UNIVERSITA' DEGLI STUDI DI PADOVA

DIPARTIMENTO DI SCIENZE ECONOMICHE ED AZIENDALI "M. FANNO"

Master Thesis in Business Administration

COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS FOR TIME SERIES SALES FORECASTING IN SUPPLY CHAIN MANAGEMENT

Supervisor

Prof. Enrico Rettore
University of Padova

Master Candidate

Ghazal Beignezhad
2088672

ACADEMIC YEAR

2024 - 2025

Il candidato dichiara che il presente lavoro è originale e non è già stato sottoposto, in tutto o in parte, per il conseguimento di un titolo accademico in altre Università italiane o straniere.

Il candidato dichiara altresì che tutti i materiali utilizzati durante la preparazione dell'elaborato sono stati indicati nel testo e nella sezione "Riferimenti bibliografici" e che le eventuali citazioni testuali sono individuabili attraverso l'esplicito richiamo alla pubblicazione originale.

The candidate declares that the present work is original and has not already been submitted, totally or in part, for the purposes of attaining an academic degree in other Italian or foreign universities. The candidate also declares that all the materials used during the preparation of the thesis have been explicitly indicated in the text and in the section "Bibliographical references" and that any textual citations can be identified through an explicit reference to the original publication.

Firma dello studente

“To the brave women of Iran—

who continue to fight for their basic human rights every day yet still shine brilliantly and break the chains that are made to stop them. Your resilience and courage are an inspiration. “Women, Life, Freedom”

To my country—

Now caught in the crossfire of the Islamic Republic and Israel, may the day come when we are no longer silenced or subdued. When we are free to live as our true selves, to choose our paths without coercion, to raise our voices without fear. May peace replace oppression, and dignity return to every home. Until then, we remember, we resist, and we hope.

To my parents—

who stayed behind so I could move forward, who sacrificed not just time but presence, watching from afar as I chased my dreams. Your unwavering love, quiet strength, and boundless support have been the foundation beneath my every step. This achievement is as much yours as it is mine.

To Helia, Fatemeh and Bahar—my chosen family, my relentless cheerleaders.

For every time you lifted me up when I felt like falling, for the patience with which you endured my exhaustion and frustration—I am endlessly grateful. You stayed up with me on deadline nights, reminding me that I was not alone. You encouraged me when I doubted myself, believed in me when I struggled to, and celebrated even my smallest victories. You carried me through this, in ways both big and small, and I will never forget it.

With all my love and gratitude,

This work is dedicated to you.”

Ghazal

Abstract

Accurate sales forecasting in supply chain management remains is a central challenge that directly impacts inventory control, logistics efficiency, and strategic planning. In order to assess how well classic statistical, hybrid, and machine learning models predict actual, seasonal sales data from a large retail dataset, this thesis compares them using features like lagged sales, price, discount, and time-based indicators in a panel of ninety-six products observed over thirty three months.

Four forecasting models were considered: Pooled Ordinary Least Squares (OLS), Prophet (a modular trend-seasonality-holiday model), Random Forest, and XGBoost. All models were trained and evaluated under a unified preprocessing and validation framework, using standardized metrics including RMSE, MAE, MAPE, and R^2. Random Forest consistently outperformed all other models, achieving the highest accuracy and lowest error rates, followed by XGBoost. Prophet demonstrated competitive performance in capturing seasonality but underperformed during promotion-driven volatility. Pooled OLS, while the least accurate, provided interpretable coefficients and served as a robust linear benchmark.

The findings suggest that tree-based machine learning models are highly effective for forecasting in retail settings characterized by seasonality and promotional effects, provided that structured features and cross-sectional variance are properly modelled. The study also highlights key trade-offs between model complexity and interpretability, emphasizing the importance of aligning forecasting tools with operational context. Practical implications for model selection and deployment are discussed, along with limitations and recommendations for future research involving deep learning architectures, hierarchical forecasting, and explainable AI methods.

Table of Contents

List of Figures and Tables

1

Chapter 1 Introduction

1.1 Background

1.2 Problem Statement

1.3 Research Objectives

1.4 Research Questions

1.5 Scope of The Study

1.1 Background

In today's economy worldwide, effective supply chain management (SCM) plays a very important role in making operations efficient and competitive advantage. In the core of SCM is the task of demand forecasting, which serves as a foundation of strategic decisions such as capacity planning, inventory control, logistics, and marketing (Chopra, 2019, pp. 204–205). Accurate forecasts reduce uncertainty, align supply with demand, and mitigate risks associated with under- or over-stocking.

However, because of fluctuating markets, shortened product lifecycles, and frequent global disruptions (Syntetos et al., 2016) SCM forecasting has become more difficult. Inaccurate forecasts can spread upstream in the supply chain, causing the well-documented bullwhip effect—amplifying demand variability, inflating costs, and reducing responsiveness (Lee, Padmanabhan, & Whang, 1997, as cited in Perera et al., 2018; Feizabadi, 2022).

Traditional time-series models like ARIMA and Exponential Smoothing have historically led the forecasting landscape (Chatfield, 2001). These models, while they are interpretable and robust when stationary conditions are met, tend to often struggle with the non-linear, multivariate, and noisy patterns which is common in real-world sales data (Makridakis et al., 2018)

The big data era has introduced vast, high-frequency datasets from point-of-sale systems, sensors, and online platforms—enabling richer consumer insights (Boone et al., 2018; Paper 3). This caused growing interest in Machine Learning (ML) and Deep Learning (DL) for demand forecasting. These methods, particularly Random Forest, Gradient Boosting, and LSTM networks, offer the potential to model complex interactions and capture non-linear temporal dependencies (Seyedan & Mafakheri, 2020; Paper 20; Douaioui et al., 2024; Paper 19; Lim & Zohren, 2021; Paper 29).

However, despite these models being promising, empirical evidence regarding the superiority of them remains mixed. Benchmarking studies (e.g., Makridakis et al., 2018; Paper 22) show that traditional statistical models specially for short and noisy business time series often perform better than ML and DL (Shih & Rajendran, 2019; Paper 17). This indicates a gap between theoretical potential and practical usage, leaving practitioners uncertain about which forecasting approach to adopt.

1.2 Problem Statement

Although Machine Learning models are increasingly proposed as superior forecasting tools, comprehensive empirical studies comparing machine learning models utilizing real-world, supply chain-relevant data are still lacking. Moreover, model performance is dependent on context and setting, like data frequency, noise, trend behavior, and seasonality. This leads to practitioners lacking evidence-based and clear guidance on how to select models.

This thesis addresses this critical gap by comparing classical, machine learning, and specialized forecasting models on a real-world, seasonal sales dataset. The study provides data-driven insights into relative performance, trade-offs, and model suitability for operational sales forecasting.

1.3 Research Objectives

This research pursues the following objectives:

1. To implement and evaluate a set of forecasting models—including Pooled OLS (baseline), Prophet (modern statistical), and Random Forest and XGBoost (machine learning).
2. To compare the models using standard metrics (e.g., RMSE, MAE, R^2, MAPE) on a structured, real-world panel dataset of monthly sales.
3. To identify performance differences across different models and offer practical recommendations for model selection in similar forecasting tasks.

1.4 Research Questions

This thesis is guided by the following research questions:

1. How do ML models (Random Forest, XGBoost) compare to each other in forecasting sales?
2. How do ML models (Random Forest, XGBoost) compare to traditional and modern statistical models (Pooled OLS, Prophet) in forecasting seasonal sales data?
3. What performance hierarchy emerges among the models, and what are the relative strengths and weaknesses of each?
4. What trade-offs exist between model complexity, computational cost, and predictive accuracy in real-world forecasting scenarios?
5. Based on the empirical results, which model(s) are most suitable for operational forecasting in retail supply chains?

1.5 Scope of the Study

This study focuses on monthly time-series forecasting of product-level sales using a structured panel dataset.

The analysis compares forecasting models based solely on predictive accuracy, using a fixed test set and standard error metrics. Broader topics such as causal inference, pricing strategy, or inventory optimization are excluded. The study aims to inform model selection for forecasting rather than broader supply chain policy.

1.6 Thesis Structure

This thesis is organized into five chapters:

- Chapter 1: Introduction provides the background and motivation for the research, defines the problem statement, outlines the objectives and research questions, and details the scope and structure of the study.
- Chapter 2: Literature Review presents a critical review of the relevant academic literature, tracing the evolution of forecasting methods and synthesizing key themes, debates, and findings in the application of statistical and machine learning models to supply chain forecasting.
- Chapter 3: Methodology describes the research design in detail. This includes a description of the dataset, the data preprocessing and feature engineering steps, the technical implementation of each forecasting model, and the evaluation protocol used to ensure a fair and robust comparison.
- Chapter 4: Results and Analysis presents the empirical findings of the comparative study. It reports the performance metrics for all models, analyses the differences in their predictive accuracy, and discusses the implications of these results.
- Chapter 5: Discussion summarizes the key findings of the thesis, provides direct answers to the research questions, and discusses the theoretical and practical implications of the work. The chapter concludes by acknowledging the study's limitations and suggesting promising avenues for future research.

2

Chapter 2 Literature Review

2.1 Introduction

In this chapter, the scholarly literature on sales forecasting in the supply chain management setting is reviewed, with a focus on the comparative studies of machine learning forecasting models. In recent years, enterprise systems, point-of-sale networks, and online retail platforms have increased the available sales and transactional data on a broader level. This availability has created new opportunities for predictive modeling using data-driven approaches. Within this environment, businesses are beginning to use machine learning methods such as Random Forest, XGBoost, and specialized tools like Prophet to forecast demand at the product level more accurately.

Many SCM components, like procurement planning, inventory control, capacity management, and promotional strategy, are fundamentally influenced by sales forecasting (Chopra, 2019). However, constantly shifting demand patterns, erratic promotional events, product-specific seasonality, and data sparsity—especially in multi-product retail environments—make real-world forecasting difficult. (Syntetos et al., 2016; Boone et al., 2018). These complexities have led researchers to seek flexible and scalable forecasting solutions that can generalize across SKUs[1] while maintaining high levels of predictive accuracy.

Tree-based machine learning models are well-suited to overseeing the complex, nonlinear patterns that characterize SKU-level sales data. Both Random Forest and XGBoost have been employed successfully in supply chain forecasting tasks due to their robustness to overfitting, tolerance for missing data, and ability to model feature interactions—attributes that make them particularly valuable in high-dimensional retail datasets (Douaioui et al., 2024; Feizabadi, 2022; Panda & Mohanty, 2023). Prophet, originally introduced as a modular forecasting framework by Facebook, has also been widely adopted, especially in business contexts where domain knowledge (e.g., known holidays and events) can be integrated into the model structure (Ensafi et al., 2022; Raiyani et al., 2021).

However, there is still a gap in the literature: Although these models have been used widely, very few research have compared and contrasted them using the same preprocessing, panel configurations, and validation settings. It is challenging to draw insightful conclusions about machine learning methods' various benefits and drawbacks since most of the current literature either tests each model separately or compares them to oversimplified statistical standards. Additionally, very few forecasting studies use panel datasets, which are collections of product

[1] A **Stock Keeping Unit (SKU)** is a unique identifier assigned to a specific product variant in a retailer's inventory system, typically defined by attributes such as brand, model, size, and color. In this thesis, SKU-level data refers to monthly sales, pricing, and promotional information tracked separately for each product variant, enabling granular forecasting and inventory planning.

time series that are simultaneously modeled using engineered features and constant time intervals (Baltagi, 2014). This is even though panel structures closely mirror the operational realities of modern retail data.

In this chapter, the literature is reviewed through the lens of the central research objective: to conduct a rigorous, data-driven comparison of Random Forest, XGBoost, and Prophet, evaluated against a statistical baseline (Pooled Ordinary Least Squares) for forecasting product-level sales monthly. The review draws from 32 articles, encompassing research in supply chain forecasting, machine learning methodology, panel data analysis, and evaluation frameworks.

The chapter proceeds with five aims:

- To define the practical role of forecasting in supply chain operations and describe the structural challenges of multi-SKU sales data.
- To synthesize academic findings on the application of Random Forest, XGBoost, and Prophet in sales forecasting.
- To outline best practices in structuring, transforming, and engineering features within panel-based machine learning pipelines.
- To examine the error metrics and evaluation strategies used for comparative model assessment in the forecasting literature.
- And to identify methodological gaps that justify the comparative empirical focus of this study.

