TIME SERIES FORECASTING VIA MACHINE LEARNING IN LARGE-SCALE E-COMMERCE USING TRANSFER LEARNING AND NONPARAMETRIC BOOTSTRAP

Máiron César Simões Chaves
mairon.chaves@olist.com
Olist - Data Science and A.I. Department

April 24, 2024

Abstract:

This study presents a forecasting approach in e-commerce using LightGBM and transfer learning to manage the complexity of a broad product catalog. Focused on optimizing the overall RMSE and individual WMAPE evaluation, the model demonstrated remarkable robustness: for SKUs with erratic demand, the median WMAPE was 36%, indicating reliability even in the face of volatile sales that represent 10.19% of the SKUs. Products with intermittent and lumpy demand, despite their inconsistencies, had median WMAPEs of 43% and 55%, respectively, demonstrating the model's adaptability. With exceptional performance, smooth demand, representing 16.96% of the SKUs, achieved a median WMAPE of 27%. The strategic use of non-parametric bootstrap to estimate confidence intervals improved the precision of the forecasts, highlighting the value of the approach for consistent and reliable predictions in large-scale scenarios.

Keywords Time Series · Transfer Learning · Retail · LightGBM · Forecasting

1. Introduction

Olist, a Brazilian platform that connects sellers to multiple marketplaces, faces unique forecasting challenges due to the diversity of SKUs it manages. The uniqueness of each SKU requires the robust forecasting system tailored to various demand patterns, as described by Croston (2005): smooth, intermittent, erratic, and lumpy.

We use ADI and CV^2 metrics to characterize and classify these demand patterns, which are crucial for adapting our machine learning algorithm, LightGBM. This is supported by feature engineering techniques to forecast sales, including days with no demand.

The continuous introduction of new SKUs and the heterogeneity of the data impose significant challenges, overcome through careful data selection. This process is validated and enriched by external references, such as the M5 Forecasting Accuracy competition.

Our methodology aims not only to meet Olist's demands but also to enhance demand forecasting practices in global e-commerce. Here, transfer learning emerges as a promising strategy, allowing a pre-trained model with existing SKUs to be quickly adapted to new products, providing reliable demand estimates from the beginning of their commercialization.

While predictive accuracy is crucial, it is equally important to quantify the associated uncertainty. In this work, we employ an adaptation of the block bootstrap method to build robust confidence intervals that capture the inherent variability of demand patterns and the operational peculiarities of Olist.

More about the company: Olist is a Brazilian company that acts as a vital bridge between small and medium-sized sellers and large marketplaces. Founded in 2015, the company revolutionizes e-commerce by facilitating the exposure of products from sellers of various sizes on high-reach platforms like Amazon and Mercado Libre, without the usual complexity and cost. Through the Olist platform, sellers can centrally manage their sales, logistics, and customer service, optimizing their operations and increasing their market visibility.

Beyond its core e-commerce activity, Olist is part of a larger group that includes other initiatives and companies focused on solving different challenges of digital commerce. This

diversification of services reinforces Olist's commitment to providing comprehensive e-commerce solutions, helping sellers succeed in an increasingly competitive market.

1. Methods

1.1 Classification of Demand Patterns

The research was guided almost entirely by the proposal of Syntetos, Boylan, and Croston (2005), where we differentiate product demands into four fundamental categories:

- **Smooth:** Regular sales in both time and quantity - easy for an algorithm to predict.
- **Intermittent:** Irregular sales in time, but regular in quantity - time series inflated with zeros due to long periods without sales.
- **Erratic:** Regular sales in time, but irregular in quantity - abrupt peaks/valleys.
- **Lumpy:** Irregular sales in time and quantity - highly complex to predict due to little or no pattern.

ADI (Average Demand Interval) and CV^2 (square of the coefficient of variation) are two metrics used to characterize and classify demand patterns. ADI measures the average frequency between periods of positive demand. The equation to calculate ADI in a daily time series is:

$$ADI = \frac{T}{N}$$

Where T represents the total number of observed periods (e.g., total days monitored), and N is the number of periods with positive demand (e.g., days when sales occurred).

CV^2 (Squared Coefficient of Variation) is a measure describing the relative variability of demand. It is calculated as the square of the coefficient of variation, which is the ratio between the standard deviation and the mean value of demand, as follows:

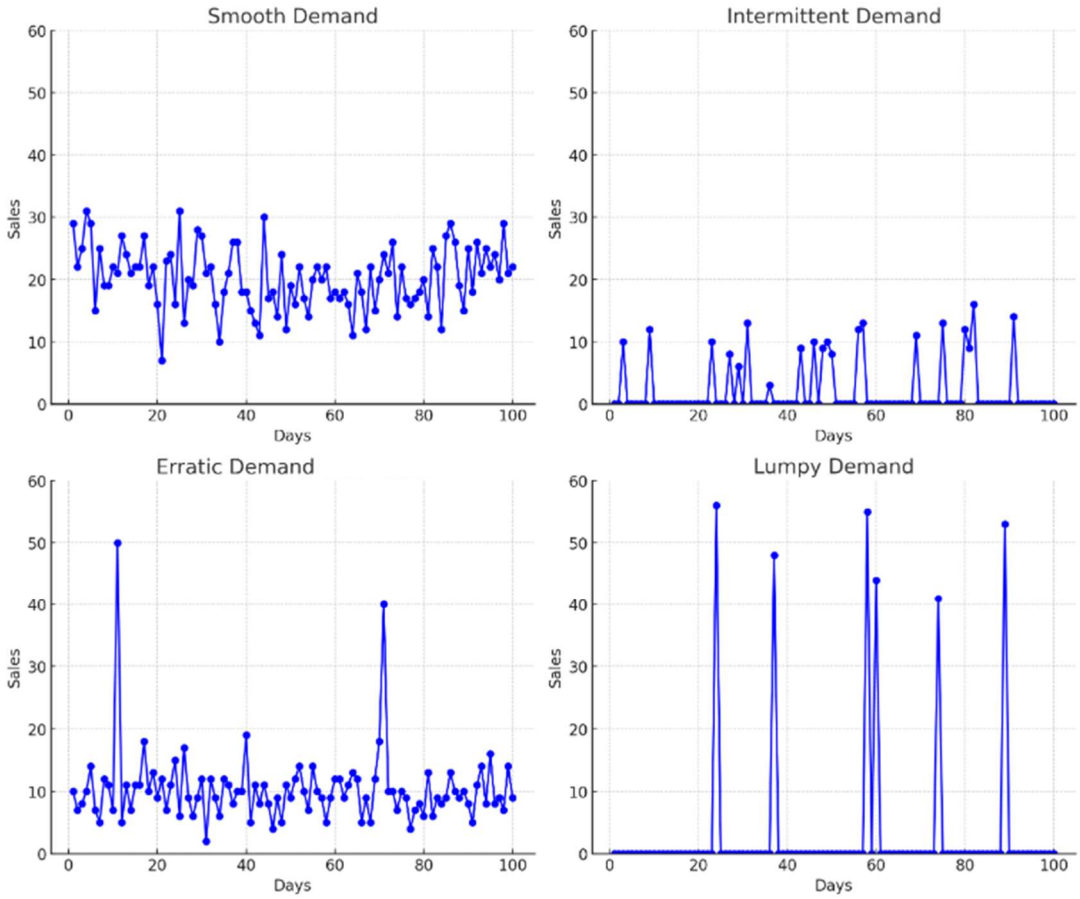
$$CV^2 = \left(\frac{\sigma}{\mu}\right)^2$$

where σ is the standard deviation of demand, and μ is the mean demand.

$$Demand\ Profile\ = \begin{cases} Smooth\ Demand & if\ ADI\ <\ 1.32\ and\ CV^2 < 0.49 \\ Intermittent\ Demand & if\ ADI\ \geq\ 1.32\ and\ CV^2 < 0.49 \\ Erratic\ Demand & if\ ADI\ <\ 1.32\ and\ CV^2 \geq 0.49 \\ Lumpy\ Demand & \text{otherwise} \end{cases}$$

Let's visualize a hypothetical graph in figure 1, showing how each profile typically appears in historical data:

Figure 1 - Retail's demand patterns



This classification serves as the basis for our modeling and analysis approach since each type of demand requires distinct forecasting techniques. In our case, the goal is to train a single machine learning algorithm on tens of thousands of historical data points, with a strong focus on feature engineering.

We carefully analyzed the four demand profiles and explored how we could create features to provide sufficient information for the algorithm to forecast days without sales for intermittent products, as well as handle abrupt variations in demand for lumpy and erratic profiles. For these

last three profiles, the use of confidence intervals is essential due to significant uncertainty, while point forecasts are suitable for products with a smooth demand profile.

1.2 Global and Local models

As presented by Montero-Manso and Hyndman (2021), global models, unlike local models, apply a single forecasting function to all series in the dataset, which may not be as limiting as local methods. In fact, global models can replicate or even exceed the forecasting performance of local models without assuming similarity among the series.

This is possible because global models can learn complex, long-term patterns that would only be available in local models through manual manipulations or specific assumptions. The local method personalizes the analysis by fitting a forecasting function to each time series in the dataset. On the other hand, the global method adopts a unified approach by applying a common predictive function to all time series in the dataset.

Contrary to initial perception, the global method reveals an equivalent ability to local methods in generating forecasts. This means that forecasts obtained through local methods can be achieved by a global model and vice versa. This equivalence is expressed as follows:

- Each time series is represented by X_i, where i identifies a specific series within the dataset.
- The forecasting function for the local method is denoted as $f_i(X_i)$, emphasizing that a specific function f_i is fitted for each individual series X_i.
- For the global method, we use the function $g(X)$, indicating a single predictive function g applied to all time series. Thus, we can represent the equivalence between the methods as follows:
    1. For each global function g, it's possible to identify a local function f_i that generates identical forecasts for the series X_i.
    2. Similarly, for each local function f_i, there's a global function g that produces the same forecasts for X_i.

