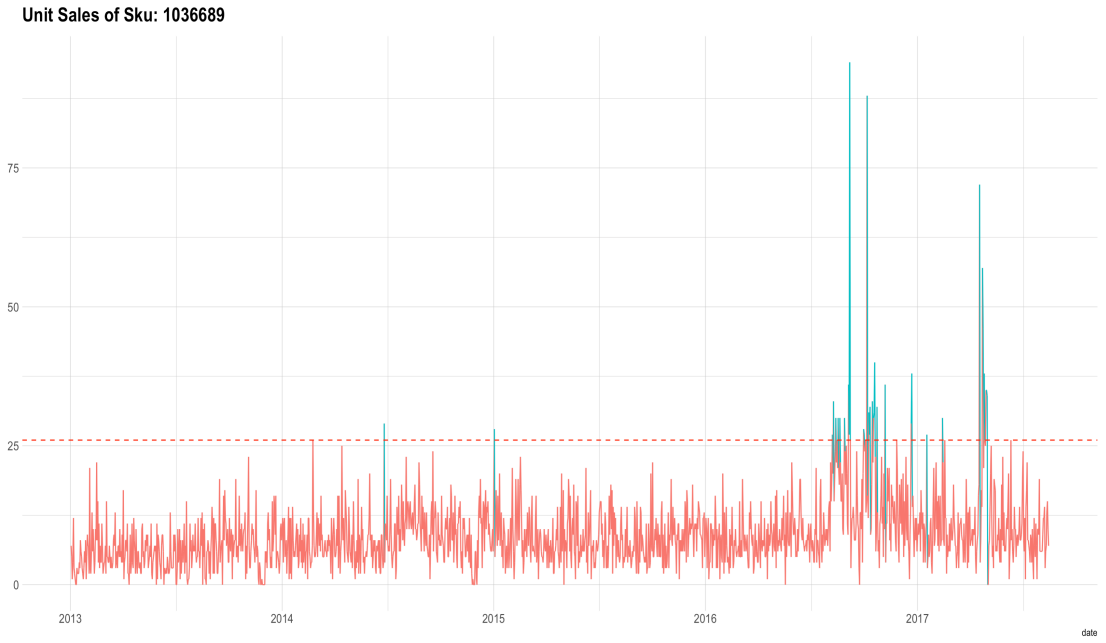

Figure 3.3: In blue are identified the unit sales that are above the threshold

mation, will be reliable. One implication of non-stationarity is that regressing a non-stationary series on one or more non-stationary series may lead to spurious regression. Fortunately, there exist many strategies to make a series stationary. Applying a logarithm transformation might help to stabilise the variance, and differentiation can help to stabilise the mean by removing trend and seasonality.

To test the stationarity condition we performed an Augmented Dickey-Fuller (ADF) test (Chatfield and Fuller, 1977) on each product sales series. In a nutshell, the ADF test basically tells us if our series contains a unit root or not. When there is presence of a unit root, the series has such a pattern that is unpredictable.

The results of the ADF test showed that none of the 106 SKU sales series had presence of a unit root. According to the test, all the series were already stationary so we did not have to apply any transformation.

Feature scaling

The last data pre-process consisted on the scaling of the time series. It has been proved that having features with the same scale helps Gradient Boosting algorithms to converge more quickly towards the solution. Therefore, we standardized our series. For each SKU series, we subtracted the mean (μ) and divided by the standard deviation (σ) each observation.

$$Z = \frac{X - \mu}{\sigma}$$

3.2 Cannibalization pipeline

Cannibalization is usually present on products within the same price range although is not limited to them. However, our dataset didn't contain any price information so we had to develop heuristics to identify potential cannibalization candidates based on their sales. Although we cannot directly observe cannibalization, we know that it may be produced as a result of product promotions. Thus, to account for dynamic cannibalization we will use the promotion effects as a proxy. Then, the set of regressors of each SKU series, will not only include its own lags and its own promotions but also the lagged series and promotions of other SKU series so we can capture the cross-product interactions. Moreover, only those SKU that can be potential candidates of causing cannibalization will be selected as predictors. This feature selection strategy comes to address the problem of dimensionality, as including all these series and promotions of other SKUs in the feature set, in turn, would introduce high dimensionality as a new hurdle. Then, to identify the presence of cannibalization between two series, we used an approach based on time series causality. We considered that it is more plausible the existance of cannibalization in causal series.

Cross Correlation Function

There are a couple of strategies to asses the directionality of causation between two time series. One is Granger Causality which is easy to apply and to understand. Basically, it tells us if one time series is useful to predict another one. However, we are more interested in finding the leader-follower dynamics and Granger does not tells us how shifted a time series must be to achieve full synchrony with another one. Thus, we made use of the sample cross correlation function. The idea behind the sample cross-correlation function (CCF) is that it tells us how shifted a series must be to make it synchronized to another. Given two time series y_t and z_t, the cross-covariance between them is:

$$\sigma_{zy}(T) = \frac{1}{n-1}\sum_{i=1}^{n}(z_{t-T} - \mu_z)(y_t - \mu_y)$$

Then the cross correlation is only a normalized version series of the cross covariance function:

$$r_{zy}(T) = \frac{\sigma_{zy}(T)}{\sqrt{\sigma_{zz}(0)\sigma_{yy}(0)}}$$

Therefore, if the peak of the cross correlation is achieved without lagging any series, it would mean that the sales of two products are synchronized in the contemporaneous moment.

For each product sales, we computed the CCF with respect to the rest of our products in the scope and we kept the peak of the absolute CCF . Then, only those series with an absolute CCF grater than 0.3 were selected to be in the feature set of a particular product and, in addition, the sales series were shifted according to the lag in which the peak of the CCF was achieved. Nonetheless, there were many CCF in which the peak was achieved at the contemporaneous moment, but considering that we cannot forecast the sales of a particular SKU at

moment t+1 using information of another SKU at t+1, we discarded the CCF at the contemporaneous moment. Thus, we guaranteed that all the series entering in the feature set were lagged; eee the pseudo-code of algorithm 1 in 16.

Algorithm 1 Cannibalization candidates

SKU:= set of all product sales
We define: $A \subseteq SKU$, $B \subseteq SKU$, $A \cup B = SKU$, $A \cap B \neq SKU$
for i in A **do**
    for j in B **do**
        for t in 1,...,6 **do**
            CCF := $|r_{ij}(T)|$
            L:= arg $\max_T$ $|r_{ij}(T)|$
            if CCF > 0.3 **then**
                j enters in the feature set of i lagged L times
                j's promotion enters in the feature set of i
                i enters in the feature set of j shifted L times
                i's promotion enters in the feature set of j
            end if
        end for
    end for
end for

3.2.1 Feature Engeneering

After the identification of cannibalization, the next step is to construct a feature set for each SKU . In figure 3.2.1 we display a description of the variables included in the feature sets. Each SKU included some feature extracted from its own series and a couple of product specific categorical variables. Basically, the feature sets

of all SKU only differ in the number of SKU sales series and promotion variables included in the feature set and, as we have seen in the previous section, this is determined by the CCF (see algorithm 16). Also, we included ten variables to identify, if present, the daily, weekly, monthly, quarterly and annual seasonality of the series. In particular, we used the sine and cosine functions to identify the seasonality patterns (see Fisher, 1993).