What follows is a detailed synthesis of these domains to contextualize the thesis's empirical framework and contribution to the field.

2.2 Supply Chain Management Forecasting

Forecasting is a foundational element of supply chain management, supporting strategic and operational decisions across procurement, production, logistics, inventory control, and customer service. Its role has only grown in importance with the increasing complexity of global supply networks and rising customer expectations for accuracy and responsiveness.

2.2.1 The Strategic Role of Forecasting in Supply Chains

Forecasting, as an essential part of supply chain management, plays a significant role in both strategic and operational levels. Whether it is determining how much raw material to procure, scheduling production, managing inventory, or planning distribution routes, the decisions across these domains are grounded in some form of demand forecast. When forecasts are

credible, firms may better match their operations with customer needs, avoid costly excess inventory, and lower the likelihood of stockouts—all of which lead to improved service performance and more effective capital usage (Chopra, 2019). Furthermore, beyond short-term operational decisions, forecasting is important in longer-term tasks and agendas like the design of networks, supplier agreements, and marketing campaign planning. In frameworks like Sales and Operations Planning (S&OP) or Vendor Managed Inventory (VMI), forecasts are not used solely but as shared tools for coordination. As noted by Perera et al. (2018), these forecasts serve as communication tools that synchronize actions between internal teams and external partners, allowing for cooperative decision-making across functional and organizational boundaries.

In retail, since product portfolios are diverse and updated frequently, forecasting cannot be based solely on aggregate observations. Instead, item-level forecasts are needed to design inventory strategies for each stock-keeping unit (SKU). When these forecasts are accurate, businesses can better match customer expectations while avoiding the inefficiencies of blanket safety stock rules. This highlights the need for forecasting models that can generalize well across a diverse product mix without sacrificing SKU-specific accuracy.

2.2.2 Forecasting Challenges in Retail Supply Chains

While demand forecasting is essential, it is not easy to execute in modern retail settings. According to Syntetos et al. (2016), retail environment creates a perfect storm for forecasting difficulties: the life cycles of many products are short, some others witness irregular or intermittent demand, while promotion usually distorts the underlying trends. complexity is the fact that historical data for many items is sparse, either because the product is new or because it sells infrequently.

Retailers today also operate in data-rich environments, but the large availability of data does not necessarily mean a better forecasting. As Boone et al. (2018) point out, while firms now have access to a wide range of behavioural and transactional data—from in-store scanners to online browsing logs—these sources are often noisy and may reflect brief and short-lived patterns.

If a model assumes some kind of stationarity or consistent behavior, then it will perhaps prove incompatible with an unstable, high-dimensional input.

One complexity in retail forecasting lies in promotions. Temporary price reductions, bundled offers, and marketing campaigns tend to generate rapid demand spikes, which are not typical seasonal patterns. If these dynamics are not taken into account by the forecasting models, they

will systematically underestimate or overestimate demand, thus disrupting operations. Some tools allow known events to be manually plugged into the model structure (e.g., holidays in Prophet). Some others, like Random Forest, require that effects be modeled indirectly via careful feature engineering, such as the use of time lags, promotion flags, or cyclical encodings.

2.2.3 Core Challenges in Supply Chain Forecasting

Despite its importance, forecasting in supply chains faces persistent and multidimensional challenges. These include the presence of seasonality and promotional distortions, demand volatility and shocks, and data sparsity combined with granularity mismatches—each of which imposes unique demands on the forecasting models used.

A. Seasonality and Promotion Effects

Seasonality in retail sales is widely accepted-going with the cyclical occurrence of sales-so it is linked to weather, holidays, school calendars, and cultural events. Models such as Holt-Winters and SARIMA consider these seasonal components as additive or multiplicative. These models, however, presuppose a seasonal environment that is fixed and regular; really, irregular demand explosions are occasioned in the real world, especially because of promotions.

Promotions entail minor price reductions or some special offer or marketing campaign and introduce sudden changes from the normal sales behaviour. These are structured deviations which typical statistical models cannot capture unless they are explicitly modeled. Boone et al. (2018), on the other hand, propose that promotional effects may be modeled using data such as social media sentiment, Google Trends, and clickstream data, thereby causing a curse of dimensionality problem.

PromoCast (Trusov et al., 2006) had tackled this problem by combining regression models and rule-based modules to address systematic promotional biases. Effective as these are, such systems are hard to configure, requiring significant amounts of domain knowledge and assistance. Modern ML models such as XGBoost, on the other hand, are able to pick up interaction effects between promotion, price and seasonality implicitly, without explicit rules, although typically at the cost of interpretability.

B. Demand Shocks and Regime Shifts

Forecasting turns out to be very unreliable when there are structural breaks, which can be also referred to as regime shifts. These breaks are caused by unexpected demand shocks that come from pandemics, viral social trends, or sudden supply constraints. Such statistical models as ARIMA and ETS are by nature backward-looking, and they are lagging in adjusting to changes

in data. According to Makridakis et al. (2018), even the most finely calibrated statistical models lose their effectiveness very fast if the data distribution changes. Fuzzy Time Series models have been suggested to be more resilient against this type of volatility. Chan et al. (2015) showed that second-order fuzzy models performed better than ARIMA and Grey Prediction in a supply chain simulation when capturing sudden shifts, majorly at upstream nodes, like manufacturers and distributors. However, fuzzy models are difficult to scale and to standardize; this has limited their usage within commercial forecasting systems.

C. Data Sparsity and Granularity Mismatches

Supply chain data is often abundant in total volume but sparse at the SKU-month level, especially when modeling disaggregated forecasts. This results in short, noisy, and intermittently populated time series. Syntetos et al. (2016) classify this as a fundamental obstacle to forecast accuracy and argue for tailored methods like Croston's method or intermittent adaptations of exponential smoothing.

Another challenge is mismatched granularity between dependent and independent variables. For example, while sales data may be available daily, pricing or promotional flags may only update weekly. Ensafi et al. (2022) chose to aggregate to monthly frequency to mitigate this mismatch, accepting some loss of short-term sensitivity for improved signal-to-noise ratio.

2.3 Traditional Forecasting Methods

The foundation of supply chain predictive analytics is made of conventional time series forecasting techniques. These techniques, which are mostly based in statistical theory, provide clear frameworks, strong interpretability, and shown results in certain conditions, especially when the data is univariate, stationary, and seasonally regular. Three fundamental groups of statistical models—ARIMA models, exponential smoothing techniques, and regression-based models—as well as their use with panel data are thoroughly reviewed in this section. Additionally, it draws attention to the advantages and disadvantages of each when applied to operational forecasting.

2.3.1 ARIMA and Seasonal ARIMA Models

Autoregressive Integrated Moving Average (ARIMA) models are among the most established techniques for time-series forecasting. ARIMA models operate on the assumption that future values of a series can be expressed as a linear combination of past observations and past forecast errors. The general ARIMA(p, d, q) model includes:

- p: order of the autoregressive part (AR),

- d: degree of first differencing to achieve stationarity (integration),
- q: order of the moving average part (MA).

When seasonality is present, ARIMA is often extended to SARIMA models, which incorporate seasonal differencing and seasonal AR/MA terms to model periodic components.

(Makridakis et al., 2018), in a large-scale empirical comparison using the M3 competition data, found ARIMA to be among the most robust performers for short-horizon, monthly time series—often outperforming machine learning models like MLP and RNN.

However, ARIMA and its seasonal extensions require:

- Manual or Automated specification and tuning of parameters (p, d, q),
- A stationary series, which often requires differencing and transformation,
- Limited capacity to handle nonlinear interactions, multivariate features, or external regressors unless extended to ARIMAX models.

2.3.2 Exponential Smoothing and Holt-Winters Models

Exponential smoothing methods have great popularity in the commercial supply chain software because of their simplicity, speed, and flexibility. These models merely predict a future value as a weighted average of the past data where recent data are given exponentially more weight.

There are several forms:

• Simple Exponential Smoothing (SES): Best for series with no trend or seasonality.

• Holt's Linear Method: Adds a trend component.

• Holt-Winters (Additive or Multiplicative): Incorporates both trend and seasonal components.

In the forecasting study by (Shih and Rajendran 2019), the Holt-Winters multiplicative model achieved competitive MAE and MAPE scores compared to ARIMA and ANN in predicting daily blood supply. (Ensafi et al. 2022) also used double and triple exponential smoothing as baseline models in their 13-model comparison for furniture demand forecasting.

The benefit of exponential smoothing includes:

• computational efficiency and simplicity,

• Low data preprocessing,

• being able to model level, trend, and seasonality with using a small number of parameters.

However, here are smoothing methods:

• Univariate, they cannot use external regressors like price or promotion,

• limited in representing nonlinear effects or abrupt changes,

• Susceptible to performance degradation in data with irregular seasonality or noise.

2.3.3 Regression and Panel Data Models

While ARIMA and smoothing methods are purely time series models, regression-based forecasting introduces explanatory variables to explain variation in the target series. This is particularly relevant in supply chain contexts, where price, promotion, product characteristics, and seasonal flags are expected to influence demand.

The general linear regression model takes the form:

$$Y_t = \beta_0 + \beta_1 X_{1t} + \beta_2 X_{2t} + \cdots + \epsilon_t$$

Where Y_t is the sales variable, and $X_{1t}, X_{2t}, \ldots$ are covariates like discount, lagged sales, or holiday indicators.

The Baltagi (2014) advocates for panel data models when dealing with cross-sectional time series. Panel regression allows one to:

- Leverage both temporal and cross-sectional variation,
- Control for unobserved heterogeneity (Fixed/Random effects),
- Perform unit-specific forecasting.

However, regression models require:

- Careful preprocessing to handle multicollinearity,
- Differencing or transformation for stationarity,
- Diagnostic testing to ensure valid inference.

2.4 Machine Learning in Sales Forecasting

ML methods have become instrumental in sales forecasting in recent years, especially in retail and supply chain sectors that rely on structured, time-indexed transactional data. These models, which are able to learn directly from the data without demanding rigorous assumptions about the distribution or functions in the relationships between variables, are catching the eye of the research community.

Traditional statistical models that are usually constrained by their linear nature and dependence on stationary, univariate inputs contrast the ML algorithms, which are capable of dealing with nonlinearities, feature interactions and high-dimensional inputs such as pricing, discounting and temporal encodings. Because of their versatility, these algorithms are very suitable for the case of product-level sales in the real-life situations that are constantly changing, where the demand may be the result of a combination of the recent performance, the seasonality and the external business factors.

This subsection discusses three machine learning models that are the most popular among the demand forecasting practitioners: Random Forest, XGBoost, and Prophet. Although Prophet is repurposing some elements of statistical modeling, it has been accepted in the ML realm because of its modular character and the easiness with which it can take features and automated pipelines. Firstly, all three models were applied in prior studies on retail and supply chain forecasting, with differential results depending on data characteristics and implementation choices.

Here, we examine how these models have been utilized in the forecasting literature, highlight their architectural features, and evaluate their reported performance, interpretability, and scalability. The review also identifies practical considerations relevant to their application in multi-product, monthly panel data structures such as the one used in this thesis.

2.4.1 Tree-Based Models

Random Forest

Random Forest (RF) is an ensemble learning method that combines the predictions of multiple decision trees to improve generalization and reduce variance. It operates by building each tree on a bootstrap sample of the data and selecting random subsets of features at each split to encourage diversity among trees. The final forecast is usually the average prediction across all trees (in regression settings), which reduces overfitting compared to single-tree models.

In the domain of sales forecasting, RF has demonstrated robust performance across various applications. Douaioui et al. (2024) identified RF as one of the most frequently used ML models in retail demand forecasting, especially when handling high-dimensional data with multiple exogenous variables.

Strengths:

- Handles nonlinear relationships and interactions: RF vigor at deriving nonlinear complex patterns and interactions between variables without explicit specification, unlike linear models.
- Tolerant to outliers and missing values: RF is insensitive to outliers due to its tree-based design and is able to deal with missing values (e.g., through imputation or surrogate splits in some implementations).
- Interpretable variable importance scores: RF generates feature importance metrics (e.g., based on mean decrease in impurity or permutation importance), which clarify your understanding of the variables.

