CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Enhancing Supply Chain Forecasting with Machine Learning

Unconstrained Demand Prediction Using Time Series and Gradient Boosting Methods

Master's thesis in Computer science and engineering

Andres Felipe Otero Salamanca
Elínborg Ása Ásbergsdóttir

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Enhancing Supply Chain Forecasting with Machine Learning

Unconstrained Demand Prediction Using Time Series and Gradient Boosting Methods

Andres Felipe Otero Salamanca
Elínborg Ása Ásbergsdóttir

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Enhancing Supply Chain Forecasting with Machine Learning
Unconstrained Demand Prediction Using Time Series and Gradient Boosting Methods

Andres Felipe Otero Salamanca
Elínborg Ása Ásbergsdóttir

Supervisor: Stefano Sarao Mannelli, Department of Computer Science and Engineering
Examiner: Ashkan Panahi, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Enhancing Supply Chain Forecasting with Machine Learning
Unconstrained Demand Prediction Using Time Series and Gradient Boosting Methods

Andres Felipe Otero Salamanca
Elínborg Ása Ásbergsdóttir
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Accurately forecasting unconstrained demand is essential for effective supply chain planning and inventory management. This thesis examines the performance of both traditional statistical methods and modern machine learning models in predicting unconstrained demand, with a particular focus on comparing global and local modeling strategies across multiple products. Using real-world sales data, models such as ARIMA, XGBoost, and LightGBM were evaluated based on their ability to produce accurate forecasts.

The comparative analysis of forecasting models across multiple products reveals that model performance varies significantly depending on product characteristics and the chosen modeling approach. Overall, modern machine learning approaches, particularly LightGBM and XGBoost, consistently outperformed the traditional ARIMA benchmark in terms of predictive accuracy. However, their effectiveness varied by modeling approach and product type.

The results indicate that local models, especially those based on LightGBM, tend to perform better for products with lower demand, as they are more capable of capturing product-specific patterns and fluctuations. In contrast, global models show stronger performance for high-demand products, where shared patterns across products can be more effectively leveraged. Although global models may not always match the accuracy of well-tuned local models, they offer significant practical advantages, including simplified model management, reduced training time, and better scalability. These findings highlight the trade-offs between accuracy and efficiency, and provide valuable insights into when global forecasting models may serve as viable alternatives to individualized local models.

Keywords: Time Series, Demand Forecasting, Machine Learning, LightGBM, XGBoost, Statistical Model, Global Model, Local Model.

Acknowledgements

We would like to sincerely thank Lukas Josefsson, Katarina Floberg, Bettina Linder, and Christian Jensen for giving us the opportunity to be part of this project. Your guidance, support, and trust throughout the process have been invaluable, and the experience has greatly contributed to our professional and personal growth. We are truly grateful for the chance to learn and contribute this has been an important step in shaping our future.

We would also like to express our sincere gratitude to Stefano Mannelli and Ashkan Panahi for encouraging and supporting the academic direction of this project.

Andres Felipe Otero Salamanca and Elínborg Ása Ásbergsdóttir, Gothenburg, June, 2025

Contents

List of Figures

List of Tables

List of Acronyms

ACF	Autocorrelation Function
AI	Artificial Intelligence
ARIMA	AutoRegressive Integrated Moving Average
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
DL	Deep Learning
EFB	Exclusive Feature Bundling
GBDT	Gradient Boosting Decision Trees
GOSS	Gradient-based One-Side Sampling
LightGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MAE	Mean Absolite Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
TFT	Temporal Fusion Transformer
TPE	Tree-structured Parzen Estimator
XGBoost	Extreme Gradient Boosting

1

Introduction

In an increasingly complex and dynamic industrial landscape, accurate demand forecasting is essential for efficient supply chain management. Some companies currently rely on manual forecasting processes, making demand prediction highly dependent on human expertise. While traditional statistical methods have historically supported supply chain forecasting, they often struggle with non-linearity, external disruptions, and scalability. Machine Learning (ML) offers a data-driven alternative that can capture complex patterns, improve forecasting accuracy, and reduce reliance on manual intervention [1].

This thesis explores the application of ML techniques to enhance unconstrained demand forecasting. The research focuses on developing predictive models for products at the company we are working with. The goal is to assess the feasibility of ML-based forecasting and its potential to improve decision-making by reducing forecasting errors and optimizing inventory planning.

The study will begin by developing models for predicting sales for the main products, given its critical role in the companys supply chain. By leveraging ML techniques such as time-series models and gradient boosting methods, this research aims to evaluate the advantages and challenges of integrating ML into an industrial forecasting workflow.

Beyond algorithmic development, this thesis also considers practical challenges such as data availability, feature engineering, and model interpretability. The outcome of this research will contribute to the companys transition from manual forecasting to a more automated, data-driven approach, offering insights into the broader applicability of ML in industrial supply chains.

1.1 Purpose

The objective of this work is to design and implement ML models that deliver improved forecasting performance over the company's current methods, targeting unconstrained demand across specific markets, segments, and product categories.

The key research questions that will help explore the impact of machine learning on unconstrained demand forecasting are:

- How can machine learning techniques improve the accuracy of unconstrained demand forecasting?
- Which features have the most impact when modeling unconstrained demand?
- To what extent can a global forecasting model generalize across different products, compared to individualized local models, in predicting unconstrained demand?

The first question helps with investigating how different machine learning methods can improve forecasting accuracy, providing insights into which approaches are most effective for this specific context. The second question focuses on the impact of input features on forecasting to understand which ones contribute most to model performance because recognizing the most influential features is essential for developing robust and reliable predictive models. Lastly, the third question aims to evaluate the extent to which a global forecasting model can generalize across different products in predicting unconstrained demand compared to individualized local models. This investigation seeks to determine whether a global model can offer comparable or superior forecasting performance, potentially improving scalability and consistency in current demand forecasting practices.

Each of these questions is explored in dedicated sections of this report. The improvement in forecast accuracy through machine learning models is analyzed in Section 5.1, where model performance is benchmarked against traditional statistical methods. Section 5.2 investigates feature importance, highlighting the most impactful predictors derived from sales history and other relevant variables. Finally, Sections 5.1 and 5.3 compares the effectiveness of global versus local and cluster-based forecasting models, discussing their relative strengths and limitations across different product categories.

1.2 Background

Research in supply chain forecasting has demonstrated the transformative potential of ML and Deep Learning (DL) to surpass traditional methods by leveraging large datasets and uncovering complex patterns. For example, Seyedan and Mafakheri [1] emphasize the role of big data analytics in improving forecasting precision, reducing supply chain inefficiencies, and mitigating uncertainties through advanced methods, like time-series analysis, regression, and neural networks.

Traditional forecasting methods in supply chain management include spreadsheet models and simple statistical techniques, such as moving averages, exponential smoothing, and ARIMA [1]. Some of their key limitations include poor scalability, limited ability to capture the complexities of modern supply chains, heavy reliance on planners' expertise, and a lack of integration with external market factors. These methods are limited by their dependence on predefined relationships and historical trends, making them less effective in predicting fluctuations.

In contrast, ML-based models can process large amounts of data and account for non-linear relationships, making them more robust and accurate [1]. Their ability

to incorporate a wider range of inputs also makes them well-suited for handling the impact of organizational and external factors that often influence demand, such as marketing strategies, economic instability, and inflation. By integrating these external variables, machine learning models can improve their adaptability to future demand shifts.

Similarly, Shavaki and Ghahnavieh [2] highlight the effectiveness of DL techniques, such as long short-term memory (LSTM) and convolutional neural networks (CNNs). The findings show that in some cases, LSTMs outperformed traditional methods like ARIMA and basic neural networks, particularly for capturing long-term dependencies. Additionally, CNNs have been primarily used for feature extraction and quality management in supply chain forecasting. However, there are notable limitations where LSTMs and CNNs require significant computational power, making them challenging to deploy in real-time applications. Moreover, these models depend on large, high-quality datasets, and their performance is significantly impacted by noisy or imbalanced data.

Furthermore, studies such as Song and Chen [3] have shown the effectiveness of Light Gradient Boosting Machines (LightGBM) in enhancing financial time series forecasting in the shipping market, demonstrating their potential for accurate and efficient predictions. This is further supported by the results of the M5 Competition [4], where LightGBM-based models ranked among the top-performing approaches, highlighting their robustness, scalability, and adaptability in complex forecasting scenarios across various industries.

1.3 Ethical Considerations

The application of ML in supply chain forecasting raises several ethical considerations. Data privacy and security in supply chain forecasting includes the responsibility of safeguarding sensitive business information. Improper handling of data can expose confidential operational details, supplier relationships, and business insights. This can damage trust, harm a companys reputation, or create competitive disadvantages. It is important to handle data carefully, ensuring that its use, storage, and transfer follow principles of confidentiality, accountability, and integrity.

The increasing reliance on advanced machine learning models in supply chain forecasting introduces ethical concerns about dependence on technology. While these models improve accuracy and efficiency, their complexity may require a deep understanding of both the models and the features used in forecasting.

While machine learning models can provide highly accurate forecasts, their complexity can lead to a lack of transparency. From an ethical standpoint, it is important to be able to explain how forecasts are made, even if the exact inner workings of the model remain complex. For this, feature importance analysis will be used to identify which groups of features are most informative for the forecasts. This provides stakeholders with insights into what factors influence the predictions, without needing to explain the impact of each feature individually.

This project explores multiple models and a broad set of features, which increases the complexity of interpretation and reduces the transparency and trustworthiness of the system's outputs. If the models fail to perform as expected, human intervention is still necessary to correct predictions, which raises questions about accountability and decision-making. Striking a balance between trust in automated systems and human expertise is essential to ensure that critical decisions remain under human oversight and to avoid overreliance on AI-generated forecasts.

2

Theory

This chapter contains the theoretical background for the methods used. It begins with an overview of time series and demand forecasting. The chapter then introduces key forecasting approaches, starting with the statistical method ARIMA, which has been widely used due to its simplicity and interpretability. Following this, the focus shifts to machine learning techniques, particularly gradient boosting decision trees, with detailed discussions of the XGBoost and LightGBM algorithms. Finally, the chapter outlines the evaluation metrics employed to assess forecasting performance, providing the necessary framework for interpreting and comparing model results in subsequent chapters.

2.1 Time Series

Time series data represents observations collected sequentially over time at regular intervals, forming the basis for temporal pattern analysis [5]. These datasets differ fundamentally from cross-sectional data due to their inherent temporal ordering and autocorrelation structure. The primary objective of time series analysis is to identify and model the underlying patterns that govern the data generation process while accounting for temporal dependencies. Key characteristics of time series include autocorrelation, stationarity, seasonality, and trends [6], each requiring specific modeling considerations to ensure accurate forecasting. Understanding these properties is essential for selecting appropriate forecasting methods and interpreting their results effectively.