This equivalence is a promising finding that paves the way for a smooth transition from traditional local methods, such as ARIMA and ETS, to more integrative global models. This

leads us to conclude that using a global method may offer a more efficient and scalable approach without sacrificing forecasting accuracy in time series datasets.
Therefore, one strategy is not necessarily better than the other, but one may be preferable over the other depending on the context. For professionals accustomed to local models, understanding global models may initially seem challenging. However, these models offer a significant advantage in terms of scalability and generalization capacity, especially useful in contexts with large datasets and multiple time series.

1.3 Data

The data used consisted of historical time series of sales from different companies within the group. All time series were processed at the daily granularity, but different levels of product hierarchy were utilized. Each cross-sectional unit is a GTIN (Global Trade Item Number) or SKU (Stock Keeping Unit). GTIN is a commercial item identifier developed and controlled by GS1, while SKU is a unique identification code created to organize and classify items in stock based on their characteristics. For simplicity, we will use the terms GTIN and SKU interchangeably throughout the article.

When stacking the data, the unique identifier for each product was renamed to 'unique_id'.

1.4 Selection of Representative Time Series

After consolidating the data from January 2021 to January 2024, the dataset contained approximately 20 million distinct products. Due to the challenges mentioned in the introduction, a methodology was required to select products whose historical series are representative in terms of variation and size.

The predominant presence of products in the so-called 'long tails' of the catalog leads to sparse data incompatible with sophisticated predictive models. Moreover, several factors prevent simply adding all SKUs to the training dataset:

- Products that performed well in the past but stopped selling months ago.
- Products with extreme intermittency, selling one unit today and another after several days.
- Products that sell one unit and then stop selling.
- Products that began selling recently, thus having very little historical data.

- Products with different start and end dates (series of different lengths).

Multiple filtering strategies were tested to select relevant time series for training, and we present the one that yielded the most representative sample. Here's how this process can be formalized:

Initial Dataset

Let *D* represent the initial dataset with entries $(i, t, y_{i,t})$ where:

- i is the unique identifier for each product (GTIN or SKU).
- t is the time index (date of sale).
- i,t is the quantity sold of product i at time t.

Filtering Criteria

1. **Observation Threshold Filtering:**

$$D' = \{ (i, t, y_{i,t}) \in D \mid \sum_{t \in T} [y_{i,t} > 0] \geq 30 \}$$

This defines the set D', which contains the tuples $(i, t, y_{i,t})$ extracted from the dataset *D*, where for each product i, the number of days with positive demand $y_{i,t} > 0$ is at least 30 across all periods t in the set T.

2. **Date Gap-Filling:**

After the initial filtering, for products with irregular time intervals, missing dates are filled to create equidistant time intervals:

$$D'' = \text{FillGaps}(D')$$

here, $\text{FillGaps}(\cdot)$ is a process where for each product i, if a sale occurs at t and $t+15$, dates from $t+1$ to $t+14$ are filled with zeros.

3. **Standardized Format for Analysis**

The resulting dataset D'' is formatted to include:

- **unique_id:** Each distinct identifier for a product.
- **ds:** The date of sale (formatted as yyyy-mm-dd).
- **y:** The filled number of units sold for each *unique id* on each date.

The final dataset D'' is now a panel of univariate time series, each represented by $(i, t, y_{i,t})$ where t is now regularized across all products, ensuring consistent time intervals.

Figure 2 – Stacked Time series

	unique_id	ds	y
0	104166695_131028872	2022-01-04	1.0
1	104166695_131028872	2022-01-05	0.0
2	104166695_131028872	2022-01-06	0.0
3	104166695_131028872	2022-01-07	1.0
4	104166695_131028872	2022-01-08	1.0
...	...	...	...
23454275	98361159_543807681	2024-04-18	1.0
23454276	98361159_543807681	2024-04-19	1.0
23454277	98361159_543807681	2024-04-20	3.0
23454278	98361159_543807681	2024-04-21	4.0
23454279	98361159_543807681	2024-04-22	3.0

23454280 rows × 3 columns

1.5 Filtering SKUs and Data Selection Process

Time series for detailed analysis were selected based on two main criteria: maximum demand and average daily demand. These filters were established to ensure that the time series used in the training set are not only active but also representative of significant behaviors or phenomena.

Let D denote the entire dataset containing multiple time series. We apply two primary filters to identify time series with significant behaviors:

1. **Maximum Demand**:

$$\max_{t \in T} \mathrm{y_t} \geq 6$$

Here T is the index set of the time points in the time series, $\mathrm{y_t}$ represents the demand at time t. The expression $\max_{t \in T} \mathrm{y_t}$ returns the maximum value of demand over all time points in the series. The threshold of 6 ensures that each selected time series contains at least one period of significantly high demand.

2. **Average Daily Demand**:

$$\frac{1}{\mathrm{T}} \sum_{\mathrm{t} \in \mathrm{T}} y_t > 1$$

Where $|T|$ is the total number of time points in the time series. The average daily demand is calculated as the mean value of y_t across all time points in T. Setting a threshold greater than 1 ensures that only series with regular and sustained activity are included.

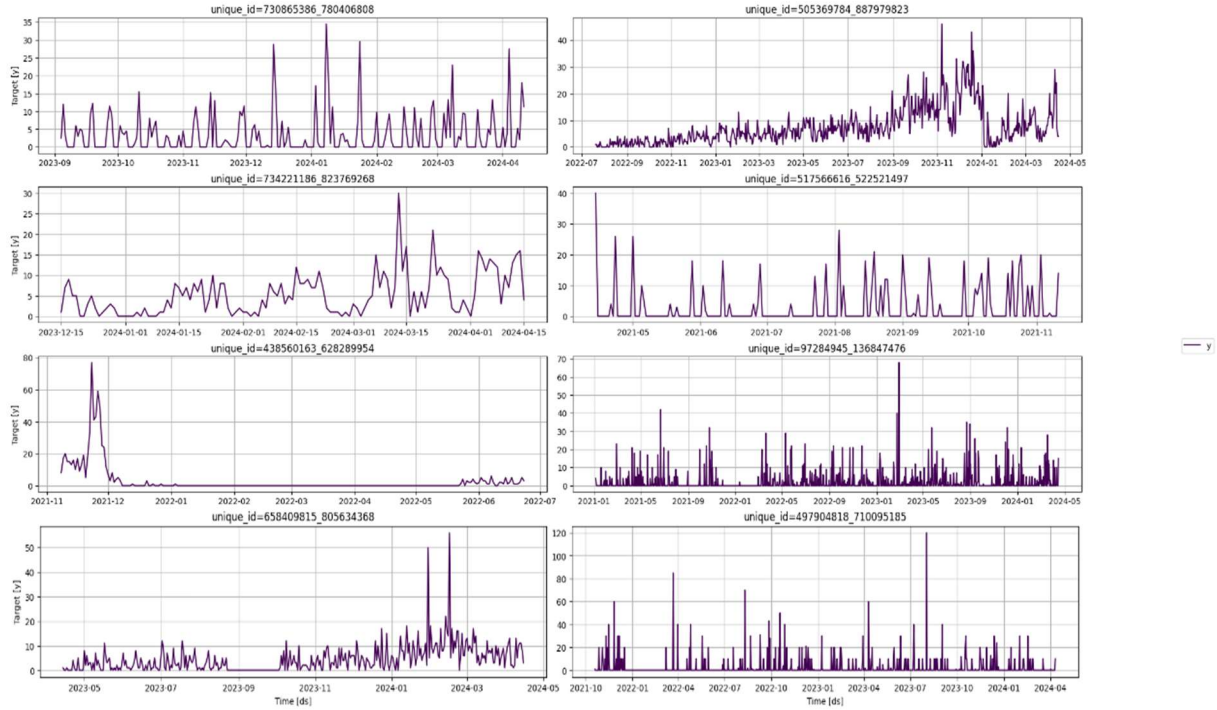
After applying these two criteria, the filtered dataset D_{filtered} can be defined as:

$$D_{\text{filtered}} = \{\text{series } s \in D \mid \max_{t \in T} y_{s,t} \geq 6 \text{ and } \frac{1}{\mathrm{T}} \sum_{t \in T} y_{s,t} > 1\}$$

This filtering process ensures that the selected series are both active and representative of significant behaviors and phenomena, providing data with high intensity and consistency for analysis.

Applying these criteria directs the analysis toward time series that offer a wealth of data in terms of intensity and consistency. This improves the quality of statistical and predictive analyses and ensures that computational and analytical resources are optimized for data that can reveal meaningful patterns or insights.

Figure 3 - Historical series of selected products used for training



About 105,000 unique_ids formed the training set, resulting in a dataset of over 25 million observations. Five thousand unique_ids formed the validation set, with the last 15 observations (the final 15 days) of each unique_id allocated to the test set.

Training and Validation Process:

Let's define the dataset D which is divided into two subsets: D_{train} for training and D_{val} for validation.

1. **Train the algorithm:**

$$Model = Train(D_{train})$$

here, Train(·) is a function that represents the process of training the algorithm using the training set D_{train} .

2. **Using the Trained Model to Forecast:**

$$\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+15} = Forecast(Model, D_{val}, t)$$

Forecast(Model, D_{val}, t) is the function that uses the trained model to analyze historical data of unique IDs in the validation set D_{val} and forecast future values for 15 days. Here, t is the time index up to which historical data is considered.

3. **Evaluate predictions.**

$$Performance = Evaluate(\hat{y}_{t+1}, \hat{y}_{t+2}, \dots, \hat{y}_{t+15}, y_{t+1}, y_{t+2}, \dots, y_{t+15})$$

Evaluate(.) is a function that assesses the accuracy of the prediction $(\hat{y}_{t+1}, \hat{y}_{t+2}, \dots, \hat{y}_{t+15})$ against the actual outcomes $(y_{t+1}, y_{t+2}, \dots, y_{t+15})$ during the test period, which are the actual observed values following the time t.

Since the goal is to use transfer learning, the model's training process was directed toward this purpose. For over a year, careful exploratory analyses and intensive research were conducted to create variables capable of capturing the pattern of frequent zeros in products with intermittent demand.