Features	Description	Type
Lags	lags 1 to 7, the 14th and the 21th	Numeric
Moving Average	Moving average based on the last 30 periods	Numeric
Moving Standard Deviation	Moving Standard Deviation over the last 30 periods	Numeric
Promotions	1 if a SKU was on promotion, 0 otherwise	Categorical
Type	Identifies the category in which the SKU belongs (e.g. grocery item, eggs, beverages,etc.)	Categroical
Sales of other SKUs	Lagged daily unit sales of SKU that are potential candidates of causing cannibalization	Numeric
Cosine day of week	From monday to Sunday	Seasonality
Sinus day of week	From monday to Sunday	Seasonality
Cosine day of year	From January 1 to December 31	Seasonality
Sinus day of year	From January 1 to December 31	Seasonality
Cosine week of year	From week 1 to 54	Seasonality
Sinus week of year	From week 1 to 54	Seasonality
Cosine month of year	From January to December	Seasonality
Sinus month of year	From January to December	Seasonality
Cosine quarter of year	From quarter 1 to 4	Seasonality
Sinus quarter of year	From quarter 1 to 4	Seasonality

Table 3.1: Feature set for each SKU

Take as example the feature set of the SKU 110512 which consisted of features extracted of its own series, a promotion variable identifying whether it was on promotion, the SKU type, if it was perishable or not, the respective series that

were causal and lagged according to the CCF , their respective promotion variables and the seasonality variables; they made up a total of 134 features.

3.3 Forecasting strategies

One we have reviewed the principal techniques and data in which this report is based, we will go trough the main forecasting strategies (see table 3.2). We will be testing Direct (D) using LGBM and and Recursive (R) forecasting using both XGBoost and LGBM, and with the approaches of cannibalization and no cannibalization. Thus, we will be testing a total of six models.

	Recursive forecasting		Direct forecasting
Cannibalization	LGBM_RC	XGBoost_RC	LGBM_DC
No Cannibalizattion	LGBM_RNC	XGboost_RNC	LGBM_DNC

Table 3.2: Proposed Models

XGBoost and LGBM forecasts with no cannibalization are used as a baseline for the forecasts that account for cannibalization. In this way, we could asses the performance of the models with cannibalization.

3.3.1 Forecasting

Our objective was to predict 10 periods ahead and two forecasting strategies will help us to accomplish it. They are the so-called direct Multi-step forecasting and Recursive Multi-step forecasting (see Chatfield and Weigend, 1994). We will entertain the two approaches in the following sections, asses how they work and the pros and the cons.

Recursive Forecasts

Recursive forecasting consists on using a one-step ahead forecast where the prediction of the previous step is used to predict the next period. However, the use of Recursive forecasting with multivariate time series is not trivial. If we have a vector of n endogenous variables $\mathbf{y}_t = [y_{1t}, ..., y_{nt}]$, a vector of m exogenous variables $\mathbf{x}_t = [x_{1t}, ..., x_{mt}]$ with information available until time T,a finite number of unknown parameters $\mathbf{\Theta}$, a vector of prediction functions $\mathbf{m}(\mathbf{y}, \mathbf{x}; \mathbf{\Theta})$ and a vector of errors ϵ_t, we can train a model that produces *one-step ahead* forecasts:

$$\mathbf{y}_t = \hat{\mathbf{m}}(\mathbf{y}_{t-1}, \mathbf{x}_t; \hat{\mathbf{\Theta}}) + \epsilon_t \quad for \quad t \leq T, \tag{3.1}$$

Then, from the estimated equation form 3.1 we can generate a forecast for $T + 1$:

$$\hat{\mathbf{y}}_{T+1|T} = \hat{\mathbf{m}}(\mathbf{y}_T, \mathbf{x}_{T+1}; \hat{\mathbf{\Theta}}) + \epsilon_{T+1},$$

Then,if we assume $\hat{\mathbf{y}}_{T+1|T}$ to be reliable we can produce a forecast for $T + 2$:

$$\hat{\mathbf{y}}_{T+2|T} = \hat{\mathbf{m}}(\hat{\mathbf{y}}_{T+1|T}, \mathbf{x}_{T+2}; \hat{\mathbf{\Theta}}) + \epsilon_{T+2},$$

and so on for future predicted horizons.

If we have information until T, the prediction of a particular SKU at T+2 would not be possible without having generated all the predictions at T+1 of all the SKUs in the scope. Although this process can be tedious, the reality is that we only have estimated one model, the rest just consists on updating the values. However, this approach accumulates errors in such a way that the performance of the model is degraded as the forecast horizon increases.

The implementation of Recursive forecasting applied to multivariate problems is not trivial. The key point of this approach, consists on updating the rows of

the regression matrix with the forecasts obtained at time t when we generate the forecast for t+1. To understand the implementation of Recursive forecasting, we are going to transform the time series problem into a regression problem. Consider a vector of n time series $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$, $\mathbf{y}_i = [y_{iS_i}, \ldots, y_{iT}]$, $for\ i = 1, \ldots, n$ where all time series end at T and start at S. Then, let $k_i = T - W - S_i + 2$ be the total number of rows of the regression matrix, where W is the number of columns. $T + S_i + 1$ is the time series length. Then, we can construct the regression matrix as the one that is displayed in table 3.3.1.

$\mathbf{Y}_i =$

Predictors				Dependent
Y_{i,S_i}	Y_{i,S_i+1}	$\cdots$	$Y_{i,S+W-2}$	$Y_{i,S+W-1}$
Y_{i,S_i+1}	Y_{i,S_i+2}	$\cdots$	$Y_{i,S+W-1}$	$Y_{i,S+W}$
Y_{i,S_i+2}	Y_{i,S_i+2}	$\cdots$	$Y_{i,S+W}$	$Y_{i,S+W+1}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$Y_{i,k-1}$	$Y_{i,k}$	$\cdots$	$Y_{i,T-1}$	$Y_{i,T}$
$Y_{i,k}$	$Y_{i,k+1}$	$\cdots$	$Y_{i,T}$	$\hat{Y}_{i,T+1}$
$Y_{i,k+1}$	$Y_{i,k+2}$	$\cdots$	$\hat{Y}_{i,T+1}$	$\hat{Y}_{i,T+2}$

Table 3.3: Regression matrix for recursive approach

As aforementioned, before generating the prediction at t for a particular SKU , we need all the predictions of all SKUs from the previous period. The rolling variables are also updated, but the rest, like the type of items or promotion variables, are treated as exogenous variables and, hence, are not updated.

The pipeline of recursive models without cannibalization is more straightforward. Here, the features matrix only consists on features generated by the own target series, its own promotions and external variables like perishables or type. Then, this process is computationally less complex and faster.