Limitations:

- Computationally intensive on large datasets: RF is time-consuming in training and forecasting of large datasets as a result of tree construction and switching at various places.
- Lacks input mechanisms for time series order: RF does not automatically consider time dependencies or autocorrelation in time series data unless some features such as lagged variables or time-based indicators are created explicitly.
- Difficulty with extrapolative trends: RF struggles to predict values outside the range of the training data (e.g., new peaks or trends) because it relies on averaging predictions from trees, which are bounded by the observed data.

XGBoost

XGBoost is a boosting algorithm that builds decision trees sequentially, with each new tree designed to correct the residual errors of the existing ensemble. It incorporates regularization techniques, such as L1 (Lasso) and L2 (Ridge) penalties, to prevent overfitting and improve generalization to out-of-sample data. Additionally, XGBoost uses advanced optimization methods, including second-order gradient approximations (Hessian-based updates), which enhance its computational efficiency and predictive accuracy compared to traditional gradient boosting. These features make XGBoost highly effective for a wide range of machine learning tasks.

Strengths of XGBoost:

- High predictive performance across domains,

- Handles missing data and sparse matrices natively,
- Incorporates regularization to reduce overfitting,
- Flexible and customizable.

Limitations:

- Less interpretable than Random Forest,
- May struggle with extreme volatility or very short series,
- Requires careful hyperparameter tuning to avoid overfitting or underfitting.

Overall, both Random Forest and XGBoost represent strong ML baselines for supply chain forecasting. The choice between them often hinges on trade-offs between interpretability, speed, and sensitivity to noise.

2.4.2 Hybrid Specialized Models: The Prophet Model

Prophet is a generalized additive model (GAM) that decomposes a time series into three interpretable components: trend, seasonality, and holidays. It models the series as:

$$Y_t = g_t + s_t + h_t + \epsilon_t$$

Where:

- g_t is the trend function (piecewise linear or logistic growth),
- s_t captures seasonality using Fourier series (daily, weekly, yearly),
- h_t accounts for user-specified holiday effects,
- ϵ_t is the error term.

Unlike ARIMA or exponential smoothing, Prophet:

- Requires no stationarity transformation,
- Automatically detects changepoints in trend,
- Incorporates known events (e.g., holidays, product launches),
- Allows for additive or multiplicative seasonality components.

These design features make Prophet particularly well-suited for business time series that are:

- Nonlinear with sharp changepoints,

- Strongly seasonal,
- Affected by known external events.

Ensafi et al. (2022) used Prophet in sales forecasting contexts and found that it significantly outperformed simpler models like exponential smoothing and SARIMA, especially when promotional or holiday effects were present. Prophet also requires minimal tuning and is accessible to non-experts, contributing to its widespread adoption in industry.

In (Ensafi et al., 2022), where Prophet achieved good overall accuracy (RMSE = 167.29) but underperformed compared to Stacked LSTM and CNN in capturing short-term volatility. Similarly, in the study by Raiyani et al. (2021), Prophet showed strong interpretability and handled holiday peaks well but missed erratic price-driven variations.

2.4.3 Prophet vs. Machine Learning Models

In sum, Prophet is particularly effective when domain expertise is available to define holidays or changepoints, when longer-term trends and periodicity dominate, and when interpretability is critical. However, for highly irregular, sparse, or promotion-heavy time series, ML models like XGBoost may outperform Prophet by capturing interaction effects and nonlinear dependencies not explicitly modeled in Prophet's structure.

Feature	**Prophet**	**ML Models**
Seasonality Modeling	Built-in (Fourier decomposition)	Learns from engineered time features
Holidays/Events	Manual input (domain-informed)	Implicit (if features are provided)
Interpretability	High (modular components)	Low (black-box variable splits)
Handling Promotions	Manual holiday inputs	Captures interactions automatically
Adaptability to Trend Shifts	Piecewise linear/logistic	Learns through lag and trees
Performance on Irregular Data	Moderate	High (if engineered correctly)
Forecast Uncertainty	Predictive intervals provided	No built-in intervals
Usability	Easy to deploy, minimal tuning	Requires hyperparameter optimization

Table 2.1: ML and Prophet Feature Comparison

2.4.4 Other Hybrid and Specialized Models

Beyond Prophet, several hybrid models have been proposed in the literature. These include:

- ES-RNN (Exponential Smoothing-Recurrent Neural Network), used in the M4 forecasting competition (Lim & Zohren, 2021), which combines exponential smoothing with a recurrent neural network in a unified architecture,
- PromoCast (Trusov et al., 2006), a semi-automated system combining regression forecasts with post-hoc rule corrections for promotion bias.

These models show strong results in niche applications, but are often complex to implement, resource-intensive, and difficult to scale across thousands of SKUs—making them more suitable for high-stakes, targeted forecasting (e.g., new product launches, pharmaceutical distribution).

In conclusion, Prophet and other hybrid models occupy an important space between traditional models and machine learning. While not always the top performer in terms of predictive accuracy, their strengths in transparency, flexibility, and usability make them essential tools in many real-world forecasting applications.

2.5 Panel Data and Time Series Forecasting

Forecasting sales at the product level often involves observing the same entities (e.g., SKUs) over time—yielding panel data, also known as longitudinal data. In contrast to the typical time series that deals with just one sequence or cross-sectional data that shows a snapshot of one point in time, panel data has both dimension of time and cross-section. The structure is especially ideal for supply chain environments which require forecasting to consider product-specific behavior over time as well as the series being heterogeneous, short and frequently irregular. This section reviews how panel data structures support effective forecasting, discusses key statistical models (Pooled OLS, Fixed and Random Effects), and describes the diagnostic tools used for appropriate model selection.

2.5.1 The Role of Panel Data in SCM Forecasting

In supply chain settings, forecasting for hundreds or thousands of SKUs is common, with each SKU having its own unique sales trajectory due to differences in demand patterns, seasonality, or promotions. Traditional univariate time series models like ARIMA or Holt-Winters (exponential smoothing) are typically applied independently to each SKU's historical data. This approach has notable drawbacks:

- Computational Inefficiency: Fitting a separate model for each SKU requires significant computational resources, especially when hyperparameter tuning (e.g., ARIMA's p, d, q orders or Holt-Winters' smoothing parameters) is involved. For thousands of SKUs, this process can be prohibitively slow and resource intensive.
- Failure for Sparse or Low-Volume SKUs: ARIMA and Holt-Winters rely on sufficient historical data to identify patterns like trends or seasonality. Sparse or low-volume SKUs, which may have intermittent demand (e.g., zero sales in many periods) or limited historical data, often lead to poor model fits, unstable parameter estimates, or unreliable forecasts.

Panel data approaches address this by leveraging cross-sectional information. For example, if a new product has only six months of sales history, a pooled or fixed-effects model can still make reasonable forecasts by borrowing strength from similar products in the dataset. This is crucial in retail chains, where product lifecycles are short, and SKU turnover is high (Baltagi, 2014).

Panel forecasting has also been used in healthcare, retail, and energy. Shih and Rajendran (2019) explored the implementation of regression for blood inventory management in different cities and emphasized the significance of capturing inter-series variability through modeling. On the other hand, Feizabadi (2022) also employed panel data and lagged macroeconomic indicators to make sales forecasts and operational KPIs more accurate in the steel manufacturing sector.

2.5.2 Panel Data Modeling Techniques

- Pooled Ordinary Least Squares (OLS)
  The most straightforward panel model is Pooled OLS, which takes it that all units (such as products) have the same intercept and slope coefficients. The model is solved by combining all observations over time and entities into one dataset and then conducting a standard OLS regression.
- Fixed Effects (FE) and Random Effects (RE) Models
  When product-specific traits (e.g., brand popularity, design) influence sales but are not represented by observable variables, a Fixed Effects model might be needed. FE model is based on the assumption that each unit has its own intercept, which is recovered by within-entity transformations. This process effectively eliminates time-invariant, unobserved heterogeneity. Alternatively, Random Effects models treat these product-specific effects as random variables drawn from a common distribution. They are more

efficient than FE if the assumption of no correlation between unit effects and regressors holds.

2.5.3 Advantages of Panel-Based Forecasting

Compared to traditional univariate forecasting, panel models offer several advantages in supply chain contexts. However, panel models still rely on strong assumptions about stationarity, linearity, and independence of errors—which limits their ability to capture nonlinear dynamics unless supplemented by feature engineering or hybridization.

Advantage	Explanation
Cross-sectional strength	Information from similar products improves forecasting for new or sparse SKUs.
Richer feature integration	Supports multivariate inputs like price, discount, seasonality.
Scalability	One model can handle hundreds of products simultaneously.
Interpretability	Coefficients offer causal insights and diagnostic clarity.

Table 2.2: Panel Based Forecast Advantages

2.6 Evaluation Metrics in Forecasting Studies

Evaluating the performance of forecasting models requires the use of appropriate error metrics that quantify the deviation between predicted and actual values. The choice of metric significantly influences both the interpretation of model performance and the outcome of model comparison. The section here goes into deeper detail about the most commonly applied sets of metrics for performance evaluation in time series and supply chain forecasting—Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R-squared (R^2)—and it also explores the mathematical nature of these metrics, their practical usefulness, and the limitations given by concrete examples from the reviewed literature.

2.6.1 Root Mean Squared Error (RMSE)

This metric frowns upon big errors more than the small ones because of the squaring function that makes it very sensitive to outliers or forecast misses in the case of sales during the peaks. RMSE is especially good in inventory-sensitive settings, where big mistakes can lead to high operational cost like stockout or overstock.

RMSE was the primary metric in several reviewed studies:

- (Feizabadi, 2022) used RMSE to compare ARIMA, Holt-Winters, and ML models for steel demand.

- (Pacella & Papadia, 2021) showed LSTM and BiLSTM achieving significant RMSE reductions over Exponential Smoothing models.

Although RMSE gives a definite indication of the error size, it is affected by the scale of the data, so it is not suitable for comparing SKUs with different sales volumes across the board unless it has been normalized.

2.6.2 Mean Absolute Error (MAE)

MAE represents the average magnitude of forecast errors without considering direction. Unlike RMSE, MAE treats all errors equally and is less sensitive to outliers. This makes it particularly useful for retail forecasts involving highly variable SKUs where a few extreme errors could disproportionately bias RMSE.

MAE was reported in numerous studies:

- (Shih & Rajendran, 2019), MAE was used to compare ARIMA, Holt-Winters, and ML models for blood supply forecasting.
- (Raiyani et al., 2021), MAE helped evaluate Prophet and Box-Cox–transformed models for retail forecasting.

Nevertheless, MAE does not have a normalized scale and may lead to wrong conclusions when deciding between products that have very different sales ranges. This makes it less ideal for unbalanced datasets unless each product is scaled.

2.6.3 Mean Absolute Percentage Error (MAPE)

Based on what we know, MAPE conveys error as a percentage, which makes it scale free and therefore appropriate for comparing performance among different products or stores. Nevertheless, it has a mathematical instability problem when Y_t gets very close to zero and becomes undefined for zero actual values—thus the problem is still there in sales data of the SKU level.

For practical usage, MAPE should be considered as a comparative indicator but still should be accompanied by scale-dependent ones such as MAE or RMSE to get a full picture of performance.

2.6.4 R-squared (R^2)

The coefficient of determination, or R^2, measures the proportion of variance in the dependent variable that is predictable from the independent variables. It is typically used in regression-based models to assess explanatory power.

While R^2 is intuitive and interpretable, it is not ideal for evaluating out-of-sample forecasts, especially for time series with strong autocorrelation. This limitation was noted in Makridakis et al. (2018), who caution that R^2 may overestimate performance in high-variance data if the model is overfit.

2.6.5 Alternative and Composite Metrics

Some studies propose more nuanced metrics:

- MASE (Mean Absolute Scaled Error) was used in (Pacella & Papadia, 2021) to benchmark LSTM performance against naïve baselines, with lower values indicating better performance.
- SMAPE (Symmetric MAPE) corrects for MAPE's asymmetry but can still suffer from scale issues it was also used in (Pacella & Papadia, 2021).
- RMSE (fWCscaled), introduced in (Bakker & Pechenizkiy, 2009), scales RMSE by worst-case error, providing a bounded [0,1] scale that facilitates comparison across products.