2.1.1 Autocorrelation and Stationarity

Autocorrelation refers to the correlation of a time series with its past values [7]. It indicates how strongly current values depend on previous observations and measures the linear relationship between lagged values of a time series. This quantifies how current values in a series relate to past values at fixed lags and is typically described using the autocorrelation function (ACF):

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}, \tag{2.1}$$

where $\rho(s,t)$ is the autocorrelation between time points s and t, $\gamma(s,t)$ is the covariance between the values of the time series at time points s and t, and $\gamma(s,s)$ and $\gamma(t,t)$ are the variances at time points s and t, respectively.

In time series analysis, stationarity refers to the property that the statistical characteristics of a time series do not change over time [7]. This means that the process that generates the time series has a constant mean and variance, and the autocovariance between two points in time depends only on the lag or time difference between them, not on the specific times themselves. Stationarity is crucial because it allows for meaningful statistical analysis and forecasting, as it implies a certain regularity in the data's behavior. A non-stationary series may exhibit trends or seasonal patterns, which can make it harder to model without transformation.

2.1.2 Trends and Seasonality

Trend refers to the long-term movement in a time series, indicating a sustained increase or decrease in the data over time. It captures the underlying direction of the data, which may be influenced by external factors such as economic growth, technological advancements, or demographic shifts.

Seasonality refers to repetitive and predictable fluctuations in a time series that occur at regular intervals. These variations are often driven by external recurring factors such as climate, holidays, or economic cycles.

2.2 Demand Forecasting

Demand forecasting is the process of estimating future customer demand for a product or service based on historical data, statistical techniques, and machine learning models. It is essential for supply chain management, inventory optimization, and production planning, enabling businesses to allocate resources effectively and reduce uncertainties in decision-making [8].

Mathematically, demand forecasting can be expressed as:

$$D_t = f(H_t, E_t, M_t), \tag{2.2}$$

where D_t is the forecasted demand at time t, H_t represents historical demand data, E_t accounts for external factors (e.g., economic indicators, weather), and M_t includes market conditions and consumer behavior.

Unconstrained demand refers to the total customer demand for a product or service, presuming there are no supply limitations [9]. It represents the quantity that customers would purchase if the products were always available, without any stockouts, backorders, delays, or capacity constraints. This concept is important for businesses aiming to understand the true market demand, as it separates actual sales from potential sales lost due to supply limitations.

2.3 Statistical Methods

Statistical methods rely on historical data and mathematical models to identify patterns and relationships in demand over time. These approaches assume that historical trends, seasonality, and correlations will persist into the future, making them useful for generating forecasts based on observed behavior. By modeling the underlying structure of time series data, statistical methods can capture important elements such as long-term trends, cyclical variations, and short-term fluctuations. They are often valued for their interpretability and solid theoretical foundation and are widely used in industries where historical demand data is available and relatively stable over time. One of the most well-known and commonly used statistical forecasting techniques is the ARIMA model, which will be described in more detail in the following section.

2.3.1 ARIMA

The AutoRegressive Integrated Moving Average (ARIMA) model is a foundational statistical method for univariate time series forecasting. It is particularly effective for series that exhibit autocorrelation and are non-stationary but can be made stationary through differencing [10]. Hyndman and Athanasopoulos provide a comprehensive and practical guide to ARIMA, including parameter estimation, model diagnostics, and forecasting techniques.

The autoregressive (AR) part of the model assumes the current value of a series depends linearly on its previous values [10]. For an AR model of order p, denoted AR(p), the equation is:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \tag{2.3}$$

where y_t is the value at time t, ϕ_i are parameters to be estimated, and ε_t is white noise.

The integrated part involves differencing the time series d times to remove trends and achieve stationarity:

$$y'_t = y_t - y_{t-1} \tag{2.4}$$

If first-order differencing is insufficient, second-order or higher differencing can be used. This step is crucial because ARIMA models assume the underlying time series is stationary [10].

The moving average (MA) part models the future values as a linear combination of past error terms [10]. An MA model of order q, denoted MA(q), is defined as:

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{2.5}$$

where ε_t are the past error terms and θ_t are the parameters.

By combining the autoregressive (AR), differencing (I) and moving average (MA) components, we obtain the ARIMA(p, d, q) model. It can be written compactly as:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{2.6}$$

This form captures both the autoregressive structure and the moving average of the noise while accounting for differencing to ensure stationarity. ARIMA models are widely used due to their interpretability and effectiveness in capturing linear patterns in time series data. They perform well for short- to medium-term forecasting and are supported by mature statistical theory [10]. However, they are limited to univariate series, assume linearity, and struggle with capturing complex seasonal or nonlinear patterns unless extended (e.g., SARIMA or ARIMAX) [10].

2.4 Machine Learning Methods

Machine learning methods use data-driven approaches to identify complex, nonlinear relationships in demand patterns. These methods leverage large datasets and advanced computational techniques to improve forecasting accuracy. Machine learning differs from statistical modeling by not relying on assumptions concerning the data distribution and the structure. It learns the patterns directly from the data, which will allow it to capture dependencies and interactions that may be difficult to model. This makes them particularly useful for dynamic or high-dimensional environments where traditional models may fail. Machine learning has been developing rapidly in the last few years, and the use of these methods has become popular within demand forecasting because of their flexibility, scalability, and fast learning to change patterns. The following subsections provide an overview of tree-based ensemble techniques, including gradient boosting decision trees and their widely used variants, XGBoost and LightGBM.

2.4.1 Gradient Boosting Decision Tree

Gradient Boosting Decision Trees (GBDT) is a popular ensemble learning technique employed for high accuracy and efficiency in various machine learning tasks, such as regression. In a forward stage-wise manner, GBDT constructs an additive model, where the negative gradients of the loss function, often referred to as residual errors, are fitted with decision trees iteratively, and the loss function is thereby consistently minimized over time [11], [12].

The basic principle of gradient boosting is to sequentially train weak learners, typically decision trees, in a way that each new tree corrects the errors of the previous ones. Let $F_m(x)$ denote the predicted value of the m-th tree and the final model is given by the sum of all previous models:

$$F(x) = \sum_{m=1}^{M} F_m(x), \tag{2.7}$$

where M is the number of trees and x represents the feature vector for a given instance.

The process begins with an initial model $F_0(x)$ that is trained by minimizing a loss function L, which measures the prediction error over a training set of input-output pairs $\{(x_i, y_i)\}_{i=1}^n$:

$$F_0(x) = \arg\min_F \sum_{i=1}^{n} L(y_i, F(x_i)) \tag{2.8}$$

where y_i are the true target values and $F(x_i)$ are the predicted values.

At each iteration m, the algorithm computes the negative gradient of the loss function with respect to the current models predictions $F_{m-1}(x_i)$ and fits a weak learner $h(x; a_m)$ to pseudo-residuals $(\tilde{y}_i)$:

$$\tilde{y}_i = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F=F_{m-1}}. \tag{2.9}$$

Then, the model is updated as:

$$F_m(x) = F_{m-1}(x) + \nu \cdot \rho_m h(x; a_m), \tag{2.10}$$

where ρ_m is the optimal step size found by minimizing the loss along the direction of h, and $\nu \in (0, 1]$ is a learning rate parameter (also called shrinkage), which acts as a form of regularization. This procedure allows the model to gradually improve its predictions by moving in the direction that most reduces the loss.

2.4.2 XGBoost

XGBoost (Extreme Gradient Boosting) is a supervised machine learning algorithm that builds upon the principles of gradient boosting. It was introduced by Chen and Guestrin (2016) and has gained popularity due to its efficiency, scalability, and predictive power across various tasks such as classification and regression [13].

2.4.2.1 Objective Function and Regularization Term

XGBoost is based on the gradient boosting framework, where it constructs models sequentially, with each model attempting to correct the errors made by the previous one. Unlike traditional GBDT, XGBoost includes a regularization term to prevent overfitting and encourage simpler models, defined as:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2, \tag{2.11}$$

where T is the number of leaves in the tree, w_j is the weight of the j-th leaf, γ controls the complexity of the tree (the number of leaves), and λ is the L2 regularization term, which penalizes large leaf weights to prevent overfitting [13].

The objective in XGBoost is to minimize a regularized objective function given by:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \Omega(f), \tag{2.12}$$

where $l(y_i, \hat{y}_i)$ is a differentiable convex loss function that measures the difference between the true value y_i and the predicted value $\hat{y}_i$, and $\Omega(f)$ is the regularization term that penalizes model complexity.

2.4.2.2 Optimization Using Second-Order Approximation

XGBoost utilizes a second-order Taylor expansion of the loss function to improve optimization:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t), \tag{2.13}$$

where $g_i = \frac{\partial l(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ is the first derivative (gradient) and $h_i = \frac{\partial^2 l(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$ is the second derivative (Hessian) of the loss with respect to the predicted output. This approximation allows the algorithm to quickly optimize the objective.

2.4.2.3 Optimal Weight for Each Leaf

To determine the optimal weight w_j for each leaf j in the decision tree, the following formula is used:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \tag{2.14}$$

where I_j represents the set of data points assigned to leaf j, and λ is the regularization parameter. The corresponding optimal value after assigning the optimal weights is:

$$\mathcal{L}^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{2.15}$$

This formulation is used during tree construction to determine the best structure and splits.

2.4.2.4 Advantages and limitations of XGBoost

One of the main advantages of XGBoost is its speed and performance. The algorithm supports parallelized tree construction, cache optimization, and efficient handling of missing values, making it well-suited for large-scale data problems [13]. It also achieves high performance by leveraging both first and second-order derivatives. Furthermore, XGBoost allows for both L1 and L2 regularization, giving users more control over model complexity than earlier gradient boosting frameworks [14]. These features make XGBoost particularly suitable for structured data and competitive ML tasks.

However, XGBoost is not without limitations. A major drawback is its complexity, where the large number of hyperparameters can be overwhelming and make model tuning time-consuming. Moreover, being a tree-based method, it may require many trees or very deep trees to capture highly complex patterns, increasing computational demands.

2.4.3 LightGBM

Light Gradient Boosting Machine (LightGBM) is an efficient and high-performance GBDT framework developed by Microsoft (2017) [15]. It is designed to handle large datasets efficiently and to reduce memory usage during training. The main innovation behind LightGBM is its histogram-based approach to constructing decision trees, which significantly accelerates training and reduces memory consumption. Furthermore, LightGBM uses a leaf-wise growth strategy instead of the level-wise growth used in other GBDT algorithms, such as XGBoost.

Implementations of GBDT, including XGBoost, suffer from scalability limitations in high-dimensional feature spaces and large datasets. This is mainly due to the high computational costs associated with scanning all data points and evaluating all possible split points on each feature while constructing decision trees. To tackle these challenges, LightGBM introduces two novel techniques, GOSS (Gradient-based One-Side Sampling) and EFB (Exclusive Feature Bundling), which together dramatically enhance training speed and scalability while preserving accuracy.