Direct forecasting

In contrast, Direct forecasting consists on directly estimating at the h-th period ahead forecast and, consequently, there will be as many estimation procedures as periods we are interested in forecasting. In addition, we can obtain a direct forecast for a specific period h without having to generate previous forecasts and, as a consequence, Direct forecasting does not aggregate errors. Forecasts of this approach are generated by:

$$\hat{\mathbf{y}}_{T+h|T} = \hat{\mathbf{m}}_h(\mathbf{y}_T, \mathbf{x}_{T+h}; \hat{\boldsymbol{\Theta}}_h) + \epsilon_t + T,$$

The problem of this strategy is the computational burden as the number of forecast horizons increases. Note that predictions are assumed independent between them when actually time series usually present dependency patterns.

The implementation of Direct forecasting architecture is reasonably more complex than recursive prediction. Then, the main complexity of this approach consists on correctly generating the training matrices for each model. The key point is that we can only use information until T-h to construct the explanatory features. The correct design of the regression matrix is illustrated at figure 3.3.1 (ma and Fildes, 2019). At each h, the predictors set remains equal while the dependent variable is updated.

The direct approach with cannibalization includes the sales series and promotion variables of other SKUs in the the predictors set. In contrast to the recursive approach, we do not need to update the predictors rows because they are invariant to h. On the other hand, the approach without cannibalization eliminates the sales and promotion variables of other SKUs from the predictors set.

$$\mathbf{Y}_i = \begin{array}{|cccc|ccc|} \multicolumn{4}{|c|}{\text{Predictors}} & \multicolumn{3}{c|}{\text{Dependent}} \\ \hline Y_{i,S_i} & Y_{i,S_i+1} & \dots & Y_{i,S+W-2} & Y_{i,S+W-1} & \dots & Y_{i,S+W+H-2} \\ Y_{i,S_i+1} & Y_{i,S_i+2} & \dots & Y_{i,S+W-1} & Y_{i,S+W} & \dots & Y_{i,S+W+H-1} \\ Y_{i,S_i+2} & Y_{i,S_i+2} & \dots & Y_{i,S+W} & Y_{i,S+W+1} & \dots & Y_{i,S+W+H} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Y_{i,k-H} & Y_{i,k-H+1} & \dots & Y_{i,T-H} & Y_{i,T-H+1} & \dots & Y_{i,T} \\ \hline Y_{i,k} & Y_{i,k+1} & \dots & Y_{i,T} & \hat{Y}_{i,T+1} & \dots & \hat{Y}_{i,T+H} \end{array}$$

Table 3.4: Direct forecasting approach regression matrix

3.4 Evaluation sets

A common task in Supervised Learning [1] prediction problems is the study and construction of algorithms that learn from input data. These input data are usually splitted in three sub sets, the Training set, which is used to fit the parameters; the Validation set, which provides an unbiased way to fine-tune the hyperparameters and evaluate the model fit on the training set; and the Test set, which provides an unbiased way to evaluate the performance of the final model.

Step Forward Cross-Validation

The usual approach to split the input data is the so called k-fold cross validation. However, when we evaluate the predictive performance of a time series algorithm it can only be used a training set that contains observations only prior to the observations that form the test set. If not, we would be breaking any time-dependency of the observations. Thus we will use a slightly different strategy that takes into

[1] Supervised learning is subcategory of Machine Learning and consists on training a function that classifies or predicts outcomes accurately.

account the time-dependency of the data and is called Step-forward CV approach. See figure 3.4 illustrating this approach.

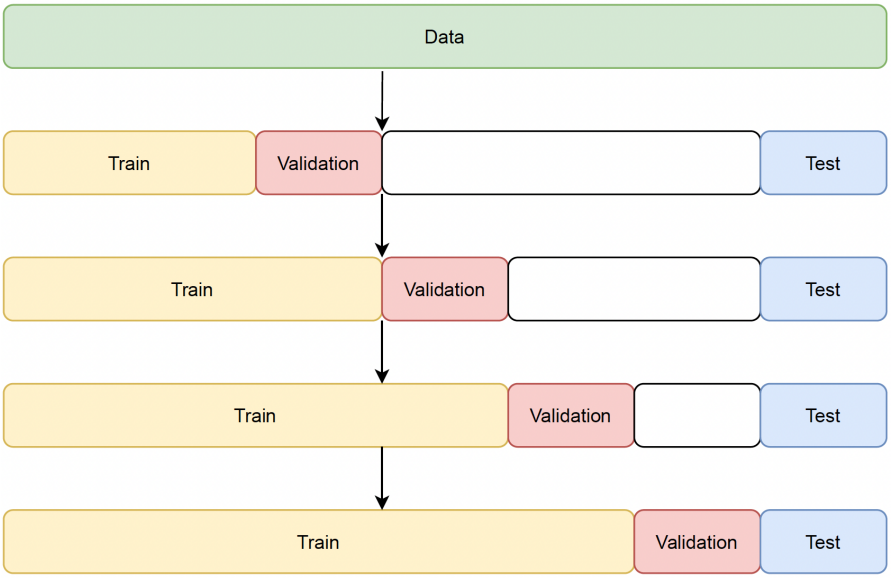

Figure 3.4: Step-forward CV

Our training set spans form 2014-04-01 to 2017-08-04, our validation set from 2017-07-26 to 2017-08-04, and our test set from 2017-08-14.

Hyperparameters optimization

Most of ML algorithms contain hyperparameters that need to be tuned and algorithms such as XGBoost and LGBM are not an exception. A correct optimization of hyperparameters is directly reflected in the performance of the model. To select the best set of hyperparameters we will use an Optimization Framework software called Optuna (see Akiba et al., 2019) which has a Python library that supports a variaty of optimization algorithms and is easy to use. Table 3.5 provides a brief summary of the specific meanings of the models hyperparameters and their respective search ranges. All the models we defined in 3.3 were specified with the same "hyperparametrization" but we are not providing their optimal values. Finally, we

should point out the fact that the search ranges are based on the linear domain of a uniform distribution and an Integer Uniform distribution for the max_depth of a tree.

Algorithm	Hyperparameters	Meanings	Search Ranges
LGBM	n_estimators	Number of trees	Unif(100-400)
	learning_rate	Shrinkage coefficient of each tree	Unif(0.001, 0.01)
	max_depth	Maximum depth of a tree	IntUnif(20-300)
XGBoost	n_estimators	Number of trees	Unif(100-400)
	learning_rate	Shrinkage coefficient of each tree	Unif(0.001-0.01)
	max_depth	Maximum depth of a tree	IntUnif(20-300)
	subsample	Subsample ratio of training samples	Unif(0.75-1)

Table 3.5: Hyperparameters description

3.5 Forecasting assessment

We used performance evaluation metrics at two different levels. First, we used metrics to tune the hyperparameters during the training and validation stages and to assess the models adequacy.