These alternatives are particularly useful when evaluating heterogeneous datasets, such as SKU-level sales, where a single metric can be misleading.

2.6.6 Metric Selection Guidelines

The reviewed literature clearly supports the use of **multiple metrics**, rather than relying on a single number, to capture different aspects of model performance.

Use Case	**Recommended Metric(s)**	**Rationale**
Single SKU forecast	RMSE, MAE	Accurate measure of error magnitude
Multi-product comparison	MAPE, MASE, RMSE (fWCscaled)	Scale-independent; better across units
Promotion forecasting	MAE, RMSE	Sensitive to peaks and outliers
Model interpretability assessment	R^2 (with caution)	Useful in regression and Random Forest analyses
Sparse, intermittent demand	MASE, SMAPE	Stable under zero or near-zero actuals

Table 2.3: Metric Selection Guidelines

2.7 Gaps in Literature and Motivation for Current Study

Despite the vast and rapidly expanding body of literature on sales and demand forecasting in supply chain management, several critical gaps remain—both **methodological** and **practical**. Such gaps have deep consequences for model choice, execution, and actual operation. This part of the paper gathers the limitations found in the papers surveyed and points out the logic behind the present research that is designed to cover some of the uncovered corners.

2.7.1 Lack of Comprehensive Comparative Studies

A recurring observation throughout the literature is the scarcity of rigorous, head-to-head comparisons between classical statistical models, machine learning algorithms, and hybrid approaches using real-world supply chain datasets.

- Makridakis et al. (2018) emphasized that many forecasting studies compare new ML/DL methods against weak baselines (e.g., naïve or outdated models), which exaggerates performance claims.
- Shih & Rajendran (2019) found that in healthcare, statistical models still outperformed ML approaches in structured environments, questioning the generalizability of ML superiority.
- Ensafi et al. remains one of the few studies that directly compares SARIMA, Holt-Winters, Prophet, CNN, and Stacked LSTM using the same dataset—but even it is limited to a single product category (furniture).

Most other studies either:

- Evaluate only one class of models (e.g., only neural networks or only regression),
- Use synthetic or simulated datasets,
- Rely on specialized forecasting frameworks (e.g., PromoCast) without testing against standard alternatives.

2.7.2 Limited Use of Panel Structures in Forecasting

While panel data techniques are well established in econometrics and health forecasting (Baltagi, 2014), their application in retail demand forecasting remains limited. Most time series forecasting studies treat each product as an independent series, ignoring the potential to share statistical strength across similar items.

2.7.3 Underemphasis on Feature Engineering and Preprocessing

While model architecture often dominates forecasting discussions, feature engineering and preprocessing are equally critical for real-world performance. Many studies fail to document:

- How missing data is handled,
- How seasonality is encoded,
- Whether log transforms or differencing are applied.

For instance:

- Prophet in (Ensafi et al., 2022) performs well when holidays are defined, but few studies explain how event flags are created.
- (Chen et al., 2018) and(Pacella & Papadia, 2021) LSTM without specifying whether data was stationarized or normalized.
- Bakker & Pechenizkiy (2009) stress that forecast evaluation is highly sensitive to data scale and structure, yet many studies neglect this.

2.7.4 Gaps in Interpretability vs. Accuracy Trade-Offs

Another underexplored area is the trade-off between model accuracy and interpretability. While ML models often outperform classical approaches on accuracy metrics, their black-box nature limits adoption in business environments where transparent decision-making is required.

2.7.5 Motivation for the Current Study

Based on the gaps identified above, the current study was motivated by the need to:

1. Conduct a rigorous, multi-paradigm comparison of forecasting models on a real-world panel dataset.
2. Assess not just raw predictive performance, but also interpretability, robustness, and practical deployment considerations.
3. Apply a uniform, replicable data pipeline to all models, including lag construction, stationarity enforcement, and diagnostic testing.
4. Support both researchers and practitioners by bridging statistical theory, ML capabilities, and SCM needs.

By positioning classical models, modern ML, and hybrid approaches side by side—under consistent data and evaluation conditions—this thesis contributes to the evidence-based selection of forecasting tools for supply chain decision-making.

3

Chapter 3 Methodology

3.1 Dataset Overview

3.2 Data Preprocessing

3.3 Model Selection

3.4 Train-Validation-Test Splitting

3.5 Evaluation Metric

3.6 Software and Libraries

3.1 Dataset Overview

A publicly available dataset "DataCo Smart Supply Chain For Big Data Analysis" used by the company DataCo Global was used for the this study. The structure of this dataset allows the use of machine learning algorithms and R Software. Areas of important registered activities are Provisioning, Production, Sales, Commercial Distribution. It also allows the correlation of Structured Data with Unstructured Data for knowledge generation.

The dataset is a real-world collection featuring 118 unique products from 11 departments, shipped to 164 countries. Product prices range from $9.99 to $1,999.99, with data spanning from 2015 to 2018. It consists of 180,519 rows and 53 columns. Each row in the dataset represents a distinct supply chain transaction, and each column corresponds to various aspects of the transaction, including product details, customer information, shipping specifics, and geographical attributes. Below are the list of the variables/features available in this dataset, organized by the categories. The expanded table can be found at the appendix.

1. Financial Information

- Sales (USD): Transaction value (9.99 - 1999.99).
- Type: Transaction type (Cash, Debit, Payment, Transfer).
- Benefit per Order (USD): Profit after costs (-4274.98 to 911.8).
- Sales per Customer (USD): Total customer spending (7.49 - 1939.99).

2. Shipping Information

- Days for Shipping (Real/Scheduled): Actual/scheduled delivery days (0-6/0-4).
- Delivery Status: Delivery progress (Advance Shipping, Late Delivery, On Time, Cancelled).
- Late Delivery Risk: Binary indicator (0 = On time, 1 = Late).
- Shipping Date (DateOrders): Shipping timestamp (01/01/2016 - 09/30/2017).
- Shipping Mode: Delivery type (Standard, First, Second Class, Same Day).

3. Category Information

- Category ID/Name: Product category details (51 unique categories, e.g., "Sporting Goods," "Electronics").

4. Customer Information

- Identifiers: Customer ID (20,652), email, and password (protected).

- Demographics: City, country (US/Puerto Rico), state (46), ZIP code (996), street (7,458).
- Segment: Customer type (Consumer, Home Office, Corporate).

5. Department Information

- Department ID/Name: Product department (11 unique, e.g., "Apparel," "Footwear").

6. Market Information

- Location: Latitude (-33.93755 to 48.78193), Longitude (-158.026 to 115.2631).
- Market Region: Geographic market (Pacific Asia, USCA, Africa, Europe, LATAM).

7. Order Information

- Order Details: Order ID (65,752), date (01/01/2015 - 09/30/2017), city (3,597), state (1,089), ZIP (610).
- Order Status: Order completion status (9 types, e.g., Complete, Pending, Cancelled).
- Order Profit (USD): Per-order profit (-4274.98 to 911.8).
- Order Total/Discount (USD): Total amount (7.49 - 1939.99), discount amount (0-500), and discount rate (0-25%).

8. Product Information

- Product Details: Product ID (118), name, and price (9.99 - 1999.99).
- Category ID/Description: Category ID (51), product description (excluded).
- Product Status: Availability (missing).

Running the summary statistics command, in general, missing values are minimal for most variables; the customer's last name has three missing values, while the customer's zip code is missing 8. Nevertheless, the four main missing values are Product description, which is entirely missing, and order zip code, which has 155,679 missing values.

Sales, Unit sold, and orders placed are time-dependent variables, product prices also change over time. The goal is to predict future sales ($) of products using historical sales data with different time series forecasting models to observe differences between the models and draw comparisons. This study aims to do that by using sales ($) as the dependent variables and product price and order date (for seasonality purposes) as independent variables and country specific and product specific variables as fixed effects where applicable.

3.2 Data Preprocessing

We'll start by preprocessing the dataset to make sure it's ready for analysis. This involves handling missing values, removing any irrelevant or duplicate data, and fixing inconsistencies that might affect the results. After that, we'll look at basic statistics to get a sense of the data. We will check for outliers, and we'll also take a closer look at the time series to understand its key features, like trends and seasonality. In order to find reliable independent variables, correlation test will be conducted. Finally, group-level transformation, feature engineering and panel data transformation will be done.

3.2.1 Missing Values:

- The dataset contained columns such as "Product Description" and "Order Zip Code" with significant missing values—87% and 100%, respectively. These columns were removed as they did not contribute meaningfully to the analysis.
- The Customer zip code and Customer Last name showed minimal missing values; therefore, to preserve the rows, the missing values in both were replaced with "Unknown."
- Customer Email and Customer Password were dropped to maintain data integrity and ensure compliance with privacy guidelines.
- Since each customer had a unique customer ID to distinguish them, Customer Last Name and Customer First Names columns were dropped, due to privacy guidelines.
- Product Image and Product Status were not of any value to this research, so they were also removed.

3.2.2 Sales Distribution over the months

While observing the company's monthly sales over the years, there was a significant disruption in data from October 2017 to January 2018. In October 2017 the items sold were all priced over 999$ which were all outliers and already dropped which made October 2017 sales nearly 0$ then for November 2017, December 2017 and January 2018, the drop in sales was significantly high. The reason for this observation is the huge difference in data available for these months compared to other months. While every month in 2015, 2016 and rest of 2017 each have approximately 3 thousand sales data available, these four months only account for 1300 sales data which explains the drop in sales. It seems like the data entry wasn't complete in the last 4 months. Therefore, the October, November and December 2017 and January 2018 data was excluded from the analysis to prevent any disruptions. The Dataset shape after dropping these three months is (171784, 16).

Month	Data Count
1 July 2017	3208
1 August 2017	3269
1 September 2017	3073
1 October 2017	951
1 November 2017	1233
1 December 2017	1210
1 January 2018	1300

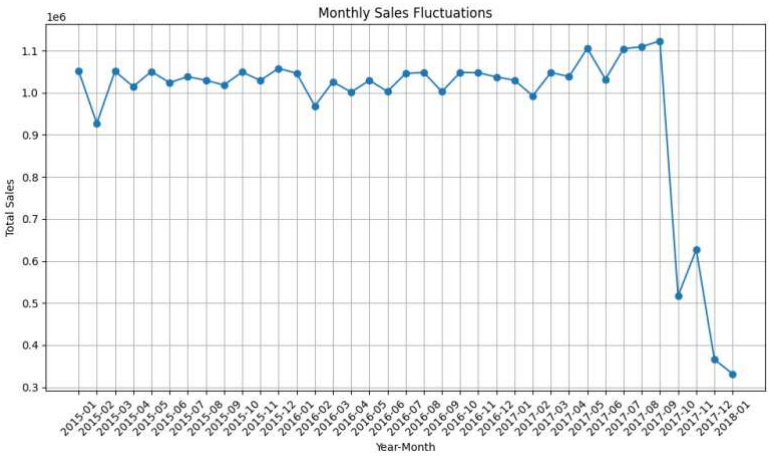

Figure 3.1: Monthly Sales Fluctuations before dropping last three months of 2017 and 2018 data

To dig deeper into the historical sales of the company and to check for trends and seasonality, monthly distribution of sales of the company was plotted in figure 3.2.