The complexity of modeling erratic and lumpy time series was known from the start, but exogenous variables were carefully created to capture these patterns adequately.

During the development of this model, the company didn't lack forecasts. Hierarchical modeling alternatives using the fantastic fable library in R were provided, adjusting an autoregressive model for the global series and disaggregating through the historical average proportion method at lower hierarchical levels[1].

Moreover, the modeltime[2] library in R was also utilized, providing excellent forecasts. However, for architectural reasons, a Python solution was necessary, leading to the discovery of the Nixtla[3] ecosystem and the MLForecast library.

1.6 Selection of SKUs Suitable for Transfer Learning

Products currently experiencing sales and meeting minimum variation criteria enter the forecasting process, ensuring that the pre-trained LightGBM can detect patterns. Thus, not all registered SKUs are suitable for forecasting. Filters are applied to the 30 days leading up to the

[1] https://otexts.com/fpp3/hts.html
[2] https://business-science.github.io/modeltime/
[3] https://www.nixtla.io/

prediction date to ensure that each product has a minimum number of days with sales and minimum variation.

This filtering resulted in approximately 70,000 SKUs for the 15-day experimental period. Recursively, forecasts are made for the selected SKUs for the next 15 days.

Even if a product has never been sold on a specific date, like a holiday or Black Friday, it's likely that during training, other SKUs with similar patterns have been learned, allowing the model to provide holiday demand estimates for this new product.

This method ensures that the global machine learning model can generalize to new products well, leveraging patterns learned from more comprehensive SKU histories. We also sought external references to validate our methods. The M5 Forecasting Accuracy competition on Kaggle is a helpful benchmark, where participants faced similar challenges predicting retail product sales for Walmart. Successful use of Gradient Boosting Machines (GBM) in M5 caught our attention toward this algorithm type.

2. Feature Engineering

In global time series forecasting models, incorporating lag variables, moving features, and calendar features are strategically relevant.

Moving averages are calculated through a sliding window to summarize information like means or variances, reflecting local trends or cyclicality within the time series.

Calendar variables represent specific temporal effects, such as weekdays, holidays, or seasons, that consistently influence the behavior of the time series.

2.1 Lags

Incorporating lags as predictors involves using sales values from previous days to predict future sales. This premise assumes that past sales provide valuable insights into trends, patterns, and cycles that will likely recur over time.

Mathematically, this means that each future value y_t of the time series can be modeled as a function of previous values $y_{t-1}, y_{t-2} \ldots, y_{t-n}$, where n is the number of lag periods considered. These lags act as explanatory variables that help the model better understand how sales on a

day are affected by sales from previous days, weeks, or even months, depending on the lag structure. These lags act as explanatory variables that help the model better understand how sales on a day are affected by sales from previous days, weeks, or even months, depending on the lag structure.

Short Lags ($y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}, y_{t-6}$) were selected to capture immediate autocorrelation, **Weekly Lags** (y_{t-7}, y_{t-1}) which are multiples of seven days, as they allow the model to capture weekly patterns, and a **Three-Week Lag** (y_{t-21}), which is important for capturing long-term patterns. Thus, we can represent the forecasts $\hat{y}_t$ as a function of its lagged values:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-7}, y_{t-14}, y_{t-21})$$

Finally, the increased computational complexity associated with larger lags does not necessarily translate to improved forecast accuracy, making lag restriction a balanced decision between capturing sales trends and operational efficiency. Therefore, the careful selection of lags up to 21 days aims to maximize the model's predictive performance while maintaining the quality and completeness of available data.

2.2 Moving Window Features

Moving window features are derived from the response variable y at time t and are incorporated into the dataset as new predictive variables after transformation. This process enriches the model with critical insights into the temporal dynamics of sales.

Features Created from y at time t=1 (Lag 1):

1. **Logarithmic Trend**

   Purpose: Captures long-term trends by identifying exponential growth or decline, particularly valuable for products in the introduction or decline stages of the market.

   Assuming t is the time index ranging from 1 to n, you can define a logarithmic trend variable L_t as follows:

$$L_t = ln(t)$$

Here, *Lt* increases logarithmically as *t* progresses from 1 to *n*, effectively transforming the linear time index into a logarithmic scale. This transformation is particularly useful in time series analysis to model data where growth or decay changes not linearly, but exponentially.

2. **Moving Averages at Different Lags:**

   Purpose: Moving averages are widely used in time series analysis to smooth out short-term fluctuations and highlight longer-term trends or cycles. For different lags, you can define moving averages to focus on specific time frames, such as short-term or medium-term trends.

   - **Short-Term (Lags 2 to 7 days):** Smooths out daily fluctuations and emphasizes weekly patterns, beneficial for products with stable demand. For short-term analysis, you may calculate moving averages using lags from 2 to 7 days. The moving average at time t for a window size n is given by:

$$MA_{t,n} = \frac{1}{n} \sum_{i=t-n+1}^{t} y_i$$

   where $MA_{t,n}$ is the moving average at time t, n is the number of days in the moving window (e.g., 2 to 7 days) and y_i is the value of the time series at time i.

   - **Medium-Term (Lags 14 and 21 days):** Reveals trends over several weeks, crucial for understanding products with erratic or lumpy demand. For medium-term trends, you might use lags like 14 and 21 days. The formula is similar but applied to longer windows:

$$MA_{t,n} = \frac{1}{n} \sum_{i=t-n+1}^{t} y_i$$

   where n takes values 14 and 21.

3. **Moving Minima and Maxima:**

   Purpose: Identifies extreme sales values, crucial for capturing demand volatility. In time series analysis, computing the moving minimum and moving maximum over a specified window can provide insights into the short-term extremes of the data, which

are particularly useful for identifying trends, support and resistance levels, or anomalies.

The moving minimum is the smallest value within a specified window of n observations at each point in time. The mathematical expression for the moving minimum at time t is:

$$min_t = min(y_{t-n+1}, y_{t-n+2}, \dots, y_t)$$

Similarly, the moving maximum is the largest value within the same window. The mathematical expression for the moving maximum at time t is:

$$max_t = max(y_{t-n+1}, y_{t-n+2}, \dots, y_t)$$

4. **Moving Standard Deviations:**

Purpose: In time series analysis, calculating a moving (or rolling) standard deviation helps to understand the variability or volatility of the data over a specified period.

The moving standard deviation for a series at time *t* over a window of size *n* is calculated using the formula:

$$\sigma_t = \sqrt{\frac{1}{n}\sum_{i=t-n+1}^{t}(y_i - \bar{y}_t)^2}$$

Where σ_t is the moving standard deviation at time t, y_i are the values of time series within the window and $\bar{y}_t$ is the moving average over the window, calculate as

$$\bar{y}_t = \frac{1}{n}\sum_{i=t-n+1}^{t} y_i$$

5. **First- and Second-Order Differences:**

Purpose: Removes trends and seasonality, emphasizing changes in sales. First-order differentiation provides information on the magnitude and direction of changes, while second-order differentiation shows how strong (or weak) these changes are.

The first difference of a time series is the series of changes from one period to the next. This can be expressed mathematically as:

$$\Delta y_t = y_t - y_{t-1}$$

where Δy_t is the first difference of the series at time *t*, and y_t is the value of the series at time *t*.

The second difference is the first difference of the first difference, effectively measuring the change in the changes. It is defined as:

$$\Delta^2 y_t = \Delta y_t - \Delta y_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-1}$$

where $\Delta^2 y_t$ is the second difference of the series at time *t*.

6. **Daily Seasonality (Combinations of Sines and Cosines):** The application of sine and cosine functions to model daily seasonality is a powerful technique that enables the forecasting model to capture cyclical patterns that repeat at regular intervals. These functions are particularly valuable for identifying and adapting to fluctuations that occur in daily, weekly, or even monthly cycles. For each day, variables are created using:

$$\sin\left(\frac{2\pi f t}{360}\right)$$
$$\cos\left(\frac{2\pi f t}{360}\right)$$

where f represents the seasonality frequency, with values ranging from 1 to 50 and t is the time index from *1* to n, where n is the length of each time series, allowing the function to capture the seasonality adjusted to the specific length of each time series in the dataset.

Features Created from y at times t ≥ 2 a t ≤ 7 (Lag 2 to Lag 7)

Creating moving window variables starting from lags preceding Lag 1 is very interesting because it allows the machine learning algorithm to capture the movement of patterns during the passage of lags, directly impacting the quality of predictions. Basically, the same variables

created from Lag 1 were used, with modifications to the moving window lengths and a reduction in the frequency f of the sine and cosine combinations.

2.3 Calendar Features

Calendar variables are crucial for capturing specific temporal patterns that influence e-commerce sales. These variables are implemented as binary indicators (0 or 1), where '1' indicates the presence of a specific condition and '0' indicates its absence. Here's a technical description of how each calendar variable is calculated:

1. **Days of the Week:**

   Each day of the week has its own indicator variable:

$$d_{i,w} = \begin{cases} 1 & if\ dayofweek(i) = w \\ 0 & otherwise \end{cases}$$

   where i represents a specific day, and w is the day of the week (e.g., 0 for Monday, 1 for Tuesday, etc.).

2. **Months of the Year:**

   Each month has its own indicator variable $m_{i,k}$:

$$m_{i,k} = \begin{cases} 1 & if\ month(i) = k \\ 0 & otherwise \end{cases}$$

   where k represents the month number (e.g., 1 for January, 2 for February, etc.).