For both, and LGBM , the Root Man Squared Error is the objective metric to be minimized and will also be used to asses the performance of the model with different hyperparameters. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{3.2}$$

Then, the final model we will use to generate out-of-sample forecast will be the best of all models trained in terms of RMSE .

To evaluate our out-of-sample predictive power, we will use two metrics that are based on the Forecast Error. They are called Forecast Accuracy (FA) and Forecast Bias (FB). Actually, there are many variations of these two metrics, but we defined them in the following way:

$$FA = \min\left\{0, 1 - \frac{\sum_{t=1}^{n} |\hat{y}_t - y_t|}{\sum_{t=1}^{n} y_t}\right\} \tag{3.3}$$

$$FB = \frac{\sum_{t=1}^{n} \hat{y}_t - y_t}{\sum_{t=1}^{n} y_t} \tag{3.4}$$

In terms of Forecast Accuracy, the better models are the ones with values close to 1, and in terms of forecast bias, close to 0.

CHAPTER 4

RESULTS

In this section, we provide a discussion of the results. The evaluation is based on the out-of-sample forecast accuracy (FA) and the out-of-sample forecast bias (FB) metrics that we defined previously in equations 3.3 and 3.4, respectively. In tables 4.1 and 4.2, we show the Forecast Accuracy and Forecast Bias results. Please note that the metrics provided in these tables are based on the average of all the 106 SKUs covered by the study between the periods 2017-08-05 and 2017-08-14; the results for each SKU can be found in the appendix.

Table 4.1: Out-of-sample Forecast Accuracy

Forecasting Strategy	Model	Cannibalization	No Cannibalization
Recursive forecasting			
	XGBoost	0.5903	0.5361
	LGBM	0.5923	0.5665
Direct forecasting			
	LGBM	0.5196	0.5153

Note that $FA = \min\left\{0, 1 - \frac{\sum_{t=1}^{n} |\hat{y}_t - y_t|}{\sum_{t=1}^{n} y_t}\right\}$

The results show that in terms of FA, cannibalization models slightly outperform the models that do not include the cannibalization effects. In addition, whether cannibalization is considered or not, the recursive strategy always improves the results obtained by the direct approach. Likewise, Recursive forecasting generates forecasts with lower FB than Direct forecasting. Also, although the FB of XGBoost with cannibalziation is slightly lower than the XGBoost without cannibalization, in future analysis we would prefer the cannibalization approach since the improvement in is substantially higher than the reduction in FB .

Table 4.2: Out-of-sample Forecast Bias

Forecasting Strategy	Model	Cannibalization	No Cannibalization
Recursive forecasting			
	XGBoost	0.1	0.07
	LGBM	0.1	0.12
Direct forecasting			
	LGBM	0.16	0.15

Note that $FB = \frac{\sum_{t=1}^{n} \hat{y}_t - y_t}{\sum_{t=1}^{n} y_t}$

Moreover, we want to point out the fact that models that do not account for cannibalization are more sensible to seasonality patterns. We found out that they achieved higher values of both FA and FB in series that presented a strong pattern of weekly seasonality. In addition, the forecasts generated were very similar to the predictions generated by the models with the cannibalization approach. Take as an example the figure 4.1 which shows the *10-steps-ahead* forecasts using the recursive strategy. It can bee seen that both cannibalization (C) and no cannibalization (NC) models generated very similar forecasts. In this particular SKU , the LGBM without cannibalization effects was the one that achieved the best performance in terms of FA. The results of FA for this SKU are provided in table 4.3.

Table 4.3: Out-of-sample FA for SKU638977

Forecasting Strategy	Model	Cannibalization	No Cannibalization
Recursive forecasting			
	XGBoost	0.81	0.8
	LGBM	0.81	0.84
Direct forecasting			
	LGBM	0.64	0.61

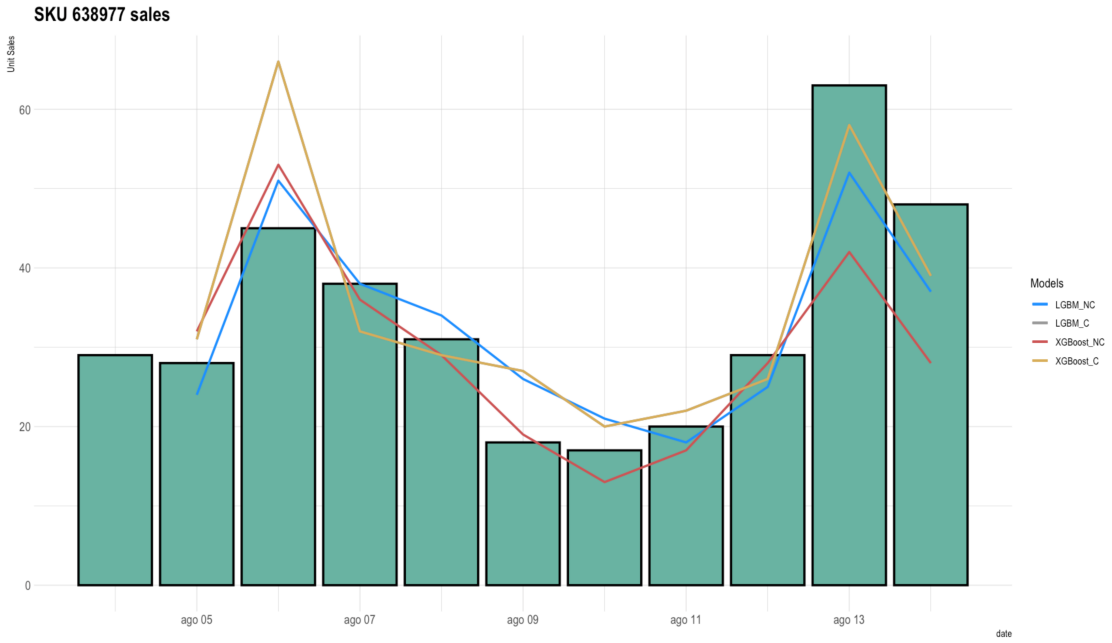

Figure 4.1: Out-of-sample FA for SKU638977

In contrast, the no cannibalization approach performed worse in series where the weekly seasonality pattern was not so significant. We can take as an example the figure 4.2 which shows the *10-step-ahead* forecasts using the recursive strategy. There, it can be seen that cannibalization models did a greater work compared to

the no cannibalization ones. The FA results for this particular SKU provided in table 4.4, show that the recursive XGBoost with cannibalization was the model that accomplished the best performance.