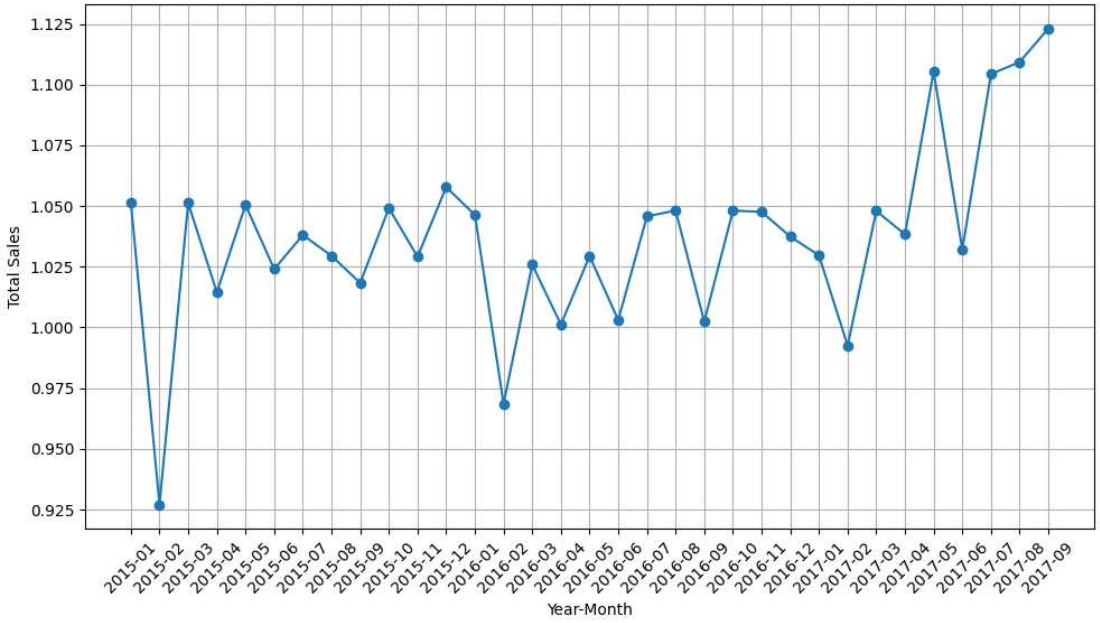

Figure 3.2: Monthly Sales Fluctuations After Dropping Last 3 months of 2017 and January 2018

3.2.3 Handling Outliers:

As shown in Figure 3.3, there are clear outliers present in key numerical variables, especially in the *Sales* column. Some sales values exceed $550, with a few reaching up to $2000. These values lie far outside the typical range defined by the interquartile range (IQR) method, and their presence can significantly distort the overall analysis. Keeping such extreme values can

lead to biased results by affecting measures like the average and by exaggerating the variation in the data.

To address this issue, outliers were removed in Sales, Product Price and Order Item Discount. This step ensures that the analysis is based on data that more accurately reflects normal business activity. After removing the outliers, the dataset was reduced from 171,784 to 165,263 rows.

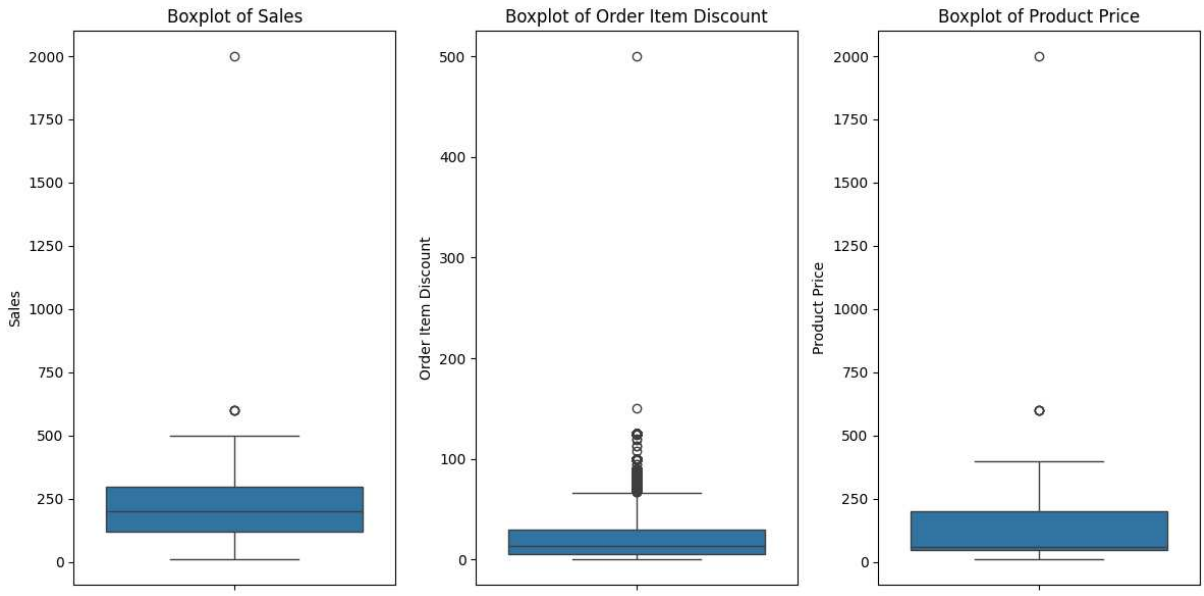

Figure 3.3: Boxplot of Sales to Determine Outliers

3.2.4 Variable Selection and Correlation Analysis

In order to decide on the independent variables for prediction of Sales (in USD), a correlation analysis has been conducted on the pre-processed dataset containing 165,263 records, after eliminating duplicates, resolving missing values, and implementing an outlier threshold at $550. Pearson correlation coefficients were computed between Sales and other numeric variables which are Late_delivery_risk, Order Item Discount, Order Item Discount Rate, Order Item Quantity, Order Item Profit Ratio, Order Profit Per Order, Benefit per order, Days for shipping (real), Days for shipment (scheduled), and Product Price. This research was designed to find variables with significant correlations to Sales, thus providing a solid foundation for feature selection in a predictive model.

The results indicated that Product Price had a strong positive correlation with Sales ($r = 0.6889$), suggesting that higher-priced products tend to generate higher revenue, which is consistent with the dataset's structure where sales reflect revenue (price × quantity). Order Item Discount also showed a moderate positive correlation ($r = 0.4893$), indicating that higher discount amounts may be associated with increased sales, as expected in retail environments. In contrast, Order Item Discount Rate had a moderate negative correlation ($r = -0.1114$), possibly reflecting

diminishing returns when discount percentages are too high. Other variables, such as Order Item Quantity (r = 0.1810) and Benefit per order (r = 0.1215), exhibited weak positive correlations, while Late_delivery_risk and Order Item Profit Ratio had negligible correlations (r = -0.0025 and r = -0.0026, respectively). Based on these findings and their practical relevance, Product Price and Order Item Discount were selected as independent variables for modeling, given their stronger and more interpretable relationships with Sales. Variables with minimal or non-significant associations, such as Late_delivery_risk and Order Item Profit Ratio, were excluded to enhance model efficiency and interpretability. Multicollinearity between the selected variables was assessed, showing a moderate positive correlation between Product Price and Order Item Discount ($r = 0.4615$, $p < 0.001$), However, this level of multicollinearity was deemed acceptable, as the Variance Inflation Factor (VIF) for these variables is approximately 1.27, well below the threshold of 5 where multicollinearity typically becomes problematic.

3.2.5 Panel Data Transformation

To support the comparative analysis of machine learning models in forecasting sales, the dataset was transformed into a proper panel data structure. Panel data integrates cross-sectional and temporal dimensions, enabling the modeling of sales (in USD) as a time-dependent variable while capturing product-specific dynamics. The transformation utilized a subset of the dataset, retaining only the columns relevant to the forecasting objective: Order Date, Product Name, Sales, Order Item Discount Rate, Product Price, and Order ID. This section outlines the steps to create a panel data frame optimized for forecasting sales ($) using historical data.

The transformation process involved the following steps:

1. Defining the Panel Structure:
    - The panel was constructed with products as the cross-sectional units and time (monthly intervals) as the temporal dimension. Each unique product (identified by Product Name, representing 86 unique products) was treated as an entity, and sales data were aggregated at the monthly level to support time series forecasting.
    - The dependent variable, Sales (in USD, ranging from 9.99 to 1999.99), was aggregated by summing the transaction values for each product per month. Independent variables included Product Price (averaged per product per month to account for price variations) and Order Item Discount (averaged per product per month to capture promotional effects). The Order Date was used to derive temporal features, such as month and year, for modeling seasonality.

2. Temporal Aggregation:

    - The Order Date was converted to a monthly time index by truncating timestamps to month-year format (e.g., January 2015). Transactions were grouped by Product Name and month-year to form the panel structure.
    - For months where a product had no sales, sales were imputed as 0 USD to maintain panel consistency, reflecting no transactions for that product in that period. To address incomplete data for October 2017 to January 2018 (as noted in Section 3.2.3.2), these months were excluded, resulting in a time frame from January 2015 to September 2017 (33 months).

3. Cross-Sectional Aggregation:

    - For each product, Sales (USD) was summed to represent the total transaction value per product per month. Product Price and Order Item Discount were averaged per product per month to create consistent panel variables, capturing time-varying price and discount dynamics.

4. Creating the Panel Data Frame:

    - The transformed dataset was structured as a panel with the following columns:
    - Product Name: Unique identifier for the cross-sectional unit (87 products).
    - Month-Year: Time index (e.g., 2015-01, 2015-02, ..., 2017-09).
    - Sales: Dependent variable, summed transaction value (USD) per product per month.
    - Product Price: Independent variable, averaged per product per month.
    - Order Item Discount Rate: Independent variable, averaged per product per month.
    - The resulting panel data frame contains up to 1761 potential observations, with some missing entries for products with no sales in specific months.

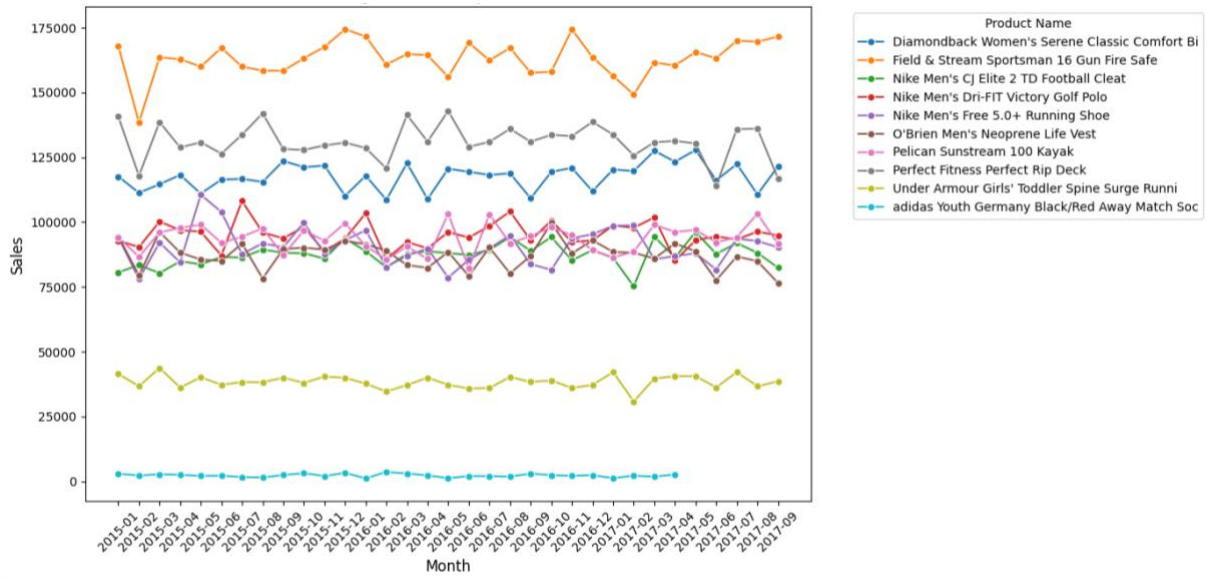

Figure 3.4 Monthly Sales Fluctuations of top 10 Products

5. Handling Panel Imbalance:

    - The panel was slightly unbalanced due to products with no sales in certain months (e.g., newly introduced or discontinued products). Zero sales (0 USD) were imputed for missing product-month combinations to ensure compatibility with time series models.