3. **Weekdays:**

   The indicator variable for weekdays, w_i, is defined as:

$$w_i = \begin{cases} 1 & if\ weekday(i) < 5 \\ 0 & otherwise \end{cases}$$

4. **Black Friday Week:**

   For the Black Friday week, the variable b_i is defined as:

$$b_i = \begin{cases} 1 & if\ i\ is\ on\ the\ last\ Friday\ of\ November\ or\ in\ the\ week\ containing\ it \\ 0 & otherwise \end{cases}$$

5. **Seasons:**

Variables for each season are defined as:

$$s_{i,season} = \begin{cases} 1 & if\ i\ is within\ the\ date\ range\ of\ the\ specific\ season \\ 0 & otherwise \end{cases}$$

For example, for summer in the Southern Hemisphere:

$$s_{i,summer} = \begin{cases} 1 & if\ December\ 21\ \leq date(i)\ \leq < March\ 20 \\ 0 & otherwise \end{cases}$$

2.4 Target Transform

The response variable (quantity sold) was not used directly; instead, a lag-7 difference was applied beforehand to prepare the data for analysis. This technique is mathematically expressed as:

$$\Delta y_t = y_t - y_{t-7}$$

This formula adjusts each point in the time series by subtracting the sales value of the same weekday the previous week. This operation is relevant for several mathematical and analytical reasons. The differentiation technique, specifically subtracting y_{t-7} from y_t, is used to mitigate the effects of weekly seasonality and promote stationarity in time series, which are crucial characteristics for effective forecasting models.

2.5 Algorithm Selection

In the search for the most effective model to predict demand across a wide variety of products in a large-scale e-commerce environment, several machine learning algorithms were tested, including LightGBM, XGBoost, Random Forest, Autoregressive Multilayer Perceptron, and K-Nearest Neighbors (KNN). For the specific context of Olist, with approximately 200 derived variables, including 100 columns exclusively for Fourier terms (sine and cosine), it was crucial to select a model that not only efficiently managed a large number of variables but also captured diverse demand patterns (smooth, erratic, lumpy, and intermittent).

2.6 LightGBM as Chosen Model

Among the models evaluated, tree-based models, specifically LightGBM, proved promising due to their ability to handle the complexity and heterogeneity of the data. In addition to the excellent ability to capture different demand patterns, LightGBM is a machine learning framework for building decision tree-based boosting models, particularly efficient for large data volumes. In the case of LightGBM, a common representation is using gradient boosting, where the model is built by iteratively adding trees to minimize the loss function. The basic equation for updating a model in each iteration in gradient boosting, which also applies to LightGBM, can be described as follows:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta \cdot h_t(x)$$

where $\hat{y}^{(t)}$ r epresents the model's prediction at iteration t, $\hat{y}^{(t-1)}$ is the model's prediction at iteration t−1, η is the learning rate, controlling the impact of each new tree added, and $h_t(x)$ is the contribution of the tree added at iteration t.

Enhancing LightGBM with Residual Training

LightGBM enhances the gradient boosting process by training each subsequent tree on the residuals of the previous tree's predictions. This approach is particularly effective in time series analysis because it specifically targets modeling the errors from previous predictions. These residuals often contain unmodeled patterns that are crucial for improving forecasting accuracy. Here's how this process can be mathematically illustrated:

Initially, the prediction from the first tree is given by:

$$\hat{y}^{(1)} = f^{(1)}(x)$$

where $f^{(1)}(x)$ is the prediction function of the first tree. The residuals after the first tree are computed as:

$$r^{(1)} = y - \hat{y}^{(1)}$$

where y is the actual value of the target variable. Each subsequent tree is then trained to predict these residuals, refining the model's accuracy by focusing on what was previously missed. The prediction of the second tree aimed at these residuals would be:

$$\hat{y}^{(2)} = \hat{y}^{(1)} + \eta \cdot h^{(2)}(x)$$

where $h^{(2)}(x)$ represents the contribution of the second tree, and η is the learning rate. The residuals are then updated as:

$$r^{(2)} = y - \hat{y}^{(2)}$$

This process repeats iteratively, with each tree learning to correct the errors made by the ensemble of all previous trees:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta \cdot h^{(t)}(x)$$

$$r^{(t)} = y - \hat{y}^{(t)}$$

where *t* indicates the iteration number. Each tree $h^{(t)}(x)$ targets the residuals $r^{(r-1)}$ from the previous step, refining the model's predictions incrementally. This targeted refinement on residuals enables LightGBM to adaptively capture complex temporal dependencies and autocorrelations characteristic of time series data.

Effectiveness in Time Series

In time series forecasting, where past information and patterns significantly influence future outcomes, focusing on residuals allows LightGBM to capture dynamics often missed by more straightforward forecasting models. This iterative correction of residuals ensures continuous improvement in forecast accuracy, making LightGBM particularly powerful for datasets with complex temporal structures.

By directly targeting these residuals, LightGBM ensures that each iteration progressively reduces the forecasting error, adapting to the intricacies and nuances of time series data effectively.

LightGBM excels in handling large datasets due to innovative techniques like Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS enhances training efficiency by focusing on observations with larger errors and randomly sampling observations with smaller errors, thus accelerating convergence and reducing overfitting. EFB tackles high dimensionality by grouping mutually exclusive features, reducing complexity without losing information and improving processing eficiency.

Gradient-Based One-Side Sampling (GOSS)

GOSS is a method that allows LightGBM to focus more on observations that are harder to predict, which generally have larger gradients. In LightGBM, observations are ranked based on the absolute values of their gradients, and a subset of the data is selected for training based on these values.

GOSS Method:

1. **Gradient Sorting and Sampling:**
   - Let $\{(x_i, y_i)\}_{i=1}^{n}$ be the training dataset where *xi* and *yi* are the features and labels respectively, and *n* is the total number of observations.
   - Compute gradients $g_i = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$ for each observation *i*, where *L* is the loss function and $f(x_i)$ is the model's prediction.
   - Sort observations based on the absolute values of gradients $|g_i|$.

2. **Select a subset for training**

   - Select the top $a\%$ of the observations with the largest gradients to keep in the training subset.
   - Randomly select $b\%$ of the remaining observations with smaller gradients.
   - Form the training subset $\{(x_i, y_i, g_j)\}_{j=1}^{m}$ using both selected groups, Where m<n.

3. **Weight Adjustment:**
   - To compensate for the data reduction from the original dataset, the observations with smaller gradients that are randomly included are given a larger weight in the training process:

$$g'_j = g_j * \left(1 + \frac{1 - a\% - b\%}{b\%}\right)$$

where g'_j adjusted gradient for observations with a smaller original gradient, g_j is the original gradient of the observation before adjustment, $a\%$ is the Percentage of observations with the largest gradients that are directly selected and $b\%$ is the percentage of observations with smaller gradients that are randomly selected.

Exclusive Feature Bundling (EFB) — method description:

EFB reduces the number of features in the dataset by bundling features that are mutually exclusive, meaning they rarely take non-zero values at the same time.

1. **Feature Grouping:**
   - Let $\{f_k\}_{k=1}^{d}$ be the set of all features in the dataset, where d is the total number of features.
   - Identify groups of features $\{f_l\}_{i=1}^{p}$ such that each group contains features that are mostly exclusive to each other.
2. **Bundling:**
   - For each group F_l, create a bundled feature F_l^b by combining the features in f_l.
   - The exact method of combining features can depend on specific criteria, such as overlapping non-zero values.

3. Hyperparameter Tuning and Evaluation Metrics

During model development for Olist, LightGBM's efficiency made hyperparameter tuning less intensive than other models, eliminating the need for GPUs. This is attributed to the careful design of predictive features and a meticulous selection of products that formed the dataset. Fine-tuning mainly focused on minimizing global RMSE, a metric that assesses model performance across varying magnitudes of demand.

3.1 Evaluation Metrics

Root Mean Squared Error (RMSE) - Global:

The global RMSE was chosen as the primary metric due to its efficacy in quantifying the average forecast error. By calculating RMSE globally across all predictions and actual values, the model's ability to handle varying sales magnitudes, from low to high, can be assessed. This approach ensures robustness and good performance regardless of specific SKU fluctuations. RMSE is mathematically expressed as:

$$\text{RMSE} = \sqrt{\frac{1}{n}||y - \hat{y}||_2^2}$$

where

- y is the vector of actual values.
- $\hat{y}$ is the vector of predicted values.
- n is the total number of observations.
- $||.||_2^2$ denotes the L2 norm (or Euclidean norm) of the vector.

Coefficient of Determination (R^2) - Global:

Besides RMSE, the R^2 coefficient was used for evaluating forecast quality. Globally, R^2 is excellent for understanding how well predictions fit actual data. A high R^2 indicates that the predicted vs. actual graph approximates a straight line, signaling high-quality forecasts. Although less informative for individual time series due to sensitivity to data mean and observation count variations, globally, it effectively reveals the model's consistent performance across different products. The matrix expression for R^2 is:

$$R^2 = 1 - \frac{||y - \hat{y}||_2^2}{||y - \bar{y}||_2^2}$$

where

- y is the vector of actual values.
- $\hat{y}$ is the vector of predicted values.
- n is the total number of observations.
- $||.||_2^2$ denotes the L2 norm (or Euclidean norm) of the vector.

Weighted Mean Absolute Percentage Error (WMAPE) - Local:

After each hyperparameter tuning step, WMAPE was used for evaluating individual product forecasts. It's valuable in e-commerce scenarios since it handles zero sales due to intermittent

product demand. This metric provides a percentage error measure easily interpretable by business units, highlighting error relative to actual sales and allowing targeted forecast adjustments. WMAPE is expressed as:

$$WMAPE = \frac{\sum|y - \hat{y}|}{\sum y}$$

where

- y is the vector of actual values.
- $\hat{y}$ is the vector of predicted values.
- $|\cdot|$ denotes the absolute value of each element in the vector.

Here's a pseudo-algorithm to illustrate the training process up to the point of calculating global metrics:

Pseudo-algorithm to illustrate the training process up to the point of calculating global metrics

1. **Initialization:**
   - Let M represent the forecasting model, parameterized by θ.
   - Initialize parameters θ for model M.
2. **Dataset Preparation:**
   - Split the dataset into training (D_{train}) and validation (D_{val}) sets.
   - Reserve the last 15 days of D_{val} as the test set (D_{test}).
3. **Model Training:**
   - Train the model M using D_{train}:

$$\theta^* = \arg\min_{\theta} \sum_{(x_i, y_i) \in D_{\text{train}}} L(M(x_i, \theta), y_i)$$

where L is the loss function, x_i are the input features, and y_i are the actual sales values.