2.4.3.1 Leaf-wise Growth Strategy

Unlike traditional gradient boosting methods that split trees level-wise, creating splits at all nodes at a given level, LightGBM uses a leaf-wise growth strategy. In this strategy, at each iteration, the algorithm chooses the leaf with the maximum delta loss to split (see Figure 2.1). This results in a deeper tree with fewer splits and better fit, as it optimizes the growth of the tree according to the residuals.

The leaf-wise strategy aims to select the split that yields the greatest reduction in the loss function, given a set of leaves L. The leaf split criterion is therefore chosen to maximize this reduction in loss, which can be expressed mathematically as:

$$\Delta\mathcal{L}_l = \mathcal{L}(y, F(x)) - \mathcal{L}(y, F(x) + \Delta F_l) \tag{2.16}$$

where ΔF_l represents the change in prediction for leaf l and $\Delta \mathcal{L}_l$ represents the reduction in loss from that change.

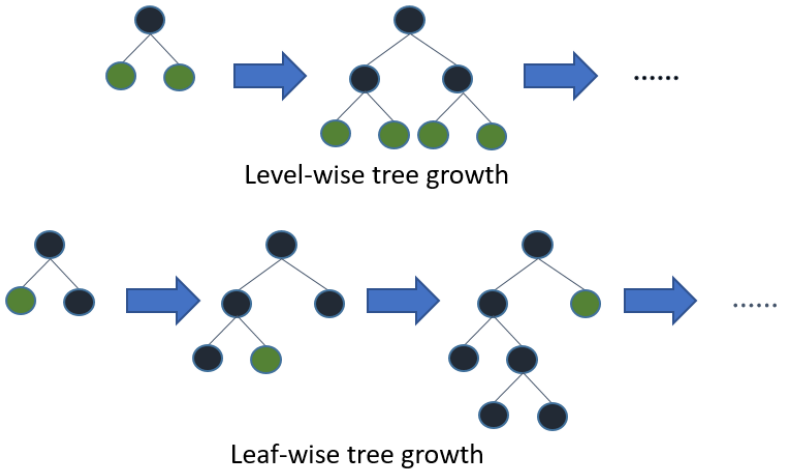

Figure 2.1: Level-wise and leaf-wise growth strategy [16]

2.4.3.2 Histogram-Based Approach

The histogram-based approach used in LightGBM reduces the number of unique values in the feature space by bucketing the continuous features into discrete bins. This reduces computation time and memory usage, as the algorithm only needs to consider a limited number of bins instead of all the raw feature values. The binning operation can be mathematically expressed as:

$$b(x_i) = \text{bin}(x_i, \{b_1, b_2, \ldots, b_k\}) \tag{2.17}$$

where $b(x_i)$ represents the binned value of feature x_i, and $\{b_1, b_2, \ldots, b_k\}$ are the predefined bin edges. This reduces the problem size, especially for high-cardinality features.

2.4.3.3 GOSS

One of the key innovations that makes LightGBM both fast and accurate is its use of Gradient-based One-Side Sampling (GOSS) [15]. This technique improves training efficiency by focusing more on the training examples that the model is currently predicting poorly. In gradient boosting, each training instance has an associated gradient that reflects how much error the model is making on that point. Large gradients mean the model is struggling to fit that instance, while small gradients indicate the model is already doing well.

GOSS takes advantage of this by keeping top $a \times 100\%$ instances with large gradients since they carry more information about what the model still needs to learn. From the rest, it randomly selects a smaller portion $b \times 100\%$. GOSS enhances the importance of instances with small gradients by amplifying their contribution

by a constant $\frac{1-a}{b}$ to the information gain calculation, helping the model focus on under-trained examples while preserving the overall data distribution. This allows the model to train on a reduced set of data without losing important information, significantly speeding up the process.

2.4.3.4 EFB

An important technique that helps LightGBM train faster without sacrificing accuracy is Exclusive Feature Bundling (EFB) [15]. This method is designed to reduce the number of features the model has to process, especially in high-dimensional datasets where many features are sparse. In many real-world datasets, particularly those that include one-hot encoded categorical variables, features often do not take non-zero values at the same time. These are called mutually exclusive features. LightGBM takes advantage of this sparsity by bundling mutually exclusive features together into a single new feature. Since the original features do not overlap in their non-zero positions, they can be safely combined without losing information. This reduces the dimensionality of the input data and helps speed up both memory use and computation.

2.4.3.5 Strengths and Limitations

LightGBM offers several advantages over traditional GBDT methods. First, its histogram-based approach significantly reduces memory usage and accelerates training, making it suitable for large datasets. Second, its leaf-wise growth strategy allows for more accurate models with fewer splits. However, the leaf-wise strategy can sometimes lead to overfitting, especially for small datasets, as it may create deeper trees. To mitigate this, hyperparameters such as `max_depth` and `num_leaves` must be carefully tuned. Additionally, LightGBM's complexity makes it less interpretable than simpler models, like decision trees or linear regression [15].

2.5 Forecasting Evaluation Metrics

Six metrics were chosen to evaluate the performance of the ML models used for forecasting unconstrained demand. They are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). These metrics are widely used in regression tasks and provide different perspectives on model accuracy and error behavior [17]. In demand forecasting, selecting appropriate evaluation metrics is essential, as they will influence how models are compared and interpreted. Each of the chosen metrics captures different aspects of the prediction error, helping to identify models that not only perform well in general but also handle outliers, scale, and interpretability in a meaningful way.

The following formulas define each of the evaluation metrics, where n is the total number of observations, y_i is the actual value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of n actual values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.19}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.20}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \cdot 100\% \tag{2.21}$$

Mean Squared Error (MSE) is a common evaluation metric that calculates the average of the squared differences between predicted and actual values. Because it squares the errors, it places greater emphasis on larger mistakes [18], which can be especially helpful in demand forecasting, where a big deviation in predicting high-demand items can lead to lost sales or excess inventory. However, one downside of MSE is that the result is expressed in squared units, which can make it harder to interpret in real-world terms. It also tends to be sensitive to outliers, sometimes pushing models to overcompensate for rare large errors at the expense of accuracy in more typical cases.

Root Mean Squared Error (RMSE) is the square root of the MSE, which brings the error back to the same unit as the original data. This makes RMSE easier to interpret while still keeping the benefit of penalizing larger errors more heavily [18]. In forecasting, RMSE is widely used because it strikes a good balance between highlighting big errors and remaining understandable. However, like MSE, it can still be quite sensitive to outliers. This means that while RMSE is good for detecting large prediction mistakes, it might not always represent the average performance across all forecasts accurately.

Mean Absolute Error (MAE) calculates the average magnitude of errors without considering their direction. Unlike MSE and RMSE, MAE treats all errors equally, which makes it a simpler and more robust metric for when the goal is to understand overall prediction accuracy without giving extra weight to outliers [18]. In demand forecasting, MAE provides a clear indication of how far-off predictions are on average, making it highly interpretable and useful in real-world scenarios. However, its main limitation is that it does not penalize larger errors more severely, which can be a drawback in cases where large forecasting mistakes carry higher consequences.

Mean Absolute Percentage Error (MAPE) expresses prediction errors as a percentage of the actual values, which makes it easy to interpret and useful for understanding how large the errors are relative to the size of the demand. This is particularly helpful when comparing forecast accuracy across different products or time periods that vary greatly in scale [18]. Despite its interpretability, MAPE has notable limitations, where it becomes unreliable when actual values are close to zero, often

leading to large or undefined error percentages. Another issue is that MAPE tends to favor models that underpredict rather than overpredict since it penalizes positive errors (underestimates) less than negative ones (overestimates). This bias can lead to misleading conclusions when comparing models, as it might make a consistently underestimating model appear more accurate than one with balanced predictions [18].

3

Methods

This chapter explains the data and methods involved in the project. It starts with an outline of the dataset and preprocessing steps carried out to prepare the data for modeling, including using feature engineering methods to improve the model performance. An outline of different forecasting models training procedures will then follow before a discussion of evaluation metrics used to assess the performance of the models.

3.1 Data

The data used in this project consists of historical sales records for different products. It is structured as time series data, organized across several hierarchical levels, including market, branch/plant, and multiple product levels. The scope was narrowed down to a single market and one specific product segment. Within this scope, the forecasting was carried out at two key levels of granularity, the branch/plant level and product level 1, which is one of the higher aggregation levels in the product hierarchy (see in Figure 3.1). Due to the nature of the business, the volume of data is relatively limited, especially when compared to e-commerce companies that can track millions of transactions daily. As a result, the smallest feasible granularity for this project is at the monthly level. Accordingly, all analyses and development of the models are based on monthly data, which provides a balance between capturing meaningful trends and having enough data points for making reliable forecasts.

The target variable in this project is a time series representing the number of units sold based on the requested ship date. This variable reflects customer demand as it was originally requested and not necessarily when the product was shipped, which is important for capturing the actual timing of demand. For this project, where the goal is to forecast unconstrained demand, that is, the demand that would occur without limitations, the requested ship date helps in estimating such unconstrained demand by focusing on when customers wanted the products, as opposed to when the company was able to deliver them.

The available data used in this project spans a 10-year period. However, all instances in the dataset do not cover the entire time period. The periods for individual products range from approximately 6 to 10 years, depending on factors such as market introduction timing.

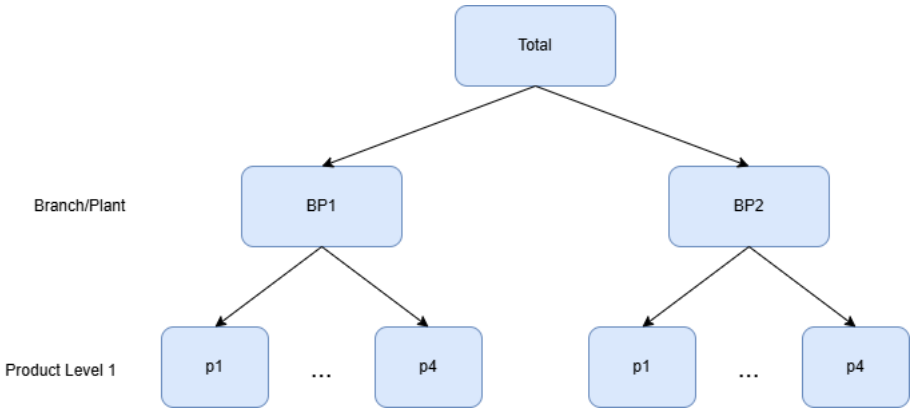

Figure 3.1: Tree diagram representing the data

In addition to differences in time span, the products in the dataset also vary in terms of demand volume and temporal behavior. Based on observed sales patterns, they can be broadly grouped into low, medium, and high demand categories. These groupings were considered when selecting representative cases for analysis, as products with different demand volumes often exhibit distinct temporal patterns, variability, and forecasting challenges. Moreover, the products show diverse time series characteristics, where some follow relatively stable trends over time, while others display dynamic behaviors such as seasonality, sudden shifts, or long-term growth or decline. These variations in both demand volume and structural patterns have important implications for model performance and guide the choice of appropriate forecasting approaches.