6. Verification of Panel Structure:

    - The panel structure was validated using Python (e.g., pandas library) to ensure each product-month combination was unique and that time series properties, such as trends and seasonality, were preserved. Summary statistics of the panel data were computed to verify the distribution of Sales (USD), Product Price, and Order Item Discount Rate across products and time. First Few rows of the data are as follows:

Product Name	YearMonth	Sales	Order Item Discount	Product Price
Bag Boy Beverage Holder	2015-02	424.8299961	9.197999954	24.98999977
Bag Boy Beverage Holder	2015-03	924.6299915	7.227692246	24.98999977
Bag Boy Beverage Holder	2015-04	924.6299778	14.46249986	24.98999977
Bag Boy Beverage Holder	2015-05	849.6599921	5.10090906	24.98999977
Bag Boy Beverage Holder	2015-06	874.6499842	10.09599991	24.98999977

Table 3.1: Panel Data Sample

3.2.7 Seasonality

Figure 3.5 displays the monthly total sales for all products from January 2015 to September 2017. A visual examination suggests a relatively stable trend with moderate fluctuations throughout the observed period. While there is no strong, consistent upward or downward long-term trend, a subtle increase in total sales can be noticed toward the end of the series, particularly in early to mid-2017. Additionally, there are indications of potential seasonality, with noticeable dips occurring around the beginning of each year (e.g., February 2015 and February 2016), followed by gradual recoveries. These recurring low points may correspond to seasonal demand variations or post-holiday slowdowns, suggesting a cyclical sales pattern.

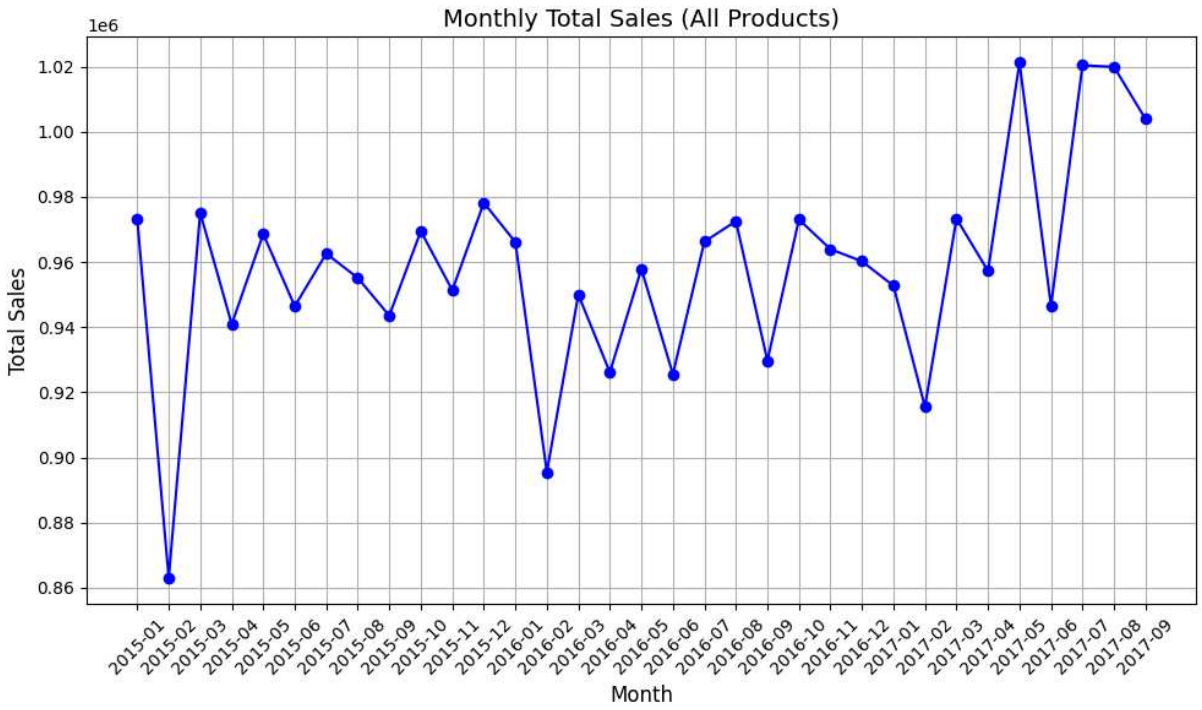

Figure 3.5: Monthly Total Sales to Check for Seasonality and Trends

Figure 3.6 presents the results of a seasonal decomposition of the monthly sales time series into its primary components: trend, seasonal, and residual. The top panel shows the original sales values, which exhibit noticeable fluctuations throughout the period. The second panel reveals the underlying trend component, which initially declines slightly before beginning a gradual upward movement in the later months, indicating mild growth in overall sales performance over time.

The third panel captures the seasonal component, which demonstrates consistent, repeating patterns across the time horizon. This confirms the presence of seasonality in the sales data, with certain months systematically performing better or worse than others. The bottom panel shows the residuals, which are relatively small and evenly distributed, suggesting that most of

the variance in the original series is well-explained by the trend and seasonal components. Overall, this decomposition validates the assumption of seasonality and a mild trend, both of which must be accounted for in any time series forecasting.

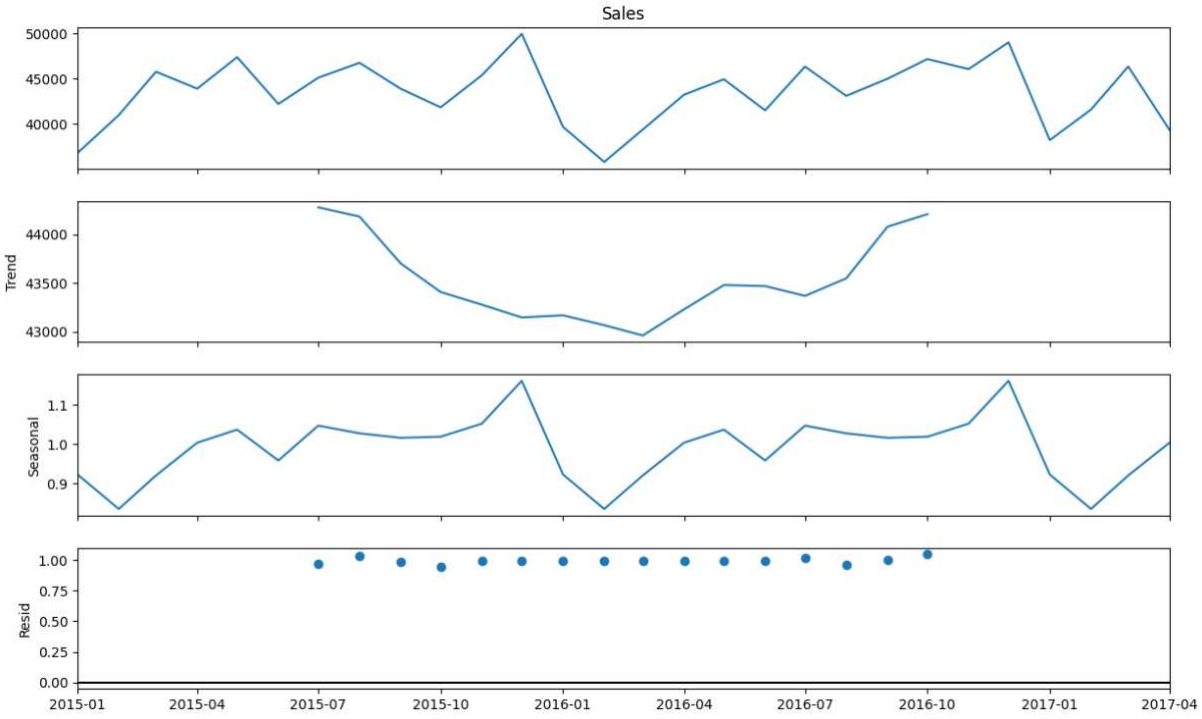

Figure 3.6: Seasonal Decomposition

Also, the heatmap of monthly sales over the years in figure 3.7 show which months' sales were significantly more or less than the rest over the years and if there were any obvious patterns in monthly sales in different years.

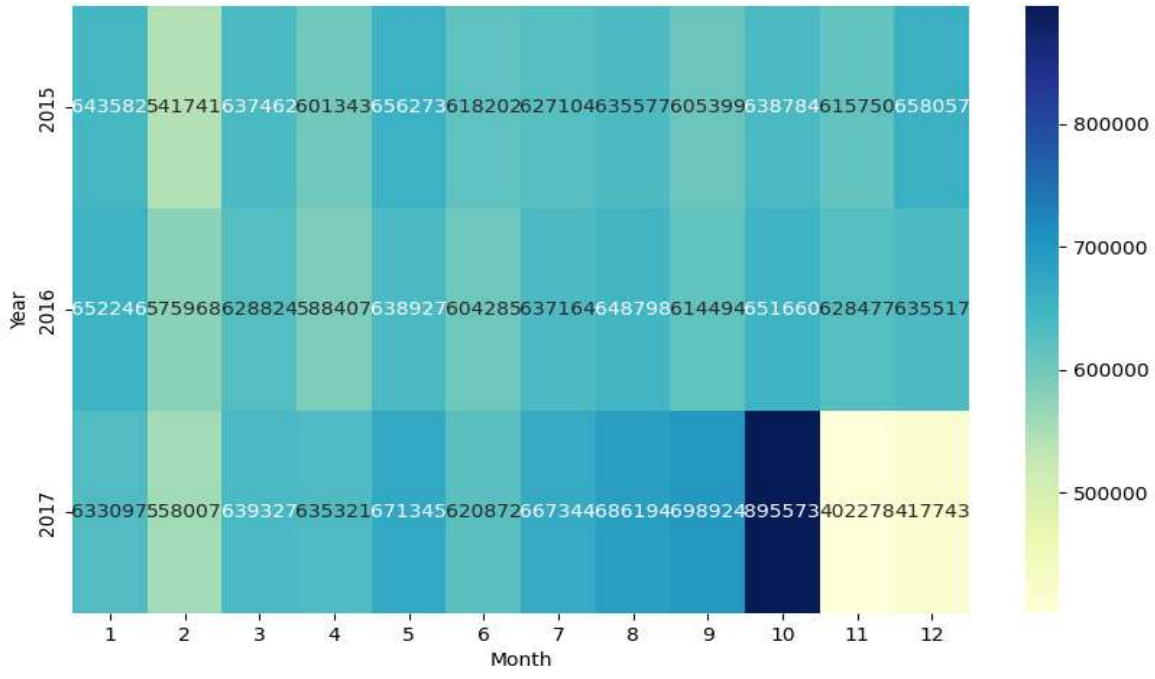

Figure 3.7: heatmap of Monthly Sales

3.2.6 Stationarity

To evaluate the stationarity of the time series, an Augmented Dickey-Fuller (ADF) test was performed on the total monthly sales (aggregated across all products) from January 2015 to September 2017. The ADF test yielded a test statistic of -2.2286 and a p-value of 0.1960. As the p-value exceeds the 0.05 threshold, the null hypothesis of a unit root cannot be rejected, indicating that the time series is non-stationary.

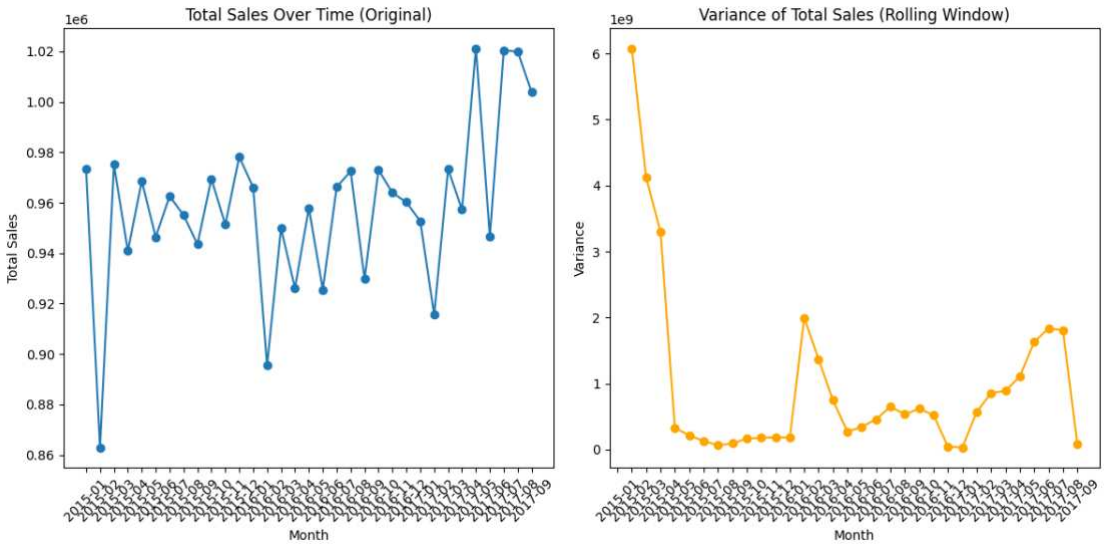

Figure 3.8: Mean and Variance Fluctuations of Sales Over Time

Visual inspection of the series revealed fluctuations in both mean and variance over time, confirming the presence of trend and heteroscedasticity. To address this, first-order differencing and logarithmic transformation were applied to stabilize the mean and variance. A subsequent ADF test on the transformed series produced a test statistic of -14.8635 and a p-value of 0.0000, confirming stationarity. Plots of the mean and variance further supported this conclusion, indicating the series is now suitable for time series modeling and forecasting.