4. **Forecasting and Evaluation:**
   - Train the model M using D_{train}:

   For each product in D_{val}, apply the trained model M to predict future sales for 15 days:

$$\hat{y}_{i,t} = M(x_{i,t}, \theta^*)$$

where $\hat{y}_{i,t}$ is the predicted sales for product i at time t.

5. **Comparison and Aggregation:**
   - Compare the forecasted values $\hat{y}_{i,t}$ with the actual values $y_{i,t}$ in D_{test}.
   - Stack the forecasts and actual values from D_{test} across all products.
4. **Calculation of Global Metrics:**
   - Calculate Root Mean Squared Error (RMSE) and Coefficient of Determination (R^2) globally across all predictions

To train de model, an instance with 96 CPUs and 384 GiB was used on Amazon SageMaker. It's worth noting that the process involves three datasets: training, validation, and testing. RMSE and R^2 are computed on the test data and compared with predictions.

3.2 Evaluated Hiperparameters

When developing global models for demand forecasting, selecting and tuning hyperparameters are crucial to ensure the model is accurate and efficient. Below, I detail the technical properties of the hyperparameters adjusted during tuning, focusing on how each one influences the model's performance in the context of large-scale sales forecasting, such as in the case of Olist.

Table 1 - LightGBM Hyperparameters Evaluated

Hyperparameters	Description	Technical Properties
Number of Estimators (n_estimators)	Represents the number of sequential trees to be built in the model.	Balances complexity with the risk of overfitting; increases stability but raises computational cost.
Learning Rate (learning_rate)	Controls each tree's contribution to the final model, affecting the model's learning speed.	Lower rates improve convergence but may require more trees, extending training.
Maximum Depth (max_depth)	Limits the number of levels in decision trees.	Limits overfitting by controlling depth; relevant for effective generalization.
Number of Leaves (num_leaves)	Defines the maximum number of leaves (terminal nodes) each tree can have.	More leaves increase flexibility and the risk of overfitting; requires a balance for model complexity.
Minimum Child Weight (min_child_weight)	Defines the minimum sum of weights of all observations needed in a child (split node).	Prevents insignificant splits by controlling overfitting and adjusting tree complexity.
Regularization (reg_alpha, reg_lambda)	L1 (reg_alpha) and L2 (reg_lambda) regularization parameters that help control	Smooths the objective function and stabilizes learning; adjustments needed to maintain effectiveness.

	overfitting by penalizing more complex models.	
Subsample	Defines the fraction of samples to be used to train each individual tree in the ensemble.	Reduces overfitting by varying data subsets; increases the model's diversity and robustness.
Colsample_bytree	Specifies the fraction of features to be used to train each tree.	Limits features per tree to prevent overfitting and improve performance in high-dimensional sets; promotes diversity among trees.

3.3 Confidence Interval Construction via Bootstrap

To build robust confidence intervals for demand forecasts at Olist, a block bootstrap method was adapted.

1. Block Size and Resampling Number Definition

Let X_t represent a time series that reflects the daily sales of a product, with t=1,2,...,n, where n is the total number of observed days. The series is divided into blocks B_i, each containing 7 consecutive days to preserve weekly sales dependencies.

The number of blocks k that can be created is calculated as:

$$k = \frac{\mathrm{n}}{7}$$

where n is the observation number in each time series.

Each block B_i is then defined as:

$$B_i = \left(X_{7(i-1)+1}, X_{7(i-1)+2}, \dots, X_{7i}\right), \quad \text{for } i = 1,2, \dots, k$$

2. Block Resampling

During the bootstrap process, blocks B_i are randomly selected with replacement to simulate different temporal sequences. This step helps capture variability in sales while maintaining the internal structure of weekly dependencies:

$$B_i^* \text{ is randomly selected from } \{B_1, B_2, \dots, B_k\}$$

3. Time Series Reconstruction

The blocks selected in the previous step are concatenated to form a new bootstrapped time series X^*:

$$X^* = B_1^* \oplus B_2^* \oplus \ldots \oplus B_k^*$$

where $\oplus$ denotes the concatenation of blocks, maintaining the temporal sequence of the data.

4. Series Identifier Adjustment

Each bootstrapped time series is given a unique identifier. This step is essential to keep track of different bootstrap iterations, allowing accurate and independent analysis of each re-sampled series.

5. Confidence Interval Calculation

Finally, after generating the bootstrapped series, the forecasting model is applied to each one to produce future forecasts. Confidence intervals are calculated using the percentiles of the bootstrapped forecasts, usually with $\mathrm{CI}_{\alpha/2}$ and $\mathrm{CI}_{1-\alpha/2}$for the lower and upper bounds, respectively:

$$\mathrm{CI}_{\alpha/2}, \mathrm{CI}_{1-\alpha/2}$$

we use $\alpha = 5$.

This methodology provides a robust approach to understanding the uncertainty in predictions by considering the variability and temporal dependence inherent in the sales series.

3.4 Transfer learning

Transfer learning is an advanced technique where a model developed for one specific task is adapted to a related new task. In this specific context, the LightGBM model is pre-trained on a vast dataset to learn patterns across numerous products, giving it a high capacity for generalization. The central idea is to apply this model directly to predict sales of new products, even if these products were not part of the original training data.

Instead of re-training or fine-tuning the model specifically for the new products, the available historical data for each new product is simply presented to the model to predict future days'

sales. Thus, this pre-trained LightGBM model utilizes the patterns learned from the original training dataset to infer sales accurately for new products.

1. **Pre-Trained Model:**
   The model f is trained on domain A using dataset D_A to learn the parameters θ:

$$f_\theta = \arg\min_\theta \sum_{(x_i,y_i)\in D_A} L(f(x_i,\theta),y_i)$$

where x_i represents features of the known products, y_i are their corresponding sales, and L is the loss function.

2. **Prediction for New Products:**
   Using the model f with already trained parameters θ, predictions are made for new products based on the available historical data x_j' (where *j* indexes new products in domain *B*):

$$\widehat{y_j'} = f(x_j', \theta)$$

here, x_j' represents the observed features of the new products, and $\widehat{y_j}$ are the predicted sales.

4. Results

This chapter examines the results of applying advanced machine learning techniques for demand forecasting on the Olist e-commerce platform. Through rigorous feature engineering and hyperparameter tuning, we optimized the LightGBM model to achieve the best predictive capacity possible.

The subsequent figure illustrates the relationship between hyperparameter settings and their respective RMSE values. Each subplot represents the RMSE variation concerning a specific hyperparameter, providing a detailed perspective on how fine-tuning can impact forecast accuracy by evaluating 100 different parameter combinations.