3.1.1 Preprocessing

The preprocessing stage involved addressing data quality challenges to ensure the reliability of the forecasting models. First, it was necessary to remove duplicate rows created by the same product being recorded multiple times as it was shipped through various intermediate locations before reaching its final delivery destination. Additionally, a logic was developed to identify the items correctly, which are related to the selected products, as orders may have multiple components.

Another critical aspect was handling outliers. Although these anomalies do not necessarily indicate data errors, they can significantly impact model performance if not properly addressed. For example, a product that is typically ordered fewer than 100 times per month may appear with an unusually high volume, such as 300 units in a single month, due to one-off external factors.

To improve the reliability of the total units (target variable) and reduce the influence of extreme or erroneous values, a forecast-based outlier-aware adjustment method was applied as suggested in [6], Chapter 13. This procedure was performed individually for each time series in the dataset. First, a simple two-period moving average forecast was calculated to establish a baseline estimate of expected values for total units. The forecast error, defined as the difference between the observed and forecasted values, was then computed for each time point. Assuming that these

errors follow a normal distribution, their probabilities were estimated using the cumulative distribution function (CDF). Observations with error probabilities falling outside the central 98% of this distribution (i.e., below 0.01 or above 0.99) were considered outliers and excluded from further calculations. The remaining errors were used to recalculate the mean and standard deviation, from which new upper and lower thresholds were derived using the 1st and 99th percentiles. Finally, the original total unit values were clipped to fall within these updated bounds, ensuring that extreme values were adjusted to a reasonable range. All adjusted values were rounded down to the nearest whole number, and a non-negativity constraint was applied to avoid negative demand values. This method ensures that the time series retains its temporal structure while mitigating the impact of anomalies, thereby supporting more robust model training and evaluation.

3.1.2 Feature Engineering

Though XGBoost and LightGBM are not generally perceived to be for time series forecasting, these gradient boosting frameworks, built for supervised learning tasks such as regression and classification, can still be effectively applied to time series problems through appropriate feature engineering and data transformation.

In the context of time series forecasting, the task is reframed as a supervised regression problem. The target variable represents the future value of the time series, that is, monthly sales, and the input features consist of lagged values and relevant covariates. This setup allows models such as XGBoost and LightGBM to learn from historical patterns and external influences.

A general formulation is given by:

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, X_t), \tag{3.1}$$

where y_t is the target variable (i.e., demand at time t), $y_{t-1}, \ldots, y_{t-p}$ are lagged observations of the same time series, and X_t includes exogenous variables or calendar-based features.

The data preparation was carried out with a rolling or expanding window approach to avoid data leakage and keep the time structure of the problem. Only past data was used to predict future values, so the models would not have any information about the future during training. Although the machine learning models used in this project do not explicitly account for time series components such as trend or seasonality, they can capture these patterns through proper feature engineering. By constructing time-based features, such as time lags or calendar-related variables, it is possible to encode temporal dynamics in a way that allows the models to learn and account for them during training.

Both XGBoost and LightGBM offer advantages such as handling missing data, being robust against outliers, and being able to model nonlinear relationships and complex interactions between features. These properties make them well-suited for forecasting tasks, especially when classical models struggle with higher-dimensional

data or nonlinear patterns. Moreover, their popularity in forecasting competitions (such as the M5 Forecasting Challenge) [19] demonstrates their effectiveness when combined with careful preprocessing and domain-specific feature engineering.

For these models the following temporal features were used:

- Month of the year: The month of the year is an important feature in capturing seasonal patterns inherent in time series data. Time series often exhibit periodic behavior tied to the calendar year, and the month variable allows the model to learn these repeating cycles.
- Year: The year feature captures long-term trends and helps the model understand the evolution of the data over time. Over multiple years, time series data may reflect structural changes such as shifts in consumer behavior, economic conditions, or business growth.
- Month of the year cyclical feature: This feature is encoded using sine and cosine transformations to capture the cyclical nature of months. Such encoding preserves the continuity between months, ensuring that temporally adjacent months (e.g., December and January) are treated as closely related. Without this representation, months would be interpreted as independent categorical values, potentially leading the model to overlook important seasonal patterns.

For these models the following statistical features features were used:

- 6-15 steps lagged values: Past observations in the time series used to predict future values. These features help the model capture temporal dependencies in the data, where the value at a given time is influenced by previous values (e.g., values from past months)
- {3, 7, 10, 15} steps rolling values: Rolling statistics, such as moving averages and moving standard deviations, are important for smoothing the time series and capturing underlying trends.
- Mean, standard deviation, maximum, and minimum values: These descriptive statistics provide an overall summary of the distribution of values in the time series. The mean represents the typical level of the data, the standard deviation indicates the variability, and the maximum and minimum values give the range of observed values.

These statistical features are grouped based on different categories, such as ID, branch/plant, or hierarchical level. Grouping statistical features in this manner plays a critical role in enhancing the quality and accuracy of forecasting models. This stratification enables the identification and modeling of localized patterns that may not be apparent when analyzing aggregated data. For instance, demand behavior can vary significantly across individual products (IDs), operational sites (branches/plants), or managerial tiers within a supply chain hierarchy. By recognizing these differences, the forecasting framework becomes more sensitive to the specific temporal dynamics and structural variations that exist within each subgroup.

After completing the necessary preprocessing steps and applying the feature engineering techniques, a comprehensive set of features was constructed to capture the relevant temporal patterns. These features include both raw variables and derived attributes designed to improve model performance and interpretability. In total, 47 features were used as inputs for the ML models, which formed the foundation for the subsequent training and evaluation of the forecasting models.

3.2 Cross-Validation

Cross-validation is a widely used technique for evaluating a model's forecasting performance. The dataset is divided into several subsets, or folds, and in each iteration, the model is trained on most of these folds and validated on the remaining one. This process is repeated several times with different train-validation splits. By averaging the results across these iterations, cross-validation reduces the risk of heavily biased evaluations caused by a specific data split, providing a more reliable estimate of the models ability to generalize to unseen data.

In this project, cross-validation was applied to assess the predictive performance of the LightGBM and XGBoost models. Both models were trained using their built-in cross-validation functions, which efficiently handle the process internally. The dataset was split into four equally sized folds. In each iteration, one fold was held out as the validation set while the model was trained on the other three. This process was repeated four times, and the performance results were averaged across all folds to obtain a final evaluation score.

Using cross-validation was especially important given the variability in data availability across products. It helped reduce the risk of overfitting and provided a more reliable basis for comparing the models. Moreover, by leveraging the built-in cross-validation functions in LightGBM and XGBoost, the process benefited from optimized performance and consistent handling of model training and evaluation.

3.3 Model Training

Given the limited data size, i.e. few data points per product, the selected forecasting models are ARIMA, XGBoost, and LightGBM. ARIMA is included as a classical statistical benchmark, widely used for time series forecasting, and employed by the company. It was trained using only the historical values of the target variable, which is the total number of units sold over time. This means the model relies solely on past observations in the time series to identify patterns such as trends and seasonality. Unlike more complex machine learning approaches, ARIMA does not incorporate additional explanatory variables, making it a purely data-driven model based on the temporal dynamics of the sales history itself.

XGBoost and LightGBM are chosen based on their strong performance in demand forecasting competitions such as the M5 accuracy competition [19], where most top-performing teams employed tree-based ensemble methods. While deep learning

models like DeepAR [20] or Temporal Fusion Transformer (TFT) [21] are popular in the forecasting literature, these typically require large amounts of high-frequency data to train effectively. In this project, an initial attempt was made to apply DeepAR, but the model did not achieve adequate results, likely due to data sparsity and the complex nature of the demand patterns. Given these limitations, deep learning approaches were not considered suitable for further development in this use case. The selected models are trained and evaluated using monthly sales data across the specified period, with a focus on forecasting accuracy and the ability to handle limited, structured datasets.

To evaluate different modeling approaches [22] and understand their strengths and limitations, the following three approaches were tested.

3.3.1 Global Model

The first approach involves training a global model to forecast demand for all selected products. This setup has the advantage of simplicity and reduced computational cost, as only one model for different items is trained and maintained. Moreover, it allows for shared learning across different items, which can be beneficial when products show similar demand patterns. However, a potential drawback is the loss of specificity, where the model may not effectively capture the unique behavior of individual products that have different demand patterns.

3.3.2 Cluster-Based Model

The second approach is a cluster-based model. In this setup, a number of models are being trained, with each cluster specializing in modeling a particular subset of the time series data. For instance, one cluster may focus on capturing short-term fluctuations, another may model long-term trends, and a third could account for seasonal components.

The cluster-based model strategy involves grouping similar time series into clusters and training a separate forecasting model for each cluster. This approach leverages the intuition that time series with similar temporal patterns, e.g., trend, seasonality, volatility, can benefit from being modeled together. By reducing heterogeneity within each group, the model can better learn relevant patterns without being overwhelmed by noise from unrelated series. This can lead to improved accuracy compared to purely global models, especially in settings where the data exhibits a clear substructure.

The main advantages of cluster-based models include better scalability than local models (fewer models overall) and more flexibility than global models since it is tailored to groups of similar series. However, their performance depends heavily on the quality of clustering, i.e. poor clustering can lead to worse forecasts [23]. Additionally, model management becomes more complex than in purely global approaches, as multiple models must be maintained and monitored, though this is still less burdensome than maintaining one model per series.

3.3.3 Local Model

The third approach adopts a local modeling strategy, in which a given model is trained for each selected instance separately, where one instance is defined as the product level at which the forecast is performed. This allows for a rather precise and tailored forecasting model as every model learns the specific time-dependent characteristics of its instance. It becomes very important in cases when products exhibit different demand behaviors or in the event that enough data is present for each item. The main downside of this approach is that it consumes more resources during computation, and the maintenance overhead will be high because the models have to be trained, validated, and deployed independently. This approach would also be prone to overfitting in some instances if the data available is limited.

3.3.4 Hyperparameter Tuning

Hyperparameter tuning was performed using Optuna [24], which is a modern and efficient hyperparameter optimization framework. It uses Bayesian optimization based on the Tree-structured Parzen Estimator (TPE) to focus the search on promising regions of the hyperparameter space, as opposed to grid search, which exhaustively tests all combinations of fixed grids.

One of Optunas main advantages is that it allows for flexible and dynamic search space definitions. Additionally, it supports early stopping (pruning) of underperforming trials, saving computational resources. These features make Optuna both more scalable and efficient compared to grid search, which becomes impractical as the number of hyperparameters increases. In practice, grid search required significantly more trials to find near-optimal results, many of which were redundant or uninformative. Optuna achieved equal or better performance in fewer trials, making it the more practical approach for this project. Being able to use this tool will help to find not only the best hyperparameters but also the features that will perform the best for the training dataset.