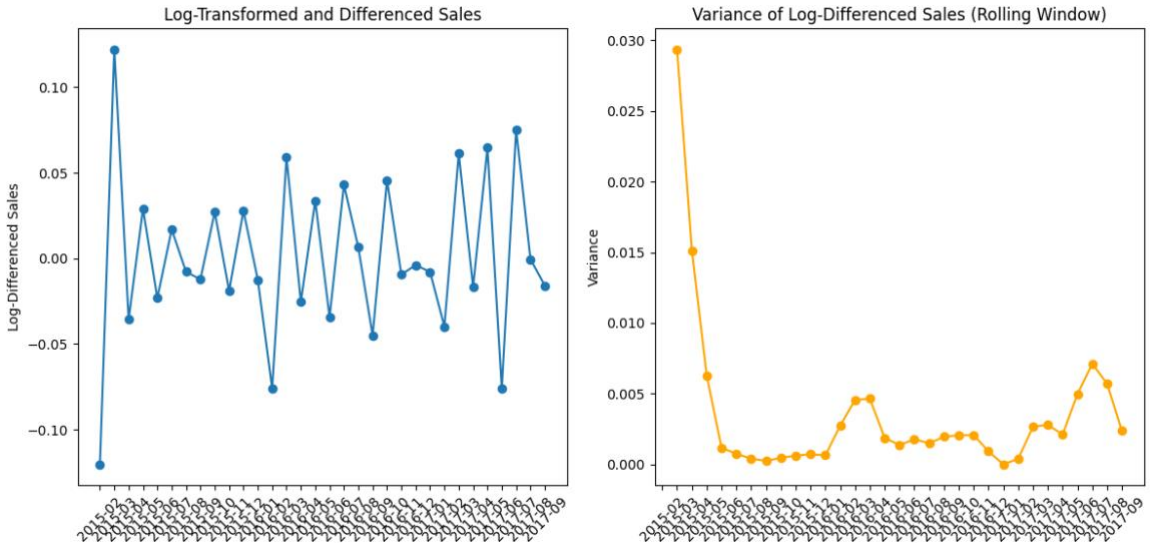

Figure 3.9: Mean and Variance of Sales After Log Transformation and First Level Differencing

3.2.8 Feature Engineering

To account for temporal dependencies and persistence in product-level sales, a lagged sales variable was introduced as part of the feature engineering process. Specifically, the natural logarithm of sales was first differenced to achieve stationarity in the previous section, and then the lag of this transformed variable was computed for each product over time. This lagged feature reflects how the previous month's change in sales influences current sales. Including lagged sales helps control for unobserved time-related factors and improves the predictive accuracy of the model by incorporating information about past performance.

3.3 Model Selection

This study compares the performance and results of Machine Learning models in predicting sales within a supply chain context. This section outlines the criteria for selecting models, describes the proposed algorithm, and discusses the relevance of the models for forecasting sales. Ultimately, the portfolio of Random Forest, XGBoost, and Prophet was strategically chosen to create a comprehensive and multi-faceted comparative analysis. Random Forest and XGBoost represent the powerful and widely adopted class of ensemble learners while Prophet offers a modern, interpretable, and automated alternative that is popular among practitioners. By comparing this diverse set of models, the study can robustly investigate the performance of different ML philosophies when applied to the specific complexities of panel data sales forecasting. This aligns with the call from researchers like Makridakis et al. (2018) to systematically compare and understand the strengths and weaknesses of various statistical and ML forecasting methods, thereby providing a thorough and insightful contribution to the field.

3.3.1 Baseline Model

After conducting careful feature selection and engineering— including the removal of outliers, ensuring stationarity in the time series, and identifying relevant predictors— the following regression model was specified:

$$Y_{it} = \alpha_0 + \alpha_1 X_{1(it)} + \alpha_2 X_{2(it)} + \alpha_3 X_{3(it)} + \epsilon_{it}$$

In this model, Y_{it} represents the sales of product i in month t; α_0 is the intercept term; $X_{1(it)}$ denotes the price of product i in month t; $X_{2(it)}$ is the discount applied to product i in month t; $X_{3(it)}$ captures the lagged sales of product i in month t and ϵ_{it} is the error term.

To determine the appropriate panel data model, the **F-Limer test** (Chow Test for Panel Data) was conducted to compare the Pooled OLS model with the Fixed Effects model. The test yielded an F-statistic of **0.6824** with a corresponding p-value of **0.9880**. Given the high p-value,

the null hypothesis that individual-specific effects are not significant could not be rejected. This indicates that there is no statistical evidence of unobserved heterogeneity across entities, and thus, the Pooled OLS model is deemed sufficient for estimating the relationship between sales and the selected independent variables.

To check if the pooled OLS model suffices, the Breusch-Pagan test was applied to evaluate the presence of heteroskedasticity in the residuals, The test yielded an LM statistic of 4.3777 with a p-value of 0.1120, indicating no significant evidence of heteroskedasticity at the 5% significance level. This supports the assumption of homoskedasticity in the Pooled OLS model. Additionally, the Durbin-Watson statistic was calculated to detect autocorrelation in the residuals, resulting in a value of 2.3986. this value is less than threshold of 2.5 suggesting that residuals are not first-order serially correlated, and the assumption of no autocorrelation is satisfied. Together, these diagnostic results affirm the statistical reliability of the Pooled OLS model estimates.

3.3.2 Tree-Based Models

The selection of Random Forest and XGBoost is motivated by the consistent success of ensemble machine learning methods in demand forecasting applications, as documented throughout the supplied literature. These models are powerful, non-linear techniques capable of capturing complex interactions and patterns in data without requiring strong assumptions about the underlying data distribution, a limitation often present in classical statistical models. Early work by Carbonneau et al. (2008) established the utility of machine learning in this domain, and more recent reviews confirm that tree-based ensembles are central to modern predictive analytics in supply chains (Seyedan & Mafakheri, 2020). Specifically, Random Forest is chosen for its robustness and relative simplicity in preventing overfitting by averaging multiple decision trees. XGBoost, a state-of-the-art implementation of gradient boosting, is included for its widely recognized high performance, computational efficiency, and built-in regularization, which often gives it a competitive edge in forecasting competitions and practical applications (Makridakis et al., 2018). Including both provides a robust representation of the most established and powerful general-purpose ML forecasting models.

3.3.3 Specialized Time Series Model

Prophet is chosen to represent a distinct and highly practical approach to forecasting that addresses the critical trade-off between performance and interpretability, a challenge highlighted in the literature (Baryannis et al., 2019; Trusov et al., 2006). While traditional statistical models offer interpretability and complex "black box" ML models often prioritize

accuracy, Prophet is designed to provide both. It is an automated forecasting procedure that decomposes a time series into trend, seasonality, and holiday components, making the forecast outputs easy to understand. This is particularly relevant given the emphasis on forecasting seasonal items in your literature (Ensafi et al., 2022). Although Prophet may not be explicitly named in older academic texts, its methodology directly confronts the core challenges—such as handling multiple seasonality, promotions, and missing data—that are pervasive in sales forecasting. It serves as an essential modern benchmark, bridging the gap between classical decomposition methods and scalable machine learning implementation.

3.4 Train-Validation-Test Splitting

In time series forecasting, preserving the temporal order of observations is crucial to prevent data leakage and ensure realistic model evaluation. Unlike random sampling methods common in cross-sectional data, this study adopted a chronologically consistent train-test split to reflect the nature of sales data evolving over time.

The complete panel dataset, consisting of monthly sales aggregated at the product level, spans from January 2015 to September 2017. To evaluate model performance on unseen future data, the dataset was split such that all months up to March 2017 were allocated to the training set, while the remaining period from April 2017 to September 2017 served as the test set. This corresponds to a roughly 80/20 temporal split, aligning with best practices in time series forecasting.

Importantly, the split was performed at the panel level—ensuring that each product's historical data remained intact across both sets. Features including lagged sales, product price and discount were all generated prior to splitting, so that the test set would benefit from the same feature engineering logic as the training set.

3.5 Evaluation Metrics

To assess the performance Machine Learning models in sales forecasting, a set of robust evaluation metrics is used. These metrics ensure a comprehensive evaluation of both predictive accuracy and model reliability,

1. Root Mean Squared Error (RMSE): Measures the average magnitude of prediction errors, emphasizing larger deviations, to penalize large errors and provide an overall measure of accuracy for continuous sales predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i \widehat{-(y_i)}^2}$$

2. Mean Absolute Error (MAE): Measures the average magnitude of prediction errors in absolute terms. This metric offers an interpretable metric to understand the average deviation of predictions from actual values.

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

3. R-Squared (R^2): Represents the proportion of variance in the target variable explained by the model. This can indicate the goodness-of-fit of the model for sales forecasting.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}\left(y_i - \underline{y_i}\right)^2}$$

4. Mean Absolute Percentage Error (MAPE): Measures the average percentage error relative to actual sales values. This is useful for comparing prediction errors across products with varying sales scales.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y_i}}{y_i}\right| \times 100$$

3.6 Software and Libraries

All data preprocessing, feature engineering, visualization, and modeling tasks were conducted using **Python (version 3.13)**, a versatile programming language widely adopted in the fields of data science and machine learning. Development was carried out in **Jupyter Notebook**, offering an interactive environment for combining code execution with documentation and visual outputs. Python was selected for its rich ecosystem of libraries tailored to time series forecasting, regression modeling, and data visualization. The following Python libraries and frameworks were extensively used in this study (Appendix B, C):

- **pandas**: Utilized for data wrangling, including reading and merging datasets, handling missing values, transforming transactional-level records into a monthly panel structure, and generating lag features. Its groupby, pivot_table, and merge functionalities enabled efficient aggregation and transformation of sales data.
- **numpy**: Used for numerical operations, including log transformations and array-based manipulations required during differencing, normalization, and model preparation.
- **matplotlib.pyplot** and **seaborn**: These libraries were used for data visualization, helping to plot time series trends, detect outliers, and visualize decomposition

components. Seaborn was particularly useful for producing heatmaps and boxplots during exploratory analysis.

- **statsmodels**: Specifically, statsmodels.api and statsmodels.tsa were employed for statistical modeling and diagnostics. The library facilitated Augmented Dickey-Fuller (ADF) and KPSS tests for stationarity, seasonal decomposition of time series, and implementation of Pooled OLS and Fixed Effects models using panel data structures.
- **linearmodels**: This specialized library supported panel data regression models, including Pooled OLS, Fixed Effects (Within), and Random Effects estimators. It also enabled the execution of model selection tests like the F-Limer test for poolability.
- **scikit-learn**: Used for implementing the **Random Forest** regressor and supporting utilities such as train-test splitting, cross-validation, and error metric calculation (e.g., MAE, RMSE, MAPE). It also provided tools for preprocessing steps, such as scaling and encoding.
- **xgboost**: Integrated via the XGBRegressor class, this library was used to build gradient boosting models tailored to capture complex non-linear patterns in the feature space. XGBoost's efficient implementation and feature importance tools made it ideal for comparative performance benchmarking.
- **prophet**: Developed by Meta, this library was used to implement the **Facebook Prophet** model. It allowed intuitive modeling of additive components (trend, seasonality, holidays) and supported custom seasonality configuration, change-point detection, and external regressors.