Figure 4 – Hyperparameters Tuning vs Global RMSE

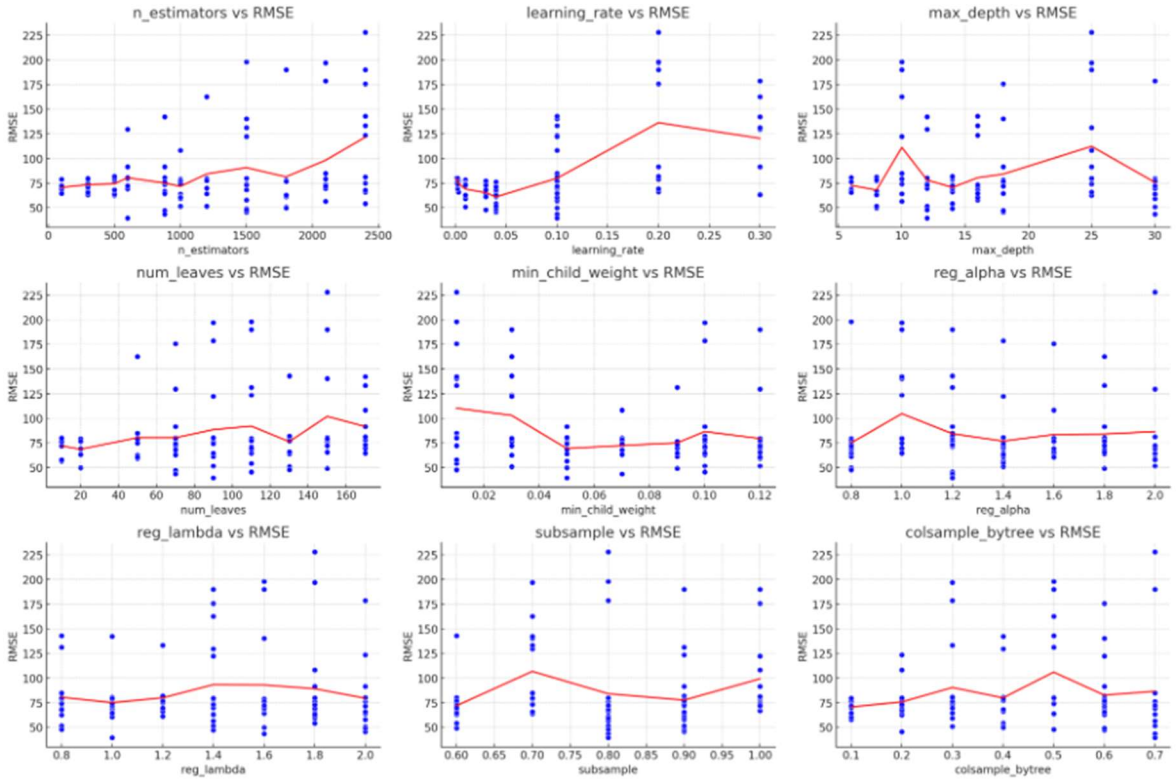


Table 2 -Best Hyperparameters

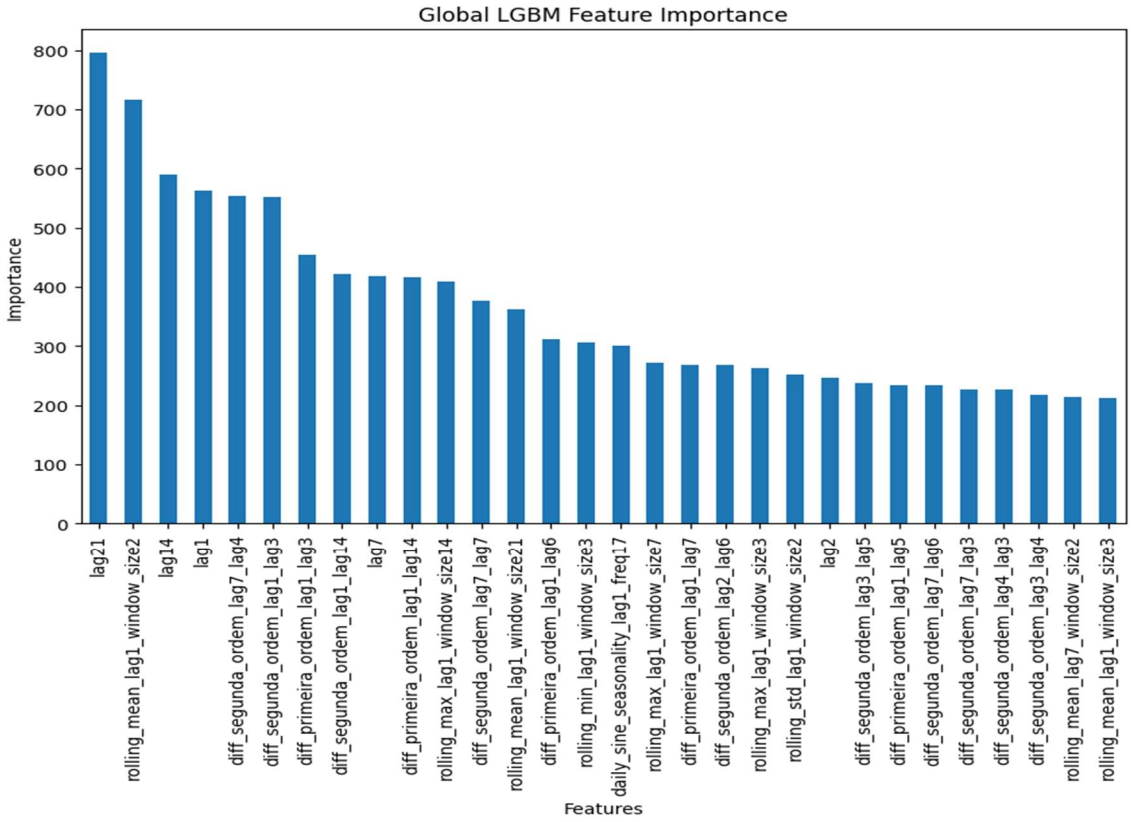
Hyperparameters	Value
n_estimators	600
learning_rate	0.1
max_depth	12
num_leaves	90
min_child_weight	0.05
reg_alpha	1.2
reg_lambda	1.0
subsample	0.8
colsample_bytree	0.7

Resulting in an RMSE of 39.49, indicating a significantly reduced error estimate, and an R^2 of 78.77%, reflecting a strong correlation between predicted and actual values.

4.1 Feature Importance

The following analysis provides variable importance, revealing the weight of each feature in LightGBM model predictions. This analysis helps identify the most influential variables and clarifies how different data contribute to understanding sales dynamics at Olist.

Figure 5 – LightGBM Feature Importance Plot



When evaluating the importance of variables in the global LightGBM model for predicting the sales of various SKUs, we identified a predominance of first and second-order differentiation variables. Differentiation is a powerful technique in time series modeling, used to transform non-stationary data into stationary form, revealing hidden trends and cyclic patterns crucial for forecasting.

First-order differentiation, represented in the model by variables derived from different lags (such as 'lag1', 'lag14', 'lag21'), calculates the difference in sales between two consecutive points in time. This transformation highlights the immediate variation in sales, capturing daily and weekly volatility and allowing the model to detect inertia in the time series—a crucial indicator of short-term trends.

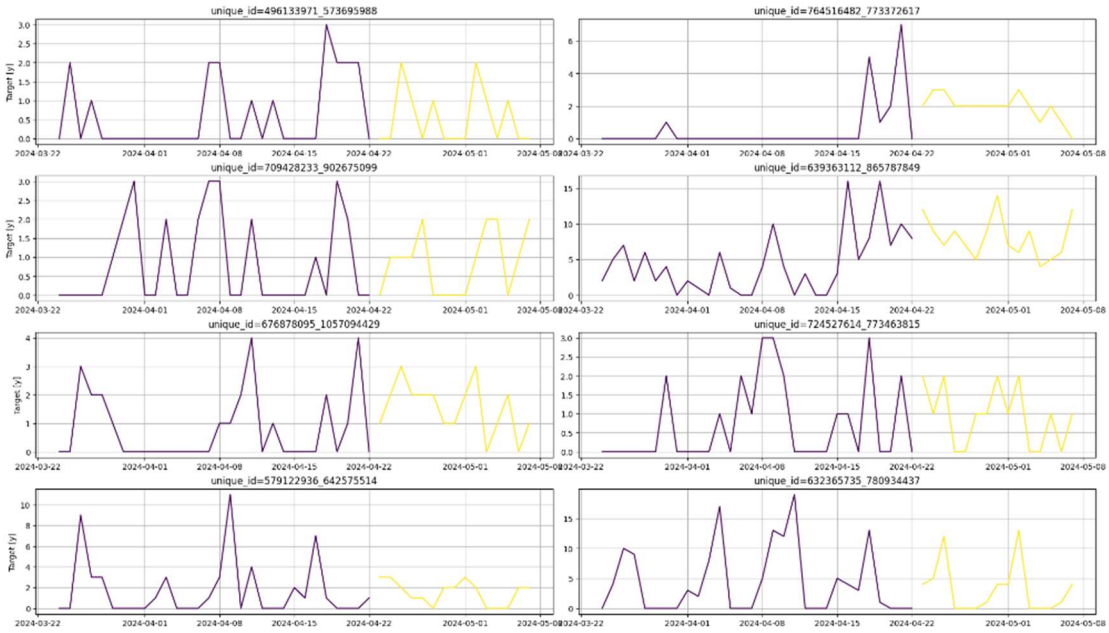
The significant presence of second-order differentiation variables suggests that the model is capturing not only changes in sales but also changes in the changes themselves—effectively, the acceleration or deceleration in sales.

This is especially evident with the second-order variable constructed from 'lag7' with a 4-day window. The captured acceleration reflects oscillations in sales that are not apparent when observing only daily sales or first-order differences. Detecting this acceleration is crucial for identifying turning points in buying behavior, which may precede significant events like promotions or seasonal changes.

Applying these differentiation techniques across multiple time windows allows the LightGBM model to adapt to sales patterns with different frequencies and magnitudes. Smaller windows focus on short-term effects and immediate fluctuations, while larger windows help capture long-term trends and broader cycles.

Additionally, integrating these differentiation variables with different lags allows for more robust modeling, as different products can have distinct sales cycles. Therefore, a lag that captures the sales behavior of a specific SKU may not be the same for another, and differentiation helps to adjust the model for this heterogeneity.

Figure 6 – Fifteen-Day Forecasts for Randomly Selected Products

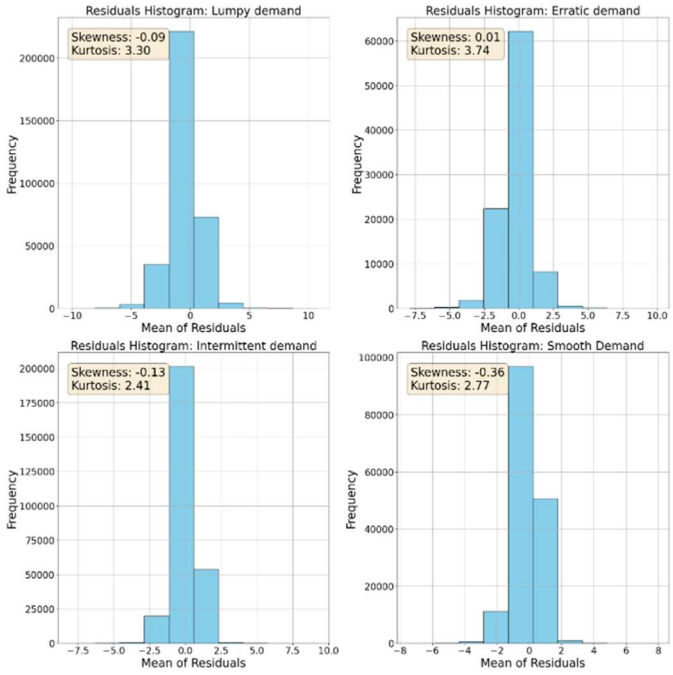


4.2 Residual Analysis

In the subsequent figure, we present histograms of the residuals associated with four different demand patterns. Each histogram provides a concrete visualization of the distribution of

forecast errors, allowing us to assess how well the model is fitting the data. Ideally, we aim for a residual distribution as close as possible to a normal distribution, indicating that the model is capturing all relevant information and leaving only the random noise inherent to the sales process. The residual distribution in a forecast model tells us a lot about possible biases toward underestimation or overestimation. The ideal scenario is for the residuals to be normally distributed around zero.

Figure 7 - Residuals Histogram by Demand Pattern



Lumpy demand, with its tendency toward quasi-random patterns and unpredictable peaks, serves as substantial proof of the model's robustness. The fact that the residual skewness is maintained so close to zero is an achievement, reflecting a remarkable balance in forecasts that can often be inherently fluctuating.

Erratic demand, often marked by abrupt and sudden variations, presents a distinct challenge for any predictive algorithm. Despite this, the model manages to maintain an extraordinarily centered error distribution, as evidenced by the almost negligible skewness. This is a clear indication of the model's ability to handle volatility without introducing systematic bias, which is especially impressive given the inherent uncertainty of this type of demand pattern.