Optuna was used to tune the hyperparameters of both the XGBoost and LightGBM models. The tuning process focused on a set of hyperparameters that are known to significantly influence model performance. For both models, the following hyperparameters were included in the search space: `tweedie_variance_power`, `learning_rate`, `n_estimators`, `subsample`, `colsample_bytree`, `lambda`, and `alpha`, as these control regularization, tree sampling, and overall learning behavior. In addition, XGBoost-specific parameters such as `max_depth` and `gamma` were tuned to adjust tree complexity and pruning behavior. For LightGBM, `num_leaves` and `min_child_samples` were optimized to control leaf growth and overfitting. A description of each parameter can be found in Table A.1 in the Appendix.

The tuning process was implemented using a custom objective function for each model, where Optuna sampled hyperparameters and evaluated their performance using cross-validation. The evaluation metric used was RMSE, and early stopping was applied to avoid unnecessary computation. The function returned the lowest validation RMSE achieved, which Optuna used as the objective value to guide its

search.

3.4 Evaluation and Comparison

A pipeline for model training and testing was developed using the data, ensuring that each model is assessed and optimized. The evaluation of the models involves the metrics MSE, RMSE, MAE and MAPE, which were used to compare the performance of the models. RMSE was chosen as the main evaluation metric because of its common use in academic research related to demand forecasting. Compared to some of the other metrics, RMSE retains the same unit as the target variable, making it easier to interpret in practical terms.

3.5 Tools

The main tools for this project include Python for developing the machine learning models and conducting data analysis, where libraries such as `pandas, NumPy`, and `scikit-learn` were utilized. For the implementation of the forecasting models, the `xgboost` [14] and `lightgbm` [16] libraries are employed. These libraries provide flexible APIs for model training and evaluation, as well as built-in cross-validation functions, which are used in combination with `optuna`, the hyperparameter optimization framework. The project is implemented using the Azure Databricks platform for data processing, machine learning development, and collaboration. The computation is performed on a configured cluster environment consisting of 1-2 worker nodes, each with 32-64 GB of memory and 8-16 cores, and a driver node with 28 GB of memory and 8 cores.

4

Results

This chapter presents the results of the demand forecasting models developed and evaluated in the project. The analysis is based on four different product instances, where each instance represents a time series data of a specific product at a particular branch/plant. These instances were chosen to reflect a range of demand volumes and temporal dynamics. Two of the cases represent low-volume demand, where the number of units sold remains consistently low but with some variability. One of these shows small spikes in demand during the test period, while the other experiences a greater spike right at the transition between the training and test periods, which can challenge the models ability to generalize across time. A third case falls into the medium-volume category, where demand increases sharply during the middle of the test period, presenting a more complex forecasting scenario. The final case represents a high-volume product with frequent sales, selected to illustrate how the models perform on items with substantial demand. By including these varied cases, the evaluation offers insights into how well the models adapt to different demands.

For each instance, the forecasting performance of the machine learning models, i.e., LightGBM and XGBoost, and the different model approaches, i.e., local, cluster-based, and global, is compared to a statistical baseline model, ARIMA. The models are evaluated using the metrics MSE, RMSE, MAE, and MAPE. The results are summarized in tables for each instance, and visual comparisons of predicted versus actual values are provided through line plots. The primary focus of this chapter is to assess how well the machine learning models perform in relation to the baseline across the different evaluation criteria and to identify any consistent patterns in model performance across the selected product instances.

4.1 Model Evaluation Scores

The results for each product are presented in separate tables, Table 4.1 to Table 4.4, where the forecasting models are compared based on their performance in terms of the evaluation metrics. Each table includes one row per model, along with the corresponding approach type. The models are listed in ascending order by their RMSE value, with the best-performing model placed at the top of each table.

Model	Approach Type	MSE	RMSE	MAE	MAPE
LightGBM	local	**3.000**	**1.732**	**1.333**	**50.00**
LightGBM	global	3.500	1.871	1.833	75.0
ARIMA	—	3.500	1.871	1.833	75.0
XGBoost	global	3.500	1.871	1.833	125.0
XGBoost	cluster-based	4.000	2.000	2.000	125.0
XGBoost	local	4.333	2.082	2.000	162.5
LightGBM	cluster-based	5.167	2.273	2.167	187.5

Table 4.1: Model Evaluation Metrics for Product 1

Table 4.1 summarizes the forecasting performance of various models for Product 1. The ARIMA model, used as a benchmark, recorded an RMSE of 1.871, MAE of 1.833, and MAPE of 75.0%. Among all the approaches, LightGBM with a local modeling approach achieved the lowest error values overall, with an RMSE of 1.732, MAE of 1.333, and MAPE of 50.0%. LightGBM using a global approach produced identical results to ARIMA across all metrics, suggesting limited added benefit from global learning in this case. Similarly, XGBoost with a global configuration matched the RMSE and MAE of ARIMA and LightGBM (global) but showed a higher MAPE of 125.0%. The cluster-based models for both XGBoost and LightGBM produced higher errors, with RMSE values of 2.000 and 2.273, and MAPE values of 125.0% and 187.5%, respectively. XGBoost using a local modeling approach had the highest RMSE (2.082) among all non-cluster methods and a MAPE of 162.5%, indicating challenges in capturing the underlying patterns for this product.

Model	Approach Type	MSE	RMSE	MAE	MAPE
XGBoost	local	**2.833**	**1.683**	**1.500**	**50.0**
LightGBM	cluster-based	3.333	1.826	1.667	66.7
LightGBM	global	4.833	2.198	1.833	75.0
LightGBM	local	5.167	2.273	1.833	75.0
XGBoost	global	5.833	2.415	2.167	83.3
XGBoost	cluster-based	6.167	2.483	2.167	**50.0**
ARIMA	—	102.333	10.116	10.000	466.7

Table 4.2: Model Evaluation Metrics for Product 2

Table 4.2 presents the forecasting performance of different modeling approaches for Product 2, illustrating a clear advantage of machine learning models over the ARIMA baseline. ARIMA produced significantly higher error scores, with an RMSE of 10.116, an MAE of 10, and an exceptionally high MAPE of 466.7%. In contrast, XGBoost using a local modeling approach achieved the best overall performance, recording the lowest RMSE of 1.683, MAE of 1.500, and MAPE of 50.0%. LightGBM with a cluster-based approach followed, with slightly higher error values, RMSE of 1.826, MAE of 1.667, and MAPE of 66.7%.

The performance of LightGBM and XGBoost global models showed increased errors relative to their local and cluster-based counterparts, respectively. LightGBM in global mode recorded an RMSE of 2.198 and MAPE of 75.0%, while XGBoost global performed slightly worse with an RMSE of 2.415 and MAPE of 83.3%. Interestingly, XGBoost with a cluster-based setup showed mixed results. Although it had higher RMSE and MAE (2.483 and 2.167, respectively), it achieved the same MAPE score (50.0%) as the best-performing local model, indicating strong performance in relative error terms despite absolute deviations.

Model	**Approach Type**	**MSE**	**RMSE**	**MAE**	**MAPE**
LightGBM	cluster-based	**4.667**	**2.160**	**1.667**	33.3
LightGBM	global	6.500	2.550	1.833	28.1
ARIMA	—	7.500	2.739	1.833	**27.1**
XGBoost	cluster-based	8.334	2.887	2.333	50.0
LightGBM	local	8.667	2.944	2.333	40.6
XGBoost	global	9.500	3.082	2.500	49.0
XGBoost	local	10.000	3.162	2.667	65.6

Table 4.3: Model Evaluation Metrics for Product 3

Table 4.3 presents the performance metrics for forecasting models applied to Product 3. The LightGBM model with a cluster-based approach achieved the lowest MSE (4.667), RMSE (2.160), and MAE (1.667). In terms of MAPE, the lowest value was recorded by ARIMA (27.1%), followed closely by LightGBM with a global approach (28.1%). The XGBoost models showed higher error metrics across all configurations. The local variant reported the highest MSE (10.000), RMSE (3.162), MAE (2.667), and MAPE (65.6%). The global and cluster-based variants of XGBoost also yielded higher errors compared to the LightGBM configurations. Among the LightGBM variants, the local model reported higher RMSE (2.944) and MAE (2.333) than the global and cluster-based versions. All models except ARIMA were outperformed by at least one LightGBM variant in MSE, RMSE, or MAE.

Model	**Approach Type**	**MSE**	**RMSE**	**MAE**	**MAPE**
LightGBM	global	**152.500**	**12.349**	**10.500**	**18.0**
XGBoost	global	238.000	15.427	13.667	24.0
XGBoost	cluster-based	253.333	15.916	10.667	19.4
LightGBM	cluster-based	301.667	17.369	15.667	24.1
LightGBM	local	775.667	27.851	25.000	43.4
XGBoost	local	941.167	30.678	25.833	50.3
ARIMA	—	1250.667	35.379	31.000	42.5

Table 4.4: Model Evaluation Metrics for Product 4

Table 4.4 summarizes the evaluation metrics for forecasting models applied to Product 4. The best overall performance was achieved by the LightGBM model using a global approach, which obtained the lowest error scores across all metrics, i.e.,

MSE (152.500), RMSE (12.349), MAE (10.500), and MAPE (18.0%). XGBoost with a global approach followed, with an MSE of 238.000, RMSE of 15.427, MAE of 13.667, and MAPE of 24.0%. The cluster-based XGBoost model recorded an MSE of 253.333, RMSE of 15.916, MAE of 10.667, and MAPE of 19.4%, while LightGBM with a cluster-based approach showed slightly higher errors across all metrics.

Local approaches for both LightGBM and XGBoost reported substantially higher error values. LightGBM local presented an MSE of 775.667, RMSE of 27.851, MAE of 25.000, and MAPE of 43.4%. The XGBoost local approach had the highest errors with an MSE of 941.167, RMSE of 30.678, MAE of 25.833, and MAPE of 50.3%. ARIMA demonstrated the largest errors overall, with an MSE of 1250.667, RMSE of 35.379, MAE of 31.000, and MAPE of 42.5%.

4.2 Comparison of Predicted and Actual Values

Figures 4.1 to 4.4 present the predicted versus actual values for each product across the end part of the training periods and the whole test periods, allowing for visual evaluation of model performance beyond quantitative error metrics. These plots offer insights into how well each model captures temporal patterns, responds to shifts in demand, and handles volatility.