Table 4.4: Out-of-sample FA for SKU 876663

Forecasting Strategy	Model	Cannibalization	No Cannibalization
Recursive forecasting			
	XGBoost	0.79	0.71
	LGBM	0.77	0.63
Direct forecasting			
	LGBM	0.6	0.56

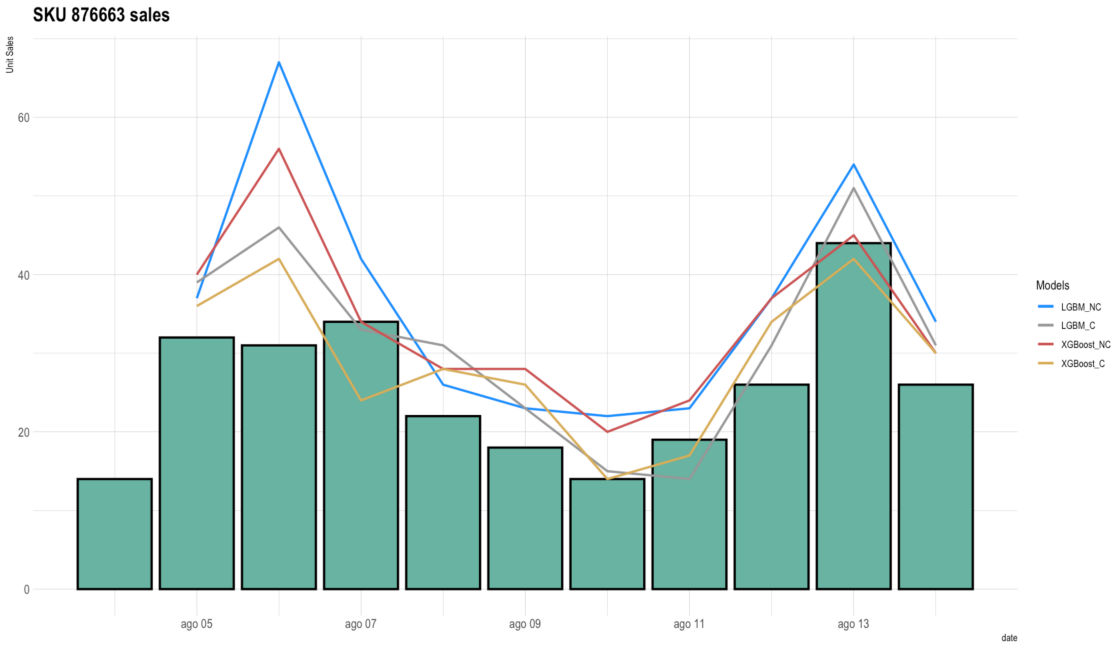

Figure 4.2: 10 steps-ahead forecast of the unit sales of SKU 876663

Finally, LGBM and XGBoost achieved similar results when they included can-

nibalization effects, either in FA and or FB. However, without the cannibalization effects the LGBM obtains better results in FA but worse in FB. In any case, we would prefer to use the LGBM model since it can generate almost identical predictive performance while the execution time of the training process is much lower.

CHAPTER 5

CONCLUSIONS

This project aimed to create a methodology able to account for cannibalization in a dynamic way. We have seen that the current literature in this field is scarce and that the majority of works are focused on estimating the cannibalization of a particular promotion or predicting future sales without including the cannibalization effects. Then, it was a major challenge to explore new approaches on how to deal with this effect while performing forecasting tasks. Along these lines, we have seen that cannibalization is not directly observed and we should use statistical tools to infer its impact. In section 3.2 we saw that since cannibalization effects are derived from promotions, it was comprehensible to consider the promotion variables as a proxy to capture the cannibalization effects. In addition, we considered time series causality to be a good start point when it comes to decide which products may be potential candidates of causing cannibalization. In particular, we considered CCF to be a consistent metric to determine the leader follower dynamics as well as being more appropriate to apply than other causality approaches. At the same time, the CCF was used to overcome the curse of dimensionality by only selecting those promotions that were potential candidates of causing cannibalization.

Using the method described above, we analyzed real market data featuring 2600 products from one store during 1200 days between 2013-01-01 and 2017-08-14. Moreover, we designed a Machine Learning framework in which two Gradient Boosting based algorithms, LGBM and XGBoost, could be able to account for cannibalization effects when it comes to forecasting sales. In addition, this framework was adapted to perform either Direct Multi-step forecasts or Recursive Multi-step forecasts. The results showed that cannibalization models achieved a higher predictive power in terms of forecast accuracy. In particular, they used to outperform the models with no cannibalization in SKU series where daily seasonality patterns were not very significant. Furthermore, we saw that the out-of-sample performance of Direct forecasting was considerably worse than Recursive forecasting. Finally, we saw that the performance of LGBM was slightly better compared to XGBoost and with a training process 20 times faster.

On the other hand, one possible limitation that our work may present is the lack of information about the promotions typology and the pricing information. In this line, conscious of the Corporacion Favorita dataset limitations, we believe that by conducting the same exercise with richer datasets that contain information about the promotions class and the products' pricing, it would be possible to contrast our results and, if the case, to consolidate the methodology we have proposed. Furthermore, we think it could be interesting to test our cannibalization framework in weekly or monthly data and with series that present different seasonality patterns, weekly, monthly or none.

Finally, we believe that the present work opens new lines of research when it comes to forecasting sales while capturing cannibalization. In particular, for future works that may tackle cannibalization, we are convinced that it would be interesting to perform a backtesting task to determine whether a particular SKU must be modeled using a cannibalization approach or not. This way, we believe that the overall accuracy of the forecasting exercise could improve substantially by selecting the best approach for each SKU.

REFERENCES

A. Bissacco, M. Y., & Soatto, S. (2007). Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. *IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1-8*. https://doi.org/10.1109/CVPR.2007.383129

Aguilar-Palacios, C., Munoz-Romero, S., & Rojo-Alvarez, J. L. (2021). Causal quantification of cannibalization during promotional sales in grocery retail. *IEEE Access*, *9*. https://doi.org/10.1109/ACCESS.2021.3062222

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ali, Ö. G., Sayin, S., Woensel, T. V., & Fransoo, J. (2009). SKU demand forecasting in the presence of promotions. *Expert Syst. Appl.*, *36*(10), 12340–12348. https://doi.org/10.1016/j.eswa.2009.04.052

Ansuj, A. P., Camargo, M. E., Radharamanan, R., & Petry, D. G. (1996). Sales forecasting using time series and neural networks.

Baltas, G. (2005). Modelling category demand in retail chains. *Journal of the Operational Research Society*, *56*(11), 1258–1264. https://doi.org/10.1057/palgrave.jors.2601972

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Routledge. https://doi.org/10.1201/9781315139470

Chatfield, C., & Fuller, W. A. (1977). Introduction to statistical time series. *Journal of the Royal Statistical Society. Series A (General)*, *140*. https://doi.org/10.2307/2344931

Chatfield, C., & Weigend, A. S. (1994). Time series prediction: Forecasting the future and understanding the past: Neil a. gershenfeld and andreas s. weigend, 1994, 'the future of time series', in: A.s. weigend and n.a. gershenfeld, eds., (addison-wesley, reading, ma), 1-70. *International Journal of Forecasting*, *10*(1), 161–163. https://EconPapers.repec.org/RePEc:eee:intfor:v:10:y:1994:i:1:p:161-163

Chen, T., & Guestrin, C. (2016). Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Christen, M., Gupta, S., Porter, J., Staelin, R., & Wittink, D. (1997). Using market-level data to understand promotion effects in a nonlinear model. *Journal of Marketing Research*, *34*, 322–334. https://doi.org/10.2307/3151895

Fisher, N. I. (1993). *Statistical analysis of circular data*. Cambridge University Press. https://doi.org/10.1017/CBO9780511564345

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Gilliland, M. (2020). The value added by machine learning approaches in forecasting [M4 Competition]. *International Journal of Forecasting*, *36*(1), 161–166. https://doi.org/https://doi.org/10.1016/j.ijforecast.2019.04.016

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Springer New York Inc.