Intermittent demand, which includes the challenge of predicting consecutive days of zero sales, shows that even in this difficulty, the model can provide reasonably confident results. Accurate forecasts in this scenario highlight the analytical precision of the model. The modest tilt toward overestimation indicates that the model remains optimistic about the possibility of

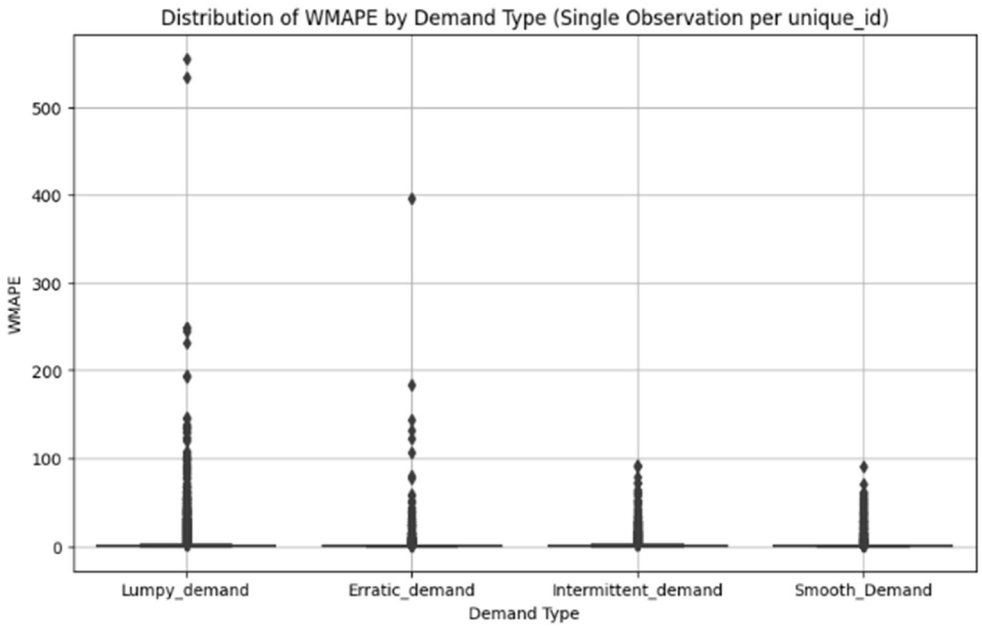
sales, given that this demand pattern is marked by long days without sales, leading to the model predicting some sales when there were none. However, this may be a preferable quality in many business and planning contexts.

Smooth demand, which could be considered the most predictable profile, reveals through the residuals that the model is attentive to this category's nuances, adapting and responding accurately to its subtle variations. The kurtosis indicates that the model is prepared to capture both ordinary days and extraordinary events, providing a reliable basis for consistent forecasts. In summary, the results achieved reinforce the model's adaptability and accuracy in the face of the multifaceted complexity of demand behavior.

4.3 Evaluation of Point Forecasts for Each Unique_id

The distribution of WMAPE by demand profile highlights an encouraging outlook on the model's predictive capabilities, especially considering the colossal uncertainty that characterizes large-scale retail.

Figure 8 – WMAPE Distribution for Each Unique_id by Demand Pattern Type



For erratic demand, which represents 10.19% of the total SKUs, we see that the median WMAPE is 36%. This indicates that despite abrupt fluctuations in sales, half of the time, the model forecasts with an error of up to 36%. This is a positive outcome, suggesting that even in times of high volatility, the model performs very well.

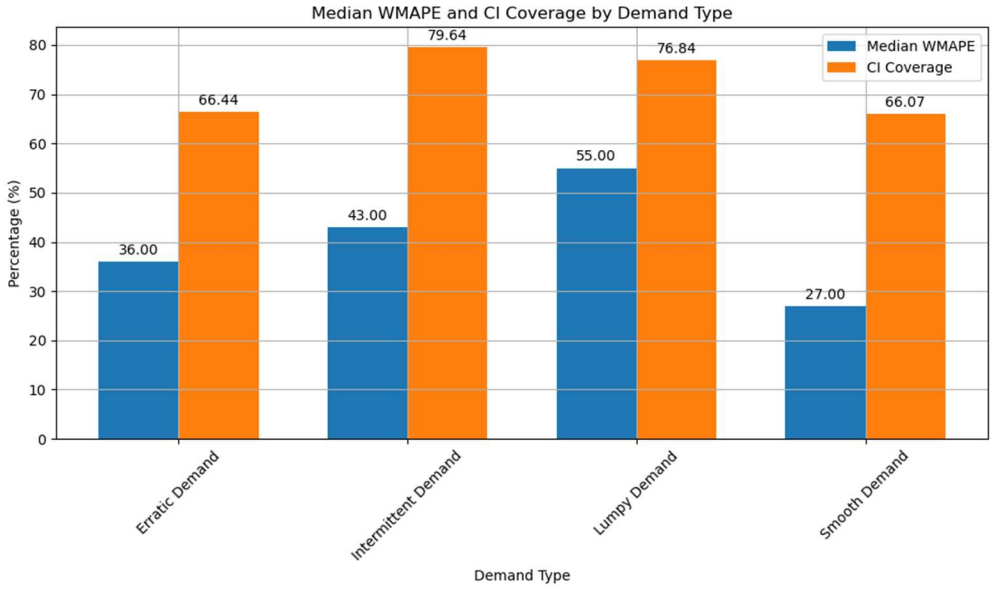
In products with intermittent demand, accounting for 32.42% of SKUs, the median WMAPE of 43% reflects the model's robust adaptability. Considering that many of these products might not register sales on several days, the result shows that the model can capture and accurately predict demand intermittence, an essential aspect in an environment with a broad range of products exhibiting this demand pattern.

For products with lumpy demand, making up 40.40% of the SKUs, the median WMAPE is 55%. This is evidence that the model is handling what could be the most complex demand pattern due to its unpredictable nature, both in timing and sales volume.

Finally, products with smooth demand, representing 16.96% of SKUs, had a median WMAPE of 27%. This result suggests that the model performs exceptionally well in this profile, demonstrating its ability to follow a regular sales pattern even in an environment where demand can fluctuate.

When considering confidence interval (CI) coverage, we have even more reasons to be optimistic. The CI coverage for erratic demand is 66.44%, for intermittent demand is 79.64%, for lumpy demand is 76.84%, and for smooth demand is 66.07%. These figures tell an interesting story: in most cases, actual demand falls within the estimated intervals.

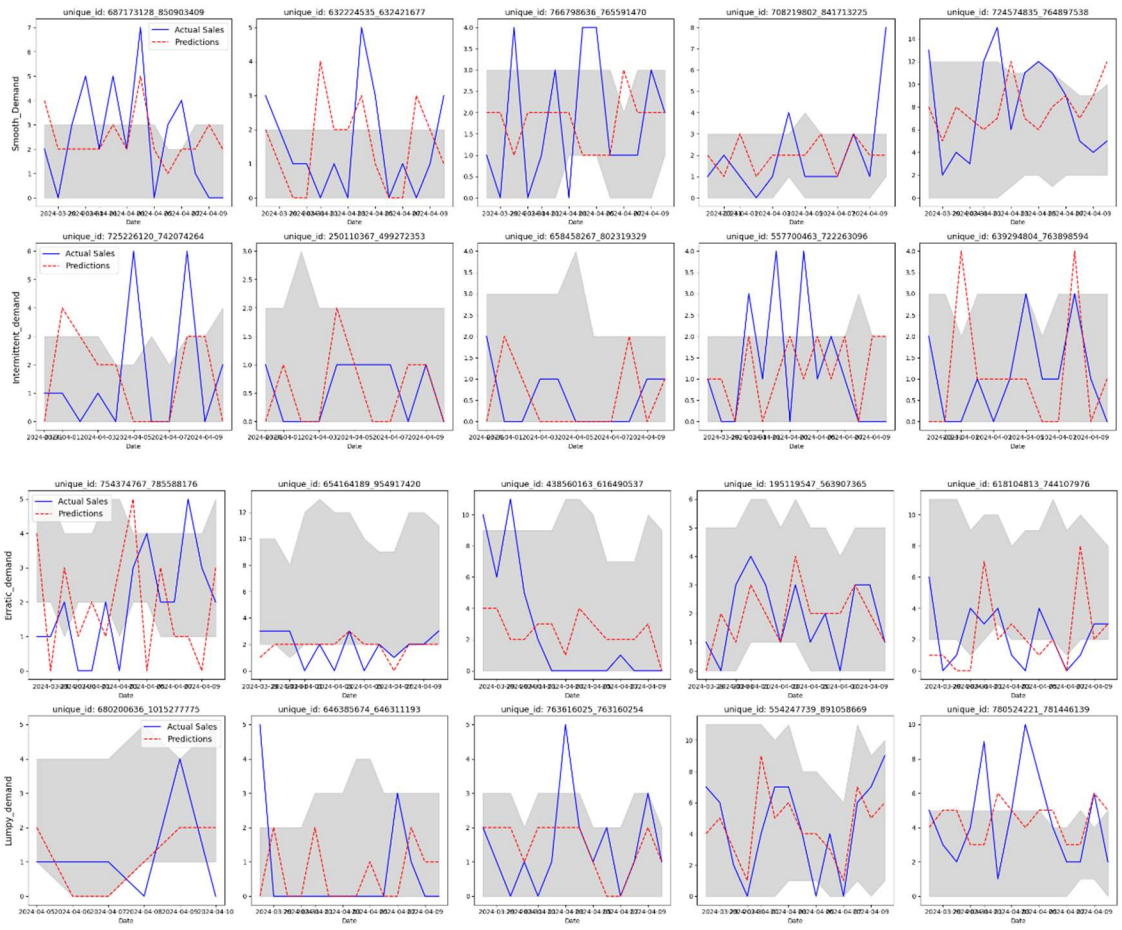
Figure 9 - Results by Demand Pattern



For Olist, this means that although predicting abrupt peaks and valleys is challenging, the model is prepared to absorb and handle these variations without compromising overall confidence in its forecasts. These local WMAPEs were obtained by maximizing global RMSE in the hyperparameter tuning process.