In Figure 4.1, most models exhibit some ability to track the trend during the training phase for Product 1. During the test period, the local LightGBM model (green line) stands out with predictions that follow the actual values closer than the other model predictions. In contrast, alternative models, such as ARIMA, cluster-based XGBoost, global XGBoost, and global LightGBM, exhibit limited variation during the test period, where they fail to adapt to the actual changes in demand. Their predictions tend to be quite flat or weakly responsive when fluctuations occur. This behavior suggests that these models lack the granularity needed to capture product-specific or local-level trends. The cluster-based LightGBM model and local XGBoost model show some responsiveness to changes in demand, but they tend to both over- and under-react at different times, lagging behind actual demand and failing to capture peaks accurately.

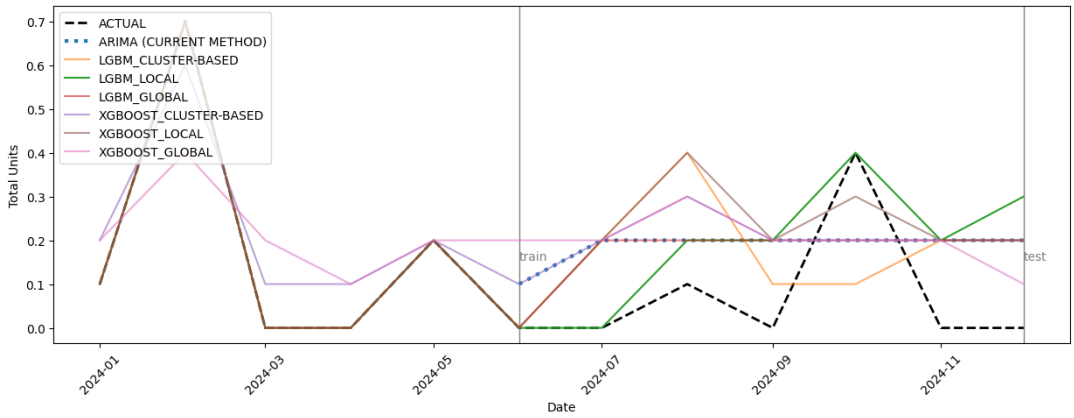

Figure 4.1: Actual and prediction values for Product 1

Figure 4.2 for Product 2 reveals that none of the models are able to closely track the actual values, shown by the black line. While most models manage to capture the timing of the peak at the train-test split, they struggle to reach the magnitude. Instead of adapting to shifts, the predictions from most models follow a similar pattern, where they flatten out and fail to reflect the real fluctuations in demand for the test period. This visual similarity makes it difficult to clearly distinguish which model performs the best, even though the evaluation metrics suggest some differences. The ARIMA model (blue line) is particularly noticeable for its overestimation and flat behavior throughout the test period.

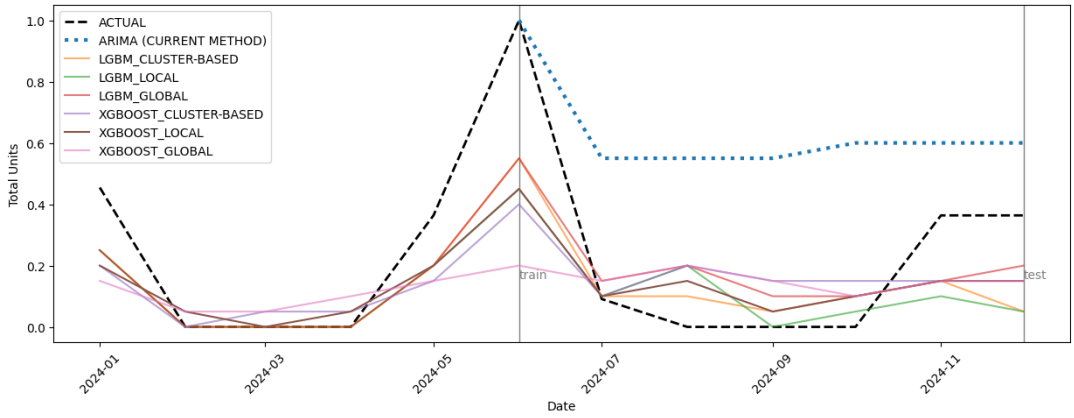

Figure 4.2: Actual and prediction values for Product 2

In Figure 4.3 for Product 3, the test period includes a distinct demand spike. The cluster-based LightGBM model (yellow line) manages to follow the general trend and some turning points of the actual demand (black line) in some way, including the timing of the largest spike and the subsequent drop. Although it underestimates the height of the spike, it seems to outperform the other models by following the shape and dynamics of the demand better. The remaining models either flatten out (i.e., ARIMA) or fail to capture the timing of the changes.

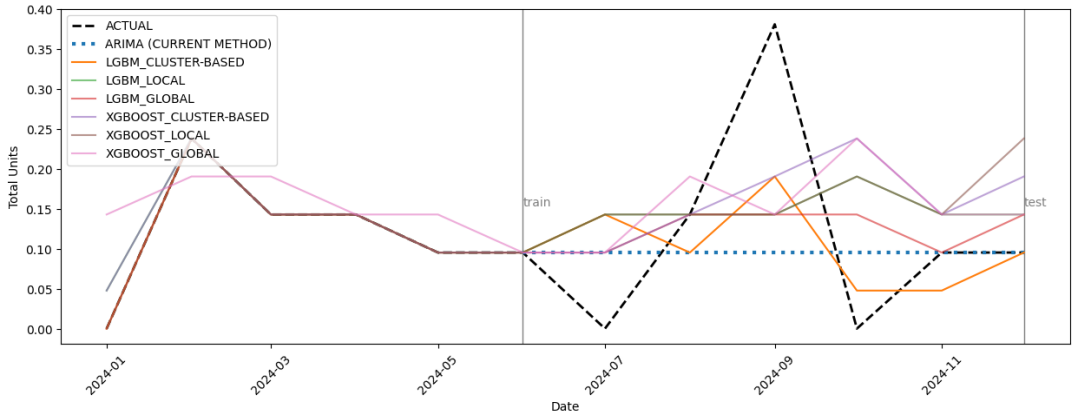

Figure 4.3: Actual and prediction values for Product 3

Figure 4.4 exhibits the actual and predicted values for Product 4. For this product, the global LightGBM model (red line) seems to provide the relatively best forecast during the test period, where it follows the actual sales pattern closer than the others overall. The global XGBoost model follows a similar trend, but is more often further away from the actual values. The cluster-based XGBoost model is smoother than the global models, and the cluster-based LightGBM fails to catch the trends and timings. Other models either underestimate the actual values, as seen with ARIMA, or consistently overshoot them, as observed in the predictions from both local models.

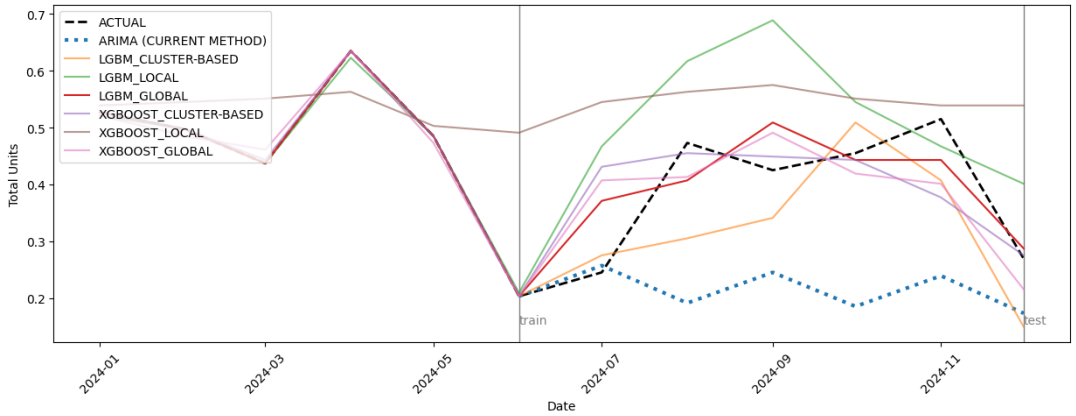

Figure 4.4: Actual and prediction values for Product 4

4.3 Feature Importance

To better understand the behavior of the trained models, feature importance analysis was conducted for both LightGBM and XGBoost models across all three modeling approaches. This analysis was not applied to the ARIMA model, as its forecasts are based solely on the historical sales time series without incorporating any external features.

Gain-based importance was used to evaluate the relative contribution of each feature to the models predictive performance. The gain metric reflects the total improvement in the loss function brought by splits involving a particular feature. Features with higher gain are considered more influential in the models decision-making process.

The features were grouped into five groups in terms of their nature. The names and descriptions of the groups can be seen in Table 4.5. For the analysis, Table 4.6 shows the top 5 features for each model in terms of the gain-based importance, where the columns represent the order of the importance and the rows are for the models. The fields in Table 4.6 are colored in accordance to the colors for each feature group from Table 4.5.

Feature group	Description	Color
Static information	Features that do not change over time, such as site or product identifiers	Yellow
Time features	Calendar-based features like week, month, year, or their cyclical encodings	Red
Lag features	Past values of the target variable, shifted by a fixed number of time steps	Green
Rolling statistics	Aggregated values (mean, std) computed over a moving window of previous time steps	Blue
Encoded aggregates	Statistical encodings (mean, std, max, min) computed per group (e.g., branch/plant or product level) over the training data	Purple

Table 4.5: Feature groups with descriptions and colors

	Top 1	Top 2	Top 3	Top 4	Top 5
LightGBM global	encoded mean for product level	encoded std for product level	encoded max for product level	encoded std for branch/plant & product level	total units lagging by 6 months
XGBoost global	encoded max for product level	encoded mean for branch/plant & product level	product level	3 months rolling mean for branch/plant	encoded min for product level
LightGBM cluster-based	encoded std for product level	encoded std for branch/plant & product level	3 months rolling mean for branch/plant & product level	total units lagging by 6 months	7 months rolling std for branch/plant & product level
XGBoost cluster-based	encoded mean for product level	product level	encoded min for product level	3 months rolling mean for branch/plant & product level	encoded std for product level
LightGBM local Product 1	total units lagging by 7 months	15 months rolling std for branch/plant & product level	total units lagging by 11 months	cos of week	7 months rolling mean for branch/plant & product level
LightGBM local Product 2	7 months rolling std for branch/plant & product level	total units lagging by 13 months	cos of week	10 months rolling std branch/plant & product level	total units lagging by 11 months
LightGBM local Product 3	15 months rolling mean for branch/plant & product level	15 months rolling std for branch/plant & product level	month	year	week
LightGBM local Product 4	7 months rolling mean for branch/plant & product level	week	cos of month	total units lagging by 9 months	year
XGBoost local Product 1	total units lagging by 7 months	total units lagging by 8 months	week	7 months rolling mean for branch/plant & product level	15 months rolling std for branch/plant & product level
XGBoost local Product 2	10 months rolling std for branch/plant & product level	total units lagging by 13 months	10 months rolling mean for branch/plant & product level	15 months rolling std for branch/plant & product level	15 months rolling mean for branch/plant & product level
XGBoost local Product 3	15 months rolling mean for branch/plant & product level	10 months rolling mean for branch/plant & product level	7 months rolling mean for branch/plant & product level	15 months rolling std for branch/plant & product level	cos of month
XGBoost local Product 4	15 months rolling std for branch/plant & product level	cos of month	15 months rolling mean for branch/plant & product level	week	10 months rolling mean for branch/plant & product level

Table 4.6: Top 5 feature for the models according to gain-based importance scores

5

Discussion

The results of this study provide evidence that model configuration, product hierarchy level, and the incorporation of temporal features impact forecasting performance across a diverse set of products. Through both quantitative error metrics and visual inspection of forecast plots, distinct patterns emerged regarding model accuracy and behavior. The performance varied across products, reflecting the interaction between model complexity and data characteristics.