Heerde, H. J. V., Leeflang, P. S., & Wittink, D. R. (2004). Decomposing the sales promotion bump with store data. *Marketing Science*, *23*. https://doi.org/10.1287/mksc.1040.0061

Heskett, J. (1976). *Marketing*. Macmillan. https://books.google.es/books?id=sO0TAQAAMAAJ

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Lomax, W., Hammond, K., East, R., & Clemente, M. (1997). The measurement of cannibalization. *Journal of Product Brand Management*, *6*. https://doi.org/10.1108/10610429710160011

ma, S., & Fildes, R. (2019). *Customer flow forecasting with third-party mobile payment data*.

Nijs, V. R., Dekimpe, M. G., Steenkamp, J. B. E., & Hanssens, D. M. (2001). The category-demand effects of price promotions. *Marketing Science*, *20*. https://doi.org/10.1287/mksc.20.1.1.10197

Optimedia, Z. (2017). Global Intelligence: Data and Insights for the new age of communication [Accessed: 2021-12-20].

Pavlyshenko, B. M. (2019). Machine-learning models for sales time series forecasting. *Data*, *4*. https://doi.org/10.3390/data4010015

Pillo, G. D., Latorre, V., Lucidi, S., & Procacci, E. (2016). An application of support vector machines to sales forecasting under promotions. *4OR*, *14*. https://doi.org/10.1007/s10288-016-0316-0

Raju, P. S. (1995). Consumer behavior in global markets: The a-b-c-d paradigm and its application to eastern europe and the third world. *Journal of Consumer Marketing*, *12*. https://doi.org/10.1108/07363769510147768

Rokach, L. (2010). *Pattern classification using ensemble methods*. World Scientific Publishing Co., Inc.

Sethuraman, R., & Tellis, G. (2002). Does manufacturer advertising suppress or stimulate retail price promotions? analytical model and empirical analysis. *Journal of Retailing*, *78*, 253–263. https://doi.org/10.1016/S0022-4359(02)00066-0

Steenkamp, J. B. E., Nijs, V. R., Hanssens, D. M., & Dekimpe, M. G. (2005). Competitive reactions to advertising and promotion attacks. *Marketing Science*, *24*. https://doi.org/10.1287/mksc.1040.0069

Struse, R. W. (1987). Commentary: Approaches to promotion evaluation: A practitioner's viewpoint. *Marketing Science*, *6*(2), 150–151. http://www.jstor.org/stable/183685

van Heerde, H. J., Srinivasan, S., & Dekimpe, M. G. (2010). Estimating cannibalization rates for pioneering innovations. *Marketing Science*, *29*. https://doi.org/10.1287/mksc.1100.0575

Ward, R. W., & Davis, J. E. (1978). A pooled cross-section time series model of coupon promotions. *American Journal of Agricultural Economics*, *60*. https://doi.org/10.2307/1239936

GLOSSARY

ADF Augmented Dickey-Fuller. 21

ANN Artificial Neural Networks. 8

CCF Cross Correlation Function. 9, 23–26

CV Cross Validation. 31

EFB Exclusive Feature Bundling. 16

FA Forecast Accuracy. 35, 37, 38

FB Forecast Bias. 33–35, 38

GBDT Gradient Boosting Decision Trees. 14, 16

GOOS Gradient-based One-side Sampling. 16

LGBM Light Gradient Boosting Machine. 1, 16, 26, 31, 32, 35, 37, 38

RMSE Root Mean Squared Error. 32, 33

SKU Stock Keeping Unit. 3, 9, 18–25, 27–29, 34, 35, 37

SVM Support Vector Machines. 9

VARX Vector Autoregressive with Exogenous Variables. 8

XGBoost Extreme Gradient Boosting. 15, 16, 26, 31, 35, 37

Appendices

APPENDIX A
RESULTS

A.1 Forecast Accuracy Results

A.1.1 Recursive Forecasts

Table A.1: Recursive forecasting: Forecast Accuracy Results

id	LGBM_RNC	XGBoost_RC	LGBM_RC	XGBoost_RNC
638977	0.843	0.813	0.813	0.804
223434	0.819	0.698	0.629	0.724
261700	0.801	0.750	0.756	0.840
258396	0.779	0.676	0.765	0.706
1084881	0.778	0.605	0.679	0.642
847863	0.762	0.525	0.534	0.686
1052563	0.757	0.656	0.732	0.519
903284	0.754	0.704	0.744	0.709
1036689	0.752	0.683	0.683	0.584
850333	0.744	0.733	0.767	0.567
261052	0.740	0.630	0.591	0.551
1087269	0.737	0.719	0.737	0.684
105576	0.734	0.677	0.677	0.734
457688	0.731	0.756	0.731	0.697
559493	0.728	0.668	0.679	0.565
155500	0.724	0.789	0.789	0.658
265559	0.721	0.721	0.744	0.643
866927	0.721	0.620	0.620	0.705
608035	0.712	0.606	0.606	0
939661	0.712	0.835	0.719	0.749
314384	0.704	0.708	0.708	0.669
357962	0.696	0.717	0.707	0.609
502331	0.694	0.746	0.730	0.778
671066	0.691	0.559	0.574	0.603
668752	0.689	0.623	0.672	0.672
828630	0.686	0.714	0.724	0.657
315277	0.683	0.667	0.663	0
357961	0.682	0.742	0.652	0.591
564533	0.674	0.590	0.611	0.632
275823	0.667	0.667	0.568	0.642
422452	0.662	0.541	0.568	0.473
830624	0.660	0.670	0.638	0.723
220432	0.658	0.611	0.617	0.772
414353	0.650	0.769	0.769	0.622
801934	0.650	0.625	0.525	0.325
208514	0.641	0.628	0.603	0.397
108797	0.641	0.699	0.699	0.592
260628	0.640	0.629	0.640	0.674
876663	0.628	0.793	0.774	0.714
848765	0.625	0.819	0.833	0.708
318932	0.617	0.553	0.660	0
759893	0.611	0.704	0.759	0.482
507870	0.600	0.556	0.578	0.511
364606	0.599	0.589	0.571	0.648
1071928	0.599	0.768	0.739	0.620
1146786	0.594	0	0	0.520
111223	0.593	0.370	0.389	0.556
464333	0.585	0.568	0.568	0.695
414426	0.579	0.579	0.702	0.614
215331	0.578	0.610	0.578	0.539
162066	0.575	0.632	0.627	0.611