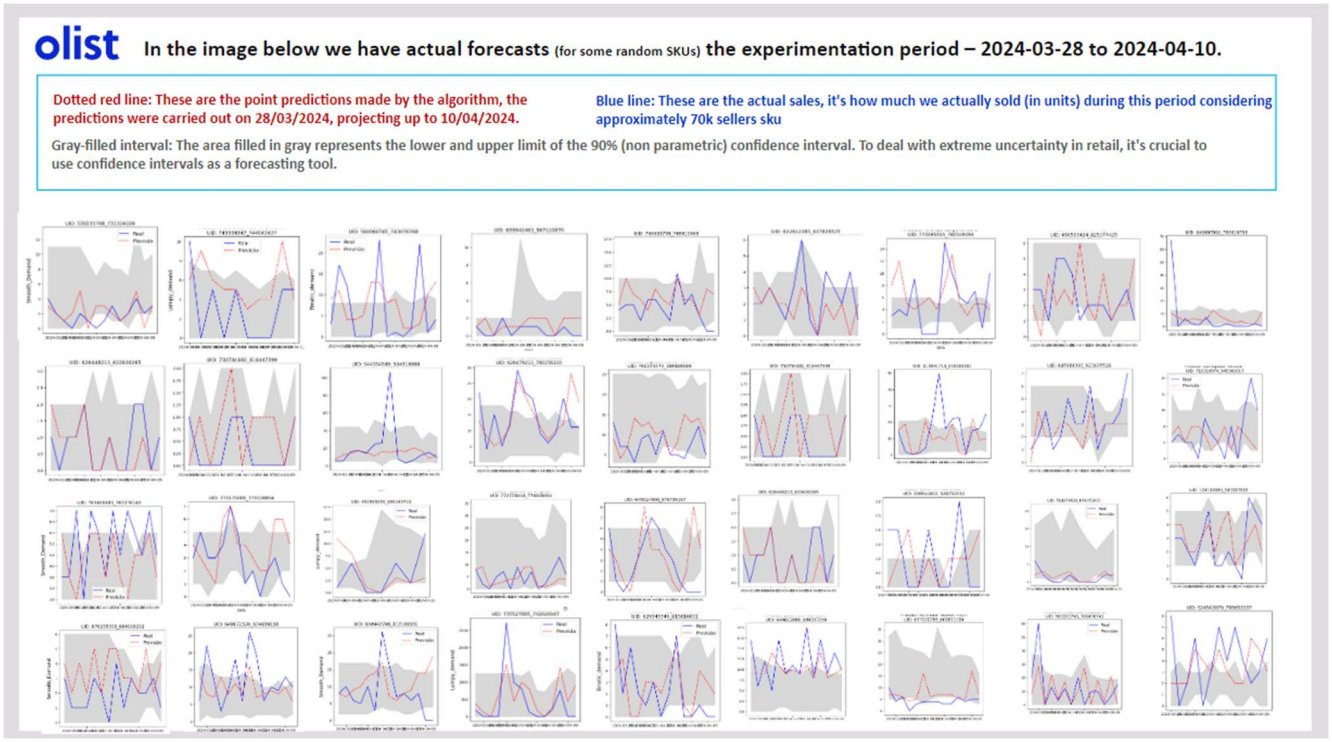
With this information, we can say that Olist has a predictive model that not only meets the challenge of an uncertain and varied retail environment but also excels in its ability to produce reliable and actionable forecasts, crucial for strategic business decisions, inventory management, and operational planning. It is a predictive approach that adapts and responds to the complexities of e-commerce, proving to be a valuable tool for maximizing operational efficiency and customer satisfaction. We can get a view of 15-day forecasts compared to actual sales at the SKU level.

Figure 10 - Actual Sales vs. Forecasts (for some randomly selected SKUs) During the Experimentation Period — 03/28/2024 to 04/10/2024



We can also visualize the summarized results presented at the executive model presentation meeting at the company.

Figure 11 - Executive Presentation Slide



In figure 12, we have a summary of forecasts evaluated by the median WMAPE, broken down by demand profile, and the confidence interval coverage, also by demand type. After all, more complex profiles like intermittent, lumpy, and lumpy have significant uncertainties, so confidence intervals assist decision-making. Additionally, the confidence interval coverage was presented, indicating the percentage of times that actual demand was within the generated confidence interval.

Figure 12 - Executive Presentation Slide – Metric Evaluation

olist **General evaluation of the predictions during the experimentation period**

Evaluation of point predictions

Demand Pattern	Number of SKUs	Number of SKUs *(relative)*	Median WMAPE	Pratical Interpretation
Erratic	7005	10.19%	36%	In half of the predictions the error was up to 36%. This tells us that despite the abrupt variations that can arise in sales from one day to the next, the predictions do well on other days.
Intermittent	22276	32.42%	43%	In half of the predictions the error was up to 43%. Given the intermittent nature with many days without sales, this means that more than half of the time the algorithm managed to predict that there would be no sale when there actually wasn't.
Lumpy	27758	40.40%	55%	In half of the predictions the error was up to 55%.
Smooth	11654	16.96%	27%	In half of the forecasts, the error was up to 27%. This is a positive result because the Smooth demand has a more regular sales pattern over time and in the quantity sold, a high error here would be quite negative.

Evaluation of confidence interval coverage

Demand Pattern	Number of SKUs	Number of SKUs *(relative)*	IC Coverage	Pratical Interpretation
Erratic	7005	10.19%	66.44%	66.44% of the actual demand was contained within the estimated confidence intervals. A positive result for Olist, as according to theory this pattern is not even predictable due to the emergence of abrupt peaks or troughs from one day to the next.
Intermittent	22276	32.42%	79.64%	79.64% of the actual demand was contained within the estimated confidence intervals, that is, in almost 80% of the time the actual demand was within the confidence interval. This is quite positive, as due to the high number of zeros (days without sales) it is not trivial to predict this demand pattern.
Lumpy	27758	40.40%	76.84%	76.84% of the actual demand was contained within the estimated confidence intervals, that is, in almost 77% of the time the actual demand was within the confidence interval. This is something surprisingly positive, as the lumpy pattern is extremely irregular over time (many days without sales) and irregular in demand (when there is a sale, there is a lot of variation in quantity).
Smooth	11654	16.96%	66.07%	66.07% of the actual demand was contained within the estimated confidence intervals. Despite a moderate coverage, the point predictions perform very well here (as per the table above).

5. Conclusions

This study made a significant contribution to the field of demand forecasting in e-commerce by employing an innovative machine learning approach using LightGBM, complemented by transfer learning techniques and non-parametric bootstrapping. The research addressed the unique challenges presented by the diversity of SKUs managed by the Olist platform, emphasizing the need for a robust and adaptable demand forecasting system. The results highlight the model's effectiveness in handling these challenges, demonstrating reliability and adaptability across various sales scenarios and emphasizing the strategic use of bootstrapping to manage sales uncertainty.

The model's success was evidenced not only by optimizing the global RMSE but also by the individual performance of the SKUs, as reflected in the WMAPE medians, which were substantial even for SKUs with erratic and intermittent demand. The model proved exceptionally efficient for smooth demand, representing a significant fraction of the SKUs. This efficiency is a clear indicator that the adopted approach aligns with large-scale e-commerce operations, providing a robust model that can be applied to different demand patterns.

The techniques used in this study, particularly transfer learning, show great promise for implementation in other e-commerce sales forecasting scenarios for different products or services. In addressing heterogeneity, the approach used here establishes a methodological benchmark that can be explored in future research and practical applications. The model's adaptability is crucial in a constantly changing market where the ability to respond quickly to consumer trends is essential for business success.
In summary, the study validated the application of a global machine learning model in e-commerce demand forecasting, providing a path to consistent and reliable forecasts. The use of non-parametric bootstrapping and transfer learning via LightGBM emerged as a methodological advance that ensures sufficiently accurate and adaptable forecasts, essential for strategic operations and competitiveness in large-scale e-commerce.

6. References

Yi, S. (2023). Walmart Sales Prediction Based on Machine Learning. *Highlights in Science, Engineering and Technology*.

LIGHTGBM**.** Oficial Documentation: https://lightgbm.readthedocs.io/

MONTERO-MANSO, Pablo; HYNDMAN, Rob J. Principles and Algorithms for Forecasting Groups of Time Series: Locality and Globality. Sydney: University of Sydney; Monash University, 30 mar. 2021.

Raizada, S., & Saini, J. R. (2021). Comparative Analysis of Supervised Machine Learning Techniques for Sales Forecasting. *International Journal of Advanced Computer Science and Applications*.

Niu, Y. (2020). Walmart Sales Forecasting using XGBoost algorithm and Feature engineering. *2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*.

Ding, J., Chen, Z., Xiaolong, L., & Lai, B. (2020). Sales Forecasting Based on CatBoost. *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*.

Islam, D. L., Farooqui, M. F., Khan, A., Wasi, M., & Shaikh, T. (2023). Walmart Sales Analysis and Prediction. *International Journal of Advanced Research in Science, Communication and Technology*.

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

Singh, B., Kumar, P., Sharma, N., & Sharma, K. P. (2020). Sales Forecast for Amazon Sales with Time Series Modeling. *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*.

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q.-H., & Seaman, B. (2019). Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology. *ArXiv*.

Zhou, T. (2023). Improved Sales Forecasting using Trend and Seasonality Decomposition with LightGBM. *2023 6th International Conference on Artificial Intelligence and Big Data (ICAIBD)*.

Pustokhina, I., & Pustokhin, D. A. (2023). A Comparative Analysis of Traditional Forecasting Methods and Machine Learning Techniques for Sales Prediction in E-commerce. *American Journal of Business and Operations Research*.