5.1 Benchmark vs. ML-based Models

ARIMA is one of the models used by the company and therefore, was used as a benchmark forecasting model. However, the results of this study show that alternative approaches, particularly GBDT models such as LightGBM and XGBoost, offer substantial improvements. To quantify these gains, ARIMA's performance is compared to the best-performing model for each product, using RMSE as the basis for comparison. RMSE was chosen for its interpretability, as it expresses forecast errors in the same units as the predicted variable, making it more actionable in a business context.

5.1.1 Product 1

For Product 1, LightGBM with a local modeling approach demonstrated the best performance across all evaluation metrics, indicating its ability to capture local patterns in the data. This was also reflected in the prediction plot (Figure 4.1), where the model was able to follow the actual trend more closely than the other models and accurately captured the volume of a sharp increase in demand during the test period.

The global models for XGBoost and LightGBM achieved the same RMSE and MAE scores as the ARIMA model, but XGBoost performed slightly worse in terms of MAPE. Interestingly, both ARIMA and LightGBM produced identical predictions for the test period, maintaining a constant forecast throughout. In contrast, XGBoost introduced some variation in its predictions, which at times brought it closer to the actual values and other times further away. This may explain the higher MAPE, as the percentage error is more sensitive to deviations when actual values are low. The similarity in RMSE and MAE suggests that all three models had

comparable average error magnitudes, but XGBoost's fluctuations likely introduced more relative error, which MAPE captures. These differences may arise from how each model handles global patterns, with LightGBM and ARIMA producing stable estimates, whereas XGBoost attempted to adapt to variations in the data.

The cluster-based and local XGBoost variants exhibited higher errors. While they achieved the same MAE score, their RMSE values differed slightly, with the local model performing marginally worse. Despite similar overall error magnitudes, the two models differ noticeably in their predictions of the trend. The local model produced more variation in its predictions, attempting to follow the actual demand pattern, although not with great accuracy. In contrast, the cluster-based model predicted nearly constant values throughout the test period, with only one point showing any deviation. This suggests that the cluster-based model failed to capture the product-specific dynamics, while the local model attempted to adapt to the trend but lacked precision.

The cluster-based LightGBM model showed the weakest performance among all approaches. It failed to follow the actual demand trend during the test period, noticeably overpredicting in response to a minor change in demand and underpredicting during a small spike. This indicates that the model struggled to generalize effectively at the cluster level for this specific product. One possible reason is that grouping the product with others in the same cluster may have diluted important product-specific signals, leading to less accurate predictions.

5.1.2 Product 2

In the case of Product 2, the superiority of machine learning approaches over ARIMA became more pronounced. XGBoost using a local modeling strategy recorded the lowest error values, underscoring its strength in capturing complex, product-specific demand patterns. Although the local XGBoost model is ranked highest based on the metrics, its predictions look very similar to those of the other ML models in Figure 4.2. This suggests that its better performance might come from smaller improvements at specific points, such as avoiding large over- or under-predictions, rather than a fundamentally better ability to follow the overall trend compared to the other models.

Following the top-performing XGBoost local model, the three LightGBM models, that are cluster-based, global, and local, ranked next in terms of RMSE, in that order. While the cluster-based variant had the lowest RMSE among the LightGBM models, it also outperformed the global and local versions in terms of MAE and MAPE. On the other hand, the global and local LightGBM models performed equally well based on MAE and MAPE. This consistency across configurations suggests that LightGBM was able to produce relatively stable predictions regardless of the modeling approach. However, the slightly better performance of the cluster-based model may indicate that grouping products by shared characteristics helped this type of model capture some localized patterns better without overfitting.

The two remaining XGBoost models, global and cluster-based, performed worse than all LightGBM models, indicating possible limitations in their ability to generalize or adapt effectively to the characteristics of Product 2. In terms of RMSE, the global model performed slightly better than the cluster-based model, though both models had the same MAE score. Notably, the global XGBoost model had the highest MAPE score of all models, indicating that it struggled the most with proportional accuracy, especially on lower demand values. In contrast, the XGBoost cluster-based model, while worse in RMSE, matched the best-performing model in terms of MAPE. This outcome may suggest that the cluster-based model was able to better estimate relative changes in demand for this product, but its absolute errors were still significant.

Lastly, the ARIMA model showed the weakest performance among all models for Product 2. In Figure 4.2, its predictions appear flat and unresponsive to the changes in actual demand during the test period. This behavior aligns with its nature as a univariate, time-series model that does not leverage other variables. As a result, ARIMA is less capable of capturing shifts in demand.

5.1.3 Product 3

For Product 3, the cluster-based LightGBM model demonstrated the best performance across the MSE, RMSE, and MAE metrics, although it did not achieve the lowest MAPE score. In Figure 4.3, this model shows a better ability to follow the general trend of the actual values compared to the other models. However, while it captures the direction and timing of changes relatively well, it struggles with accurately predicting the magnitude, underestimating the size of the demand spike during the test period. This suggests that although the model can identify when demand is likely to increase or decrease, it may be less sensitive to extreme variations in volume.

The next best-performing models are the global LightGBM model and the ARIMA model, in that order based on RMSE. Despite their differences in RMSE, both models share the same MAE score, indicating similar average absolute errors. Interestingly, ARIMA achieved the lowest MAPE among all models, while the global LightGBM model also outperformed the cluster-based variant in terms of MAPE. This suggests that although ARIMA may struggle with reducing absolute prediction errors, it remains relatively accurate when errors are evaluated in proportion to actual values. In Figure 4.3, ARIMA predicts a flat, constant value throughout the test period, which helps explain its stable relative error performance. On the other hand, the global LightGBM model attempts to follow the fluctuations in demand but fails to align with the actual trend, potentially due to the challenge of learning diverse product behaviors within a single model.

All remaining models, including the XGBoost variants (cluster-based, global, and local) and the local LightGBM model, performed worse than the ARIMA benchmark in terms of RMSE. Among them, the XGBoost cluster-based model had the lowest RMSE, followed closely by the local LightGBM model, but they shared the same MAE score. The global and local XGBoost models followed, with the local variant

performing the worst across all metrics. When evaluating MAPE, the local LightGBM model achieved the best relative performance among this group, followed by the XGBoost global, cluster-based, and finally, the local model. Notably, all LightGBM models performed better in terms of MAPE and had equal or better MAE scores compared to the XGBoost models. In Figure 4.3, these models failed to accurately follow the true demand trend, although their predictions did vary across the test period. These results suggest that the models may have overfit or struggled to generalize due to less relevant training patterns for this particular product. Additionally, the lower performance of the local and XGBoost models could be due to challenges in capturing the demand signal, especially when handling irregular patterns or sudden demand spikes.

5.1.4 Product 4

For Product 4, the results indicate that the modeling approach, whether global, cluster-based, or local, plays a significant role in model performance. Among these, the global models outperformed the others in terms of RMSE, followed by the cluster-based models, and lastly, the local models. The best-performing model for Product 4 was the global LightGBM model, which achieved the lowest scores across all evaluation metrics. In Figure 4.4, this model remained relatively close to the actual values throughout the test period, although it did not fully capture the underlying trend. Its consistent performance suggests that the global approach leveraged shared patterns across multiple products, helping to generalize better to unseen data.

The second-best model in terms of RMSE was the global XGBoost model, which followed a similar prediction pattern as the global LightGBM model in Figure 4.4. However, it had worse MAE and MAPE scores compared to the cluster-based XGBoost model. The cluster-based XGBoost and LightGBM models ranked third and fourth, respectively. While the cluster-based XGBoost model produced smoother predictions, it still lagged behind the global models in terms of accuracy. The cluster-based LightGBM model struggled to capture both the trend and timing of changes in demand, which likely impacted its overall performance.

The local models for both LightGBM and XGBoost were among the worst-performing in this case. Both overshot the actual demand during the test period. The local LightGBM model produced a large spike that did not align with the true demand values, suggesting potential overfitting to noise in the training data. On the other hand, the local XGBoost model failed to adapt to variations in the data, showing limited volatility in both the training and test periods. This behavior indicates underfitting, likely caused by insufficient information in the local dataset for effective learning.

The ARIMA model showed the weakest performance in terms of RMSE and MAE. However, it achieved a better MAPE score than the local models. In Figure 4.4, ARIMA consistently underpredicted the demand throughout the test period. This may help explain its relatively better MAPE score, as MAPE tends to penalize overpredictions more heavily than underpredictions. These results highlight the limitations of relying only on MAPE for model evaluation.

5.2 Feature Importance Analysis

To gain deeper insight into the behavior and decision-making processes of the different models, a feature importance analysis was conducted, focusing on the top five most influential features identified for each modeling approach and their corresponding feature groups. For the local models, the most important features are mainly rolling statistics, lag, and time features (see in Table 4.6). The rolling statistical features for the mean and standard deviation appear frequently, suggesting that smoothing local fluctuations over different time windows is beneficial for capturing short-term trends in demand at the individual product level. The lag features indicate a strong reliance on autoregressive signals, where past demand contributes to future demand predictions. The time-based features play a critical role in capturing calendar-related seasonality and periodic trends in the data. When such features are identified as important by the model, it indicates that temporal patterns significantly influence the target variable.

In contrast, the global and cluster-based models assign the highest importance to encoded aggregates, followed by static information features. For example, features for standard deviation or maximum units based on product level and branch/plant capture how much demand varies or how high it can get for the combination. These kinds of features act as proxies for the underlying demand behavior across different products. Their influence in the global and cluster-based models highlights the importance of capturing variation between the products, especially when one model is used to make predictions for many different products. Additionally, some rolling statistics and lag features emerge among the top 5 features, typically in the third to fifth position. This suggests that while temporal dynamics play a meaningful role, they are generally secondary to the aggregated and static characteristics in driving model performance for the global and cluster-based models.

5.3 Implications

While often considered a strong baseline for time series forecasting, ARIMA generally ranked in the middle or bottom among the evaluated models for the four products, often placing around third or last out of seven. The lack of flexibility for the ARIMA model in capturing nonlinear patterns may explain its inconsistent performance relative to gradient boosting methods.