Table A.2: Recursive forecasting: Forecast Accuracy Results

id	LGBM_RNC	XGBoost_RC	LGBM_RC	XGBoost_RNC
123601	0.561	0.561	0.584	0.422
115850	0.551	0.584	0.607	0.663
220435	0.545	0.653	0.636	0.446
570917	0.543	0.500	0.500	0.609
622958	0.530	0.561	0.561	0.621
1149579	0.526	0.671	0.630	0.590
621300	0.526	0.512	0.586	0.312
205381	0.511	0.574	0.532	0.468
165594	0.506	0.706	0.712	0.459
979195	0.500	0.583	0.479	0.625
158956	0.490	0.592	0.571	0.184
314393	0.490	0.438	0.521	0.729
464336	0.462	0.492	0.508	0.231
368136	0.460	0.351	0.386	0.540
364832	0.454	0.565	0.537	0.500
1146801	0.449	0.528	0.417	0.331
581078	0.448	0.396	0.515	0.366
574898	0.432	0.432	0.364	0.341
841612	0.431	0.552	0.534	0.500
114790	0.424	0.485	0.485	0.394
513853	0.424	0.525	0.576	0.339
648313	0.390	0.561	0.659	0.463
164647	0.370	0	0	0.030
1047396	0.361	0.417	0.333	0.417
949297	0.344	0.721	0.656	0
1105212	0.328	0.276	0.397	0.362
255161	0.319	0	0.191	0.298
877513	0.317	0.268	0.293	0.098
460804	0.273	0.242	0.182	0.242
830625	0.270	0.707	0.695	0
421066	0.262	0.357	0.286	0.167
214381	0.258	0.364	0.409	0.455
584123	0.250	0.438	0.438	0.312
1089845	0.232	0.366	0.339	0.214
1047772	0.227	0.413	0.400	0.120
830797	0.225	0.450	0.375	0
850460	0.191	0.191	0.191	0
564534	0.130	0	0.217	0.043
1047773	0.123	0	0.148	0
172343	0.086	0.517	0.379	0.517
315474	0.074	0.309	0.338	0
559870	0.044	0.367	0.222	0
598414	0.030	0.333	0.424	0.182
1047775	0.025	0.173	0.272	0.012
273528	0	0.507	0.522	0
315221	0	0.510	0.449	0
417763	0	0.083	0	0
463901	0	0.286	0	0

A.1.2 Direct forecasting

A.2 Forecast Bias Results

A.2.1 Recursive forecasting

Table A.3: Direct forecasting: Forecast Accuracy resuts

id	LGBM_DC	LGBM_DNC
638977	0.635	0.644
223434	0.457	0.543
261700	0.554	0.605
258396	0.426	0.485
1084881	0.691	0.630
847863	0.242	0.655
1052563	0.783	0.723
903284	0.764	0.719
1036689	0.624	0.663
850333	0.600	0.678
261052	0.323	0.465
1087269	0.719	0.596
105576	0.652	0.652
457688	0.681	0.664
559493	0.554	0.511
155500	0.621	0.646
265559	0.643	0.775
866927	0.535	0.543
608035	0.621	0.470
939661	0.678	0.584
314384	0.696	0.720
357962	0.707	0.707
502331	0.558	0.533
671066	0.485	0.559
668752	0.525	0.525
828630	0.562	0.533
315277	0.561	0.504
357961	0.470	0.515
564533	0.528	0.375
275823	0.605	0.580
422452	0.459	0.324
830624	0.702	0.638
220432	0.436	0.483
414353	0.734	0.769
801934	0.550	0.575
208514	0.218	0.167
108797	0.660	0.660
260628	0.472	0.461
876663	0.560	0.602
848765	0.708	0.708
318932	0.681	0.617
759893	0.323	0.401
507870	0.578	0.644
364606	0.625	0.671
1071928	0.479	0.465
1146786	0.305	0.411
111223	0.111	0.111
464333	0.703	0.627
414426	0.404	0.281
215331	0.597	0.721
162066	0.699	0.663

Table A.4: Direct forecasting: Forecast Accuracy results

id	LGBM_DC	LGBM_DNC
123601	0.358	0.462
115850	0.663	0.629
220435	0.603	0.678
570917	0.478	0.522
622958	0.561	0.455
1149579	0.584	0.595
621300	0.488	0.479
205381	0.766	0.723
165594	0.400	0.459
979195	0.583	0.583
158956	0.531	0.592
314393	0.458	0.302
464336	0.477	0.277
368136	0.485	0.525
364832	0.500	0.500
1146801	0.433	0.449
581078	0.537	0.522
574898	0.545	0.568
841612	0.517	0.500
114790	0.515	0.576
513853	0.186	0.237
648313	0.171	0.268
164647	0	0
1047396	0.417	0.389
949297	0.508	0.525
1105212	0.121	0.241
255161	0.426	0.426
877513	0.293	0.195
460804	0	0.212
830625	0.534	0.489
421066	0.310	0.452
214381	0.439	0.242
584123	0.562	0.375
1089845	0.250	0.286
1047772	0.293	0.187
830797	0.400	0.100
850460	0.128	0.170
564534	0	0
1047773	0.383	0
172343	0.448	0.569
315474	0.088	0.162
559870	0.322	0.233
598414	0.333	0.364
1047775	0.123	0.222
273528	0.420	0.420
315221	0.571	0.551
417763	0	0
463901	0	0.036

Table A.5: Recursive forecasting: Forecast Bias results

id	LGBM_RNC	XGBoost_RC	LGBM_RC	XGBoost_RNC
417763	0.917	1.083	0.625	2.833
315221	0.163	0.224	0.143	1.306
463901	0.893	1.107	0.964	1.214
1085686	0.459	0.514	0.541	1.135
273528	0.232	0.174	0.246	1.029
172343	−0.086	−0.241	−0.310	0.845
584123	0.625	0.438	0.562	0.750
850460	0.447	0.574	0.596	0.638
598414	−0.091	0	−0.091	0.606
559870	0.589	0.233	0.178	0.578
1047775	0.111	0.012	0.210	0.556
255161	0.532	0.532	0.596	0.553
830625	0.201	0.155	0.075	0.546
214381	0.576	0.258	−0.015	0.530
564534	0.391	0.522	0.435	0.522
949297	0.311	0.295	0.180	0.492
314393	0.552	0.271	0.479	0.490
460804	0.364	0.576	0.394	0.485
220435	0.198	0.281	0.306	0.446
165594	−0.047	−0.212	−0.182	0.435
574898	0.341	0.364	0.364	0.432
830797	0.600	0.100	0.125	0.425
648313	0.390	0.732	0.098	0.415
759893	0.288	0.428	0.210	0.389
158956	0.245	0.306	0.102	0.388
1071928	0.493	0.437	0.232	0.387
876663	0.346	0.425	0.180	0.372
315474	0.162	0.294	−0.044	0.368
1047396	0.333	0.194	0.278	0.361
318932	0.170	0.064	0.170	0.340
123601	0.237	0.376	0.150	0.335
570917	0.174	0.130	0.196	0.326
1149579	0.012	−0.116	0.116	0.324
220432	0.248	0.362	0.221	0.302
364606	0.176	0.176	−0.061	0.288
830624	0.085	0	0.319	0.277
421066	0.071	0.310	−0.095	0.262
1105212	0.552	0.810	0.190	0.259
357961	0.303	0.379	0.106	0.258
502331	0.104	−0.061	0.152	0.247
564533	0.431	0.194	0.139	0.243
1146801	−0.173	0.016	0.047	0.236
208514	0.654	0.654	0.295	0.231
314384	0.148	0.202	0.128	0.226
111223	0.741	0.667	0.463	0.222
1047773	0.667	−0.025	0.062	0.210
877513	0.707	0.512	0.268	0.195
1047772	0.813	0.653	0.493	0.187
668752	0.443	0.475	0.164	0.180
866927	0.395	0.419	0.225	0.171
513853	0.627	0.407	0.085	0.169