4

Chapter 4 Results

4.1 Chapter Overview

This chapter presents the results of various machine learning models applied to the task of time series forecasting of sales. Each model is introduced individually, with its forecasting performance, implications, and limitations discussed in detail. Following the individual evaluations, a comparative analysis is conducted based on the evaluation metrics defined in Section 3.7.

The dataset used for modeling consists of cleaned panel data consisting of 86 unique products sales observed in a 33-month period along with associated features such as price discounts and lagged sales.

As outlined in Section 3.6, the following models were implemented:

- Pooled Ordinary Least Squares (OLS) as the baseline model,
- Random Forest and XGBoost as tree-based models,
- Facebook Prophet as the specialized time series model.

The results from each of these models are presented in the subsequent sections, followed by a comparative analysis to identify the most effective approach for sales forecasting in this context.

4.2 Baseline Model Results

After preparing the dataset for panel analysis, the F-Limer test indicated that a pooled OLS model was sufficient, as fixed or random effects were not statistically necessary. When estimating the model, Product Price was found to exhibit perfect multicollinearity, meaning it is a linear combination of other variables. As a result, it was automatically excluded from the model. The final specification includes two explanatory variables: Order Item Discount and diff_log_lagged_sales. To ensure reliable inference, robust standard errors (HC1) were applied. These adjust for potential heteroskedasticity and provide more consistent estimates of standard errors. The model results, shown in Table X, reveal that:

- Order Item Discount has a positive and statistically significant effect on changes in log sales (coefficient = 0.0040, p = 0.026). This suggests that offering discounts is associated with a modest increase in sales growth.
- diff_log_lagged_sales has a strong negative and highly significant relationship with current sales growth (coefficient = -0.4548, $p < 0.001$), implying that higher past growth is associated with slower current growth—potentially reflecting mean reversion.

- The intercept is also statistically significant and negative (coefficient = -0.0637, p = 0.013), indicating a baseline decline in sales growth when the other explanatory variables are zero.

The model has an R-squared of 0.209, meaning it explains approximately 21% of the variation in the change in log sales. Evaluation metrics indicate moderate predictive accuracy, with Mean Absolute Error of 0.3318 and Root Mean Squared Error of 0.4652. The Mean Absolute Scaled Error (MASE) is 0.5250, which is well below 1. This means the model performs about 47.5% better than a naive forecast, a strong indicator of its predictive usefulness. MAPE was consequently in million percent which happens when the log transformed data were near 0 and MAPE is sensitive to near 0 or 0 observations. Since the study's focus is on comparing the models, and machine learning models are using raw sales as the dependant variable not the log transformed sales, the predictions were inverse-transformed, and then the evaluation metrics were recalculated. The result showed MAE of 1320.27, RMSE of 3143.53 and the MAPE was 38.55%. The recalculated evaluation metrics after inverse-transforming the predictions show a more direct reflection of the model's performance in raw sales terms. The MAE of 1320.27, although higher than the log-scale MAE, still provides a manageable error size depending on the overall sales volume. However, the RMSE of 3143.53 points to significant deviations, particularly when predicting extreme sales figures. The model's predictive accuracy is still moderate but reveals a tendency to struggle with larger sales values.

Pooled OLS Regression Results						
Dep. Variable:		diff_log_sales	R-squared:			0.209
Model:		OLS	Adj. R-squared:			0.208
Method:		Least Squares	F-statistic:			131.1
Date:		12-Jun-25	Prob (F-statistic):			2.22E-53
Time:		07:51:08	Log-Likelihood:			-1031.3
No. Observations:		1578	AIC:			2069
Df Residuals:		1575	BIC:			2085
Df Model:		2	Covariance Type:			HC1
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0637	0.0260	-2.4780	0.0130	-0.1140	-0.0130
Order Item Discount	0.0040	0.0020	2.2200	0.0260	0.0000	0.0080
diff_log_lagged_sales	-0.4548	0.0290	-15.7120	0.0000	-0.5120	-0.3980
Omnibus:		147.3700	Durbin-Watson:			2.399
Prob(Omnibus):		0.0000	Jarque-Bera (JB):			337.456
Skew:		-0.5570	Prob(JB):			5.28E-74
Kurtosis:		4.9730	Cond. No.			28.1

Table 4.1: Pooled Ols Regression Result

Diagnostic tests show no major violations of model assumptions:

- The Breusch-Pagan test yields a *p*-value of 0.112, indicating no significant heteroskedasticity.
- The Durbin-Watson statistic of 2.3986 is less than 2.5, suggesting no serious autocorrelation in the residuals.

4.3 Tree Based Model Results

4.3.1 Random Forest Results

To complement the linear modeling approach, a Random Forest Regressor was trained to forecast sales using the same set of explanatory variables. After tuning the model with grid search and cross-validation, the optimal hyperparameters were determined to be: n_estimators = 300, min_samples_split = 5, min_samples_leaf = 6, and max_depth = 20.

The model's predictive performance is summarized as follows:

- R^2 Score: 0.651
- Mean Absolute Error (MAE): $332.70
- Root Mean Squared Error (RMSE): $420.21
- Mean Absolute Percentage Error (MAPE): 35.65%

An R^2 of 0.651 indicates that the Random Forest model explains approximately 65.1% of the variance in actual sales in the test period, reflecting a strong fit within the evaluation sample. The pooled OLS model, with an R^2 of 20.9% over the entire period, serves as a baseline reference, although the two figures are not directly comparable due to differences in their evaluation windows. The MAPE of 35.65% reflects a reasonably good forecasting performance, though still with some margin of error, especially for higher-value sales.

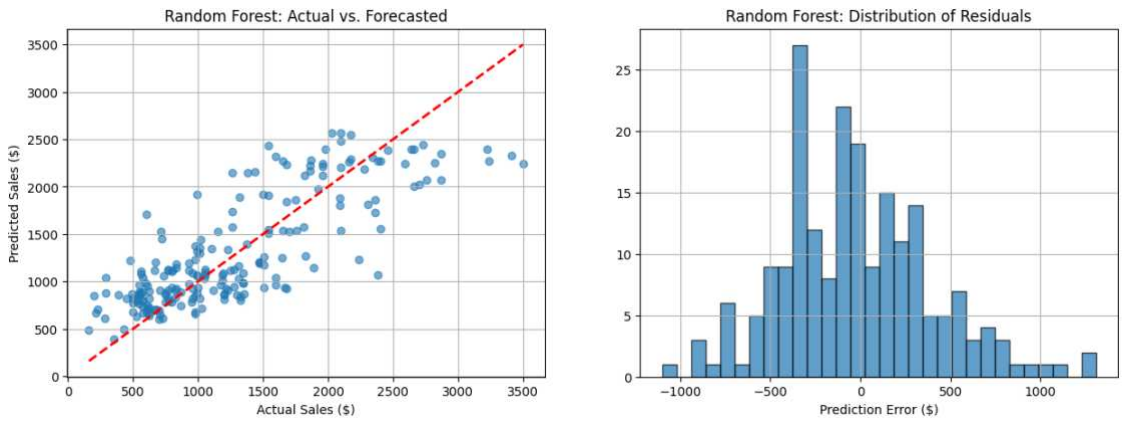

Figure 4.1: Random Forest Result

Figure 4.1 displays two diagnostic plots. The left plot shows the actual vs. predicted sales values, where most data points cluster around the red dashed 45-degree line, indicating general accuracy in prediction. However, some dispersion is evident, especially at higher sales values, which the model tends to underestimate slightly.

The right plot presents the distribution of residuals (prediction errors). The residuals are approximately symmetric and centred around zero, which is a desirable property, indicating that the model does not suffer from systematic bias. The distribution also shows a moderate spread, with most prediction errors falling within the ±500 range.

4.3.2 XGBoost Results

An Extreme Gradient Boosting (XGBoost) model was also employed to forecast sales, leveraging its strength in handling nonlinearities, variable interactions, and overfitting control through regularization. Hyperparameter tuning was performed using grid search and cross-validation, yielding the optimal settings: n_estimators = 100, max_depth = 5, learning_rate = 0.05, subsample = 0.7, and colsample_bytree = 0.8.

The model's performance is summarized below:

- R^2 Score: 0.615
- Mean Absolute Error (MAE): $344.41
- Root Mean Squared Error (RMSE): $441.12
- Mean Absolute Percentage Error (MAPE): 35.96%

The R^2 value indicates that 61.5% of the variance in sales is explained by the model, which, although slightly lower than the Random Forest's R^2 of 0.651, still represents a substantial improvement over the pooled OLS model. The MAE and RMSE values suggest prediction errors comparable to those of the Random Forest model, while the MAPE of 35.96% indicates that the average prediction deviates from actual values by approximately 36%.

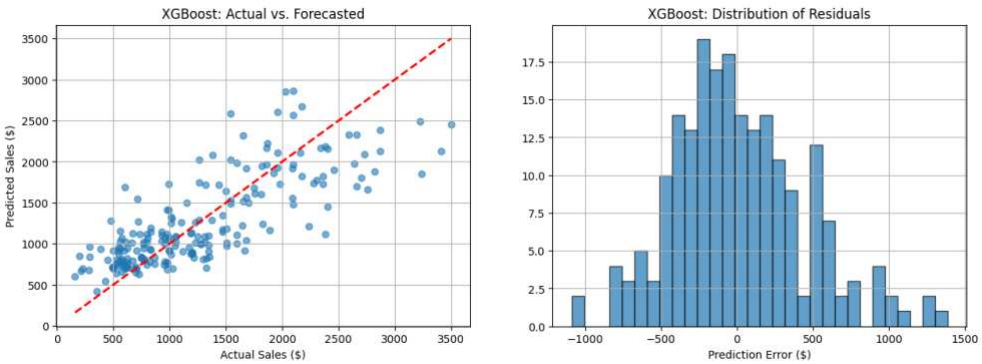

Figure 4.2: XGBoost Result

The diagnostic plots in Figure 4.2 support these findings. The left panel illustrates the predicted vs. actual sales values, showing that most predictions fall close to the 45-degree reference line, with some underestimation at the higher end of sales values. The right panel presents the distribution of residuals, which are relatively symmetric and centered around zero, although with a mild right skew. This pattern suggests that while the model performs well overall, it may slightly under-predict higher sales values more frequently than it over-predicts them.

Despite its slightly lower R^2, the XGBoost model demonstrates robust predictive performance and offers the added benefit of faster training and better handling of sparse data relative to ensemble models like Random Forest.

4.4 Specialized Time Series Model Results

The Facebook Prophet model was employed to forecast monthly product-level sales using a time series approach augmented with exogenous variables. Prophet is a modular additive model specifically developed for time series forecasting with strong seasonal effects and historical trend components. It is particularly effective in business settings where data is noisy and domain knowledge (e.g., holidays or events) can be incorporated.

In its standard configuration, Prophet is a univariate model, meaning it forecasts a single target variable (e.g., sales) using time as the sole regressor (ds, date) and the dependent variable (y). However, the model can be extended to include additional external regressors, which was the approach taken in this analysis. Specifically, price, order item discount, and lagged log sales were added to the model to capture influential economic and behavioral factors.

Despite this extension, the model's forecasting accuracy was relatively low compared to the machine learning models evaluated (e.g., Random Forest and XGBoost).

The Mean Absolute Error (MAE) of approximately $456 and Root Mean Squared Error (RMSE) of over $615 indicate that, on average, Prophet's predictions deviated from actual sales values by a substantial margin. Furthermore, the average residual standard deviation of $610.67 and wide prediction intervals (mean width of $1,264.25) suggest that the model exhibited high uncertainty and imprecise fit across the time series forecasts.

These subpar results can be attributed to several structural and data-related limitations:

1. Modeling Limitations with Multivariate Structures: Although Prophet supports external regressors, it models their effects in an additive linear fashion. This restricts its ability to learn complex, non-linear relationships or interactions between features, such as how discount and prior sales jointly affect demand. In contrast, machine learning models like

Random Forest or XGBoost can flexibly capture such dependencies, leading to superior predictive performance.

2. Coarse Time Granularity: The data was aggregated at a monthly level, which likely diminished the model's ability to detect sharp fluctuations, short-term trends, or promotional effects that might occur over shorter periods (e.g., weekly or daily). Prophet performs best when the time series shows clear seasonal cycles and frequent data points to enable automatic changepoint detection.