When examining the best-performing models in relation to the volume group of each product, a clear pattern emerges. Products with low-volume demand (Product 1 and Product 2) achieved the best results using local models, suggesting that individualized modeling helps capture the unique patterns of such products more effectively. For medium-volume demand (Product 3), the cluster-based model yielded the best performance, indicating that grouping similar products can strike a balance between individual variation and shared demand structure. In contrast, for high-volume demand (Product 4), global models performed best, likely due to their ability to generalize well and benefit from larger amounts of data to capture broader

patterns.

This distinction in model performance is also reflected in the feature importance analysis. For low-volume products, features like rolling statistics, lag variables, and time-based features were most influential, likely because these features help detect short-term fluctuations and seasonality, which are more pronounced in low-volume series. On the other hand, for medium- and high-volume products, features such as encoded aggregate features and static information played a more significant role, likely because these products benefit from consistent trends and richer contextual information captured at the branch/plant and product level.

These findings suggest that no single modeling approach performs best across all products. Instead, the choice of the model approach should be tailored to the specific characteristics of the product, particularly its demand volume and data distribution. Products with different demand volume benefit from different modeling approaches and rely on distinct types of features, indicating that both model structure and feature engineering need to be adapted accordingly.

6

Conclusion

In summary, this study demonstrates that machine learning techniques can enhance forecasting accuracy compared to traditional models (Section 5.1). Features influencing demand results were identified (Section 5.2). Additionally, the evaluation revealed that global models offer scalable and flexible forecasting capabilities, although their performance varies depending on product characteristics compared to local and cluster-based models (Sections 5.1 and 5.3).

This study evaluated the forecasting performance of various statistical and machine learning models, specifically ARIMA, LightGBM, and XGBoost, across four different products using multiple modeling approaches, including global, local, and cluster-based. The results demonstrate that the effectiveness of each method is highly dependent on the specific characteristics of the product being forecasted.

In general, machine learning models outperformed ARIMA, especially in terms of RMSE and MAE, highlighting their ability to capture complex, non-linear patterns in the data. However, no single modeling approach was universally optimal. Local models performed best for Products 1 and 2, where individualized patterns likely dominated. For Product 3, cluster-based LightGBM offered superior performance, suggesting benefits from grouping similar time series. Product 4, in contrast, favored a global LightGBM model. As shown in Figure 4.4, this model effectively captures the two-peak pattern in the data and preserves the overall dynamics within consistent upper and lower bounds. Therefore, LightGBM consistently outperformed XGBoost across most products, highlighting its strength in managing diverse patterns in the data.

These findings underscore the importance of adopting a flexible, data-driven modeling strategy tailored to the characteristics of each forecasting scenario. Local models tend to perform better when forecasting products with low demand, as they can more precisely capture product-specific patterns. In contrast, global models often perform better for products with higher demand, offering the added benefits of reduced model maintenance and better scalability. Moreover, while cluster-based approaches can offer performance gains, their success is contingent on appropriate cluster formation. Ultimately, careful evaluation of error metrics, both absolute and relative, should guide model selection, especially when dealing with products exhibiting high variability or sparse data.

6.1 Future Work

This work lays the foundation for a more advanced forecasting pipeline and provides empirical support for replacing the current traditional methods with modern machine learning approaches. Future research may explore integration with downstream planning systems to further elevate forecasting performance.

While this study primarily focused on time-based patterns, approach type and model architecture, there is significant potential to improve forecast accuracy through the integration of external and contextual features. Future research should explore the inclusion of additional variables that are known to influence demand, such as:

- **Product prices:** Changes in price often affect customer purchase behavior and should be modeled to account for price elasticity.
- **Discounts and promotions:** Temporary sales campaigns and discounts can cause demand spikes that time-based models alone cannot anticipate.
- **Customer data:** Features such as customer segment, loyalty status, or regional purchasing trends can provide deeper insight into demand drivers.

Incorporating these features would enable the models to move beyond pure time series forecasting and toward context-aware predictive modeling, potentially improving responsiveness and precision even further. This would support more accurate planning in environments affected by marketing activities, competitive pricing, or changing customer behavior, and would bring the forecasting system closer to real-world complexity and decision-making needs.

6.2 Limitations

While this study demonstrates improvements in forecast accuracy using machine learning, several limitations should be acknowledged:

- **Sensitivity to Economic Shocks:** Although the models were trained on historical data, including periods of volatility such as the COVID-19 pandemic, their performance in future periods of economic disruption (e.g., recessions, supply chain crises, or major policy shifts) remains uncertain. Large, unforeseen changes in economic conditions could significantly reduce model reliability.
- **Limited Feature Scope:** The models were trained using only features derived from historical sales data. Excluding external variables like pricing, promotions, and customer characteristics limits the models ability to capture real-world demand drivers.
- **Narrow Product and Market Coverage:** The evaluation was conducted on a limited set of products and markets. Results may vary when applied to a broader range of products, especially those with highly irregular or sparse demand patterns.

- **Cold Start for New Products:** The models rely heavily on historical sales data. For new or recently launched products with little or no sales history, the models may not perform well, limiting their usefulness in product introduction or lifecycle planning.

Bibliography

[1] M. Seyedan and F. Mafakheri, "Predictive big data analytics for supply chain demand forecasting: Methods, applications, and research opportunities," *Journal of Big Data*, vol. 7, p. 53, 2020. [Online]. Available: https://doi.org/10.1186/s40537-020-00329-2.

[2] F. Hosseinnia Shavaki and A. Ebrahimi Ghahnavieh, "Applications of deep learning into supply chain management: A systematic literature review and a framework for future research," *Artificial Intelligence Review*, vol. 56, pp. 4447–4489, 2023. [Online]. Available: https://doi.org/10.1007/s10462-022-10289-z.

[3] X. Song and Z. S. Chen, "Enhancing financial time series forecasting in the shipping market: A hybrid approach with light gradient boosting machine," *Engineering Applications of Artificial Intelligence*, 2023, https://www.sciencedirect.com/science/article/pii/S095219762401100X.

[4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings and conclusions," *International Journal of Forecasting*, vol. 38, no. 4, pp. 1356–1376, 2022.

[5] P. J. Brockwell, "Time series," in *International Encyclopedia of Statistical Science*, M. Lovri, Ed., Berlin, Heidelberg: Springer, 2011, pp. 1583–1585. DOI: 10.1007/978-3-642-04898-2_595. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_595.

[6] N. Vandeput, *Data Science for Supply Chain Forecasting*, 2nd. De Gruyter, 2021, ISBN: 9783110671124. DOI: 10.1515/9783110671124.

[7] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*, 4th ed. Springer, 2017. [Online]. Available: https://link.springer.com/book/10.1007/978-3-031-70584-7.

[8] C. Chatfield, *Time-Series Forecasting*, 1st ed. Chapman and Hall/CRC, 2000, ISBN: 9781584880639.

[9] J. B. Rice and J. B. Frantz, "Supply chain big data series part 1: Understanding the value of big data," MIT Center for Transportation & Logistics, Tech. Rep., 2015.

[10] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/.

[11] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[12] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002, Accessed: 2025-05-08. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167947301000652.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *arXiv preprint arXiv:1603.02754*, 2016. [Online]. Available: https://arxiv.org/abs/1603.02754.

[14] XGBoost Developers, *XGBoost Python Package Documentation (v1.0.0)*, https://xgboost.readthedocs.io/en/release_1.0.0/python/index.html, 2020.

[15] G. Ke, Q. Meng, T. Finley, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[16] LightGBM Developers, *LightGBM Python Package Documentation*, https://lightgbm.readthedocs.io/en/latest/Python-Intro.html, 2024.

[17] R. L. Kashyap and S. B. Jain, "A survey of predictive modeling for time series forecasting," *Journal of Forecasting*, vol. 22, no. 2, pp. 87–106, 2003.

[18] A. Godbole, W. Krichene, and A. Deoras, "A comprehensive survey of regression based loss functions for time series forecasting," *arXiv preprint arXiv:2211.02989*, 2022. [Online]. Available: https://arxiv.org/abs/2211.02989.

[19] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *International Journal of Forecasting*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207021001874?via%3Dihub.

[20] V. Das, T. Mao, Z. Geng, C. Flores, D. Pelloso, and F. Wang, "Enhancing sell-in and sell-out forecasting using ensemble machine learning methods," *Amazon Science*, 2023. [Online]. Available: https://assets.amazon.science/75/de/b0f1b7444460b3761a0d7fb4753a/enhancing-sell-in-and-sell-out-forecasting-using-ensemble-machine-learning-method.pdf.

[21] D. Eki and T. Kaya, "Retail demand forecasting using temporal fusion transformer," in *Intelligent and Fuzzy Systems - Intelligent Industrial Informatics and Efficient Networks: Proceedings of the INFUS 2024 Conference*, C. Kahraman, S. Cevik Onar, S. Cebi, B. Oztaysi, I. Ucal Sari, and A. C. Tolga, Eds., ser. Lecture Notes in Networks and Systems, vol. 1090, Springer, 2024, pp. 165–170. DOI: 10.1007/978-3-031-67192-0_21. [Online]. Available: https://doi.org/10.1007/978-3-031-67192-0_21.

[22] P. Montero-Manso and R. J. Hyndman, "Principles and algorithms for forecasting groups of time series: Locality and globality," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1632–1651, 2021. DOI: 10.1016/j.ijforecast.2021.03.004. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2021.03.004.

[23] R. Pathak, R. Sen, N. Rao, N. B. Erichson, M. I. Jordan, and I. S. Dhillon, "Cluster-and-conquer: A framework for time-series forecasting," *arXiv preprint arXiv:2110.14011*, 2021. [Online]. Available: https://arxiv.org/abs/2110.14011.

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

A

Appendix 1

Table A.1: Description of hyperparameters used for XGBoost and LightGBM models

Hyperparameter	**Model(s)**	**Description**
`tweedie_variance_power`	Both	Controls the variance of the Tweedie distribution.
`learning_rate`	Both	Shrinks the contribution of each tree, which helps with model generalization (also called `eta`).
`n_estimators`	Both	Number of boosting rounds or trees to be built.
`subsample`	Both	Fraction of training instances randomly sampled for each tree (controls overfitting).
`colsample_bytree`	Both	Fraction of features randomly sampled for each tree.
`lambda` (`reg_lambda`)	Both	L2 regularization term on weights, which helps prevent overfitting.
`alpha` (`reg_alpha`)	Both	L1 regularization term on weights, which promotes sparsity in the model.
`max_depth`	XGBoost	Maximum tree depth, which limits model complexity and controls overfitting.
`gamma`	XGBoost	Minimum loss reduction required to make a split (pruning control).
`num_leaves`	LightGBM	Maximum number of leaves per tree, which is the main parameter for controlling complexity in LightGBM.
`min_child_samples`	LightGBM	Minimum number of samples required to create a new leaf node, which prevents overfitting.