Table A.6: Recursive forecasting: Forecast Bias results

id	LGBM_RNC	XGBoost_RC	LGBM_RC	XGBoost_RNC
261052	0.394	0.614	0.409	0.165
848765	0.181	0.208	0.028	0.153
801934	−0.025	0.050	0.125	0.150
507870	−0.044	−0.022	−0.200	0.133
258396	0.132	0.250	0	0.132
841612	0.121	−0.069	−0.017	0.121
464336	0.262	−0.400	−0.308	0.108
422452	0.189	0.081	−0.270	0.095
847863	−0.184	0.749	0.413	0.094
979195	−0.250	0.042	0.062	0.083
1146786	0.240	0.420	1.020	0.082
261700	0.063	0.108	0.196	0.078
850333	0.144	0.311	−0.033	0.078
608035	0.348	−0.015	0.212	0.076
1036689	−0.040	−0.079	−0.178	0.069
164647	1.793	1.489	0.570	0.067
275823	0.173	0.099	0.160	0.062
457688	0.218	0.168	0.017	0.050
260628	0.180	0.191	0.135	0.045
223434	−0.095	−0.422	−0.147	0.043
115850	0.213	0.135	0.011	0.022
671066	0	0.132	−0.015	0.015
364832	0.185	0.037	−0.130	0.009
939661	0.371	0.067	−0.026	0.004
155500	0.138	0.163	0.065	−0.015
622958	0.212	−0.045	−0.015	−0.015
315277	0	−0.130	0.020	−0.024
638977	0.012	0.045	0.039	−0.033
215331	0.006	0.208	−0.084	−0.045
1084881	0.123	0.086	−0.198	−0.049
1087269	0.228	−0.070	−0.053	−0.053
265559	0.070	0.062	−0.070	−0.062
105576	0.133	0.070	0.171	−0.063
205381	−0.106	0.191	−0.213	−0.064
414426	0.228	−0.281	−0.193	−0.070
162066	0.119	−0.104	−0.062	−0.073
903284	0.192	0.069	−0.118	−0.118
828630	0.276	0.210	0.067	−0.124
559493	−0.196	−0.272	−0.212	−0.163
621300	−0.372	−0.335	−0.219	−0.186
1052563	−0.203	0.054	−0.225	−0.225
108797	−0.010	−0.029	−0.107	−0.243
1089845	−0.161	−0.125	−0.214	−0.268
357962	−0.207	−0.120	−0.207	−0.283
464333	0.034	−0.042	−0.314	−0.297
414353	−0.049	−0.112	−0.203	−0.308
368136	−0.386	−0.426	−0.356	−0.312
114790	0.030	0.242	0.061	−0.394
581078	−0.269	−0.194	−0.396	−0.418

A.2.2 Direct forecasting

Table A.7: Direct forecasting: Forecast Bias results

id	LGBM_DC	LGBM_DNC
417763	0.333	1.250
315221	0.327	1.735
463901	0.429	0.786
1085686	0.324	0.703
273528	0.348	1.971
172343	−0.379	−0.034
584123	0.562	0.688
850460	0.638	0.894
598414	−0.121	0.576
559870	0.256	0.600
1047775	0.259	0.543
255161	0.787	0.362
830625	0.132	1.149
214381	0.091	0.091
564534	0.696	0
949297	0.213	0.770
314393	0.458	0.021
460804	0.394	0.273
220435	0.231	0.512
165594	−0.082	0.518
574898	0.386	0.432
830797	0.050	0.550
648313	0.293	0.341
759893	0.226	0.518
158956	0.163	0.735
1071928	0.232	0.141
876663	0.102	0.286
315474	−0.044	−1
1047396	0.194	0.306
318932	0.277	1
123601	0.266	0.578
570917	0.239	0.261
1149579	0.179	0.145
220432	0.336	0.054
364606	−0.033	0.291
830624	0.245	0.170
421066	−0.167	0.452
1105212	0.069	0.328
357961	0.076	0.106
502331	0.231	0.186
564533	0.132	0.174
1146801	0.079	0.354
208514	0.244	0.474
314384	0.152	0.230
111223	0.370	0.333
1047773	0.012	0.444
877513	0.341	0.512
1047772	0.373	0.773
668752	0.246	0
866927	0.225	0.264
513853	0.169	−0.085

Table A.8: Direct forecasting: Forecast Bias Results

id	LGBM_DC	LGBM_DNC
261052	0.323	0.370
848765	0.014	0.069
801934	0.175	0.425
507870	−0.222	0.044
258396	−0.118	0.059
841612	−0.069	0.052
464336	−0.262	0.400
422452	−0.270	0.149
847863	0.430	0.117
979195	0.042	0.042
1146786	1.019	−0.209
261700	0.202	−0.009
850333	−0.022	0.278
608035	0.242	1.106
1036689	−0.218	0.139
164647	1.007	−0.200
275823	0.111	0.185
457688	−0.008	−0.134
260628	0.124	−0.056
223434	−0.129	0.155
115850	0.034	−0.112
671066	0.059	0.191
364832	−0.083	0.315
939661	0.030	−0.213
155500	0.065	−0.085
622958	−0.076	0.045
315277	−0.008	−1
638977	0.039	−0.119
215331	0.013	0.136
1084881	−0.198	−0.086
1087269	−0.070	−0.070
265559	−0.047	0.248
105576	0.158	−0.203
205381	−0.128	−0.404
414426	−0.175	−0.281
162066	−0.140	0.005
903284	−0.099	−0.113
828630	0.095	−0.057
559493	−0.147	−0.228
621300	−0.228	−0.363
1052563	−0.328	−0.406
108797	−0.107	−0.388
1089845	−0.205	−0.232
357962	−0.196	0.261
464333	−0.314	−0.186
414353	−0.189	−0.308
368136	−0.371	−0.262
114790	0.061	−0.152
581078	−0.500	−0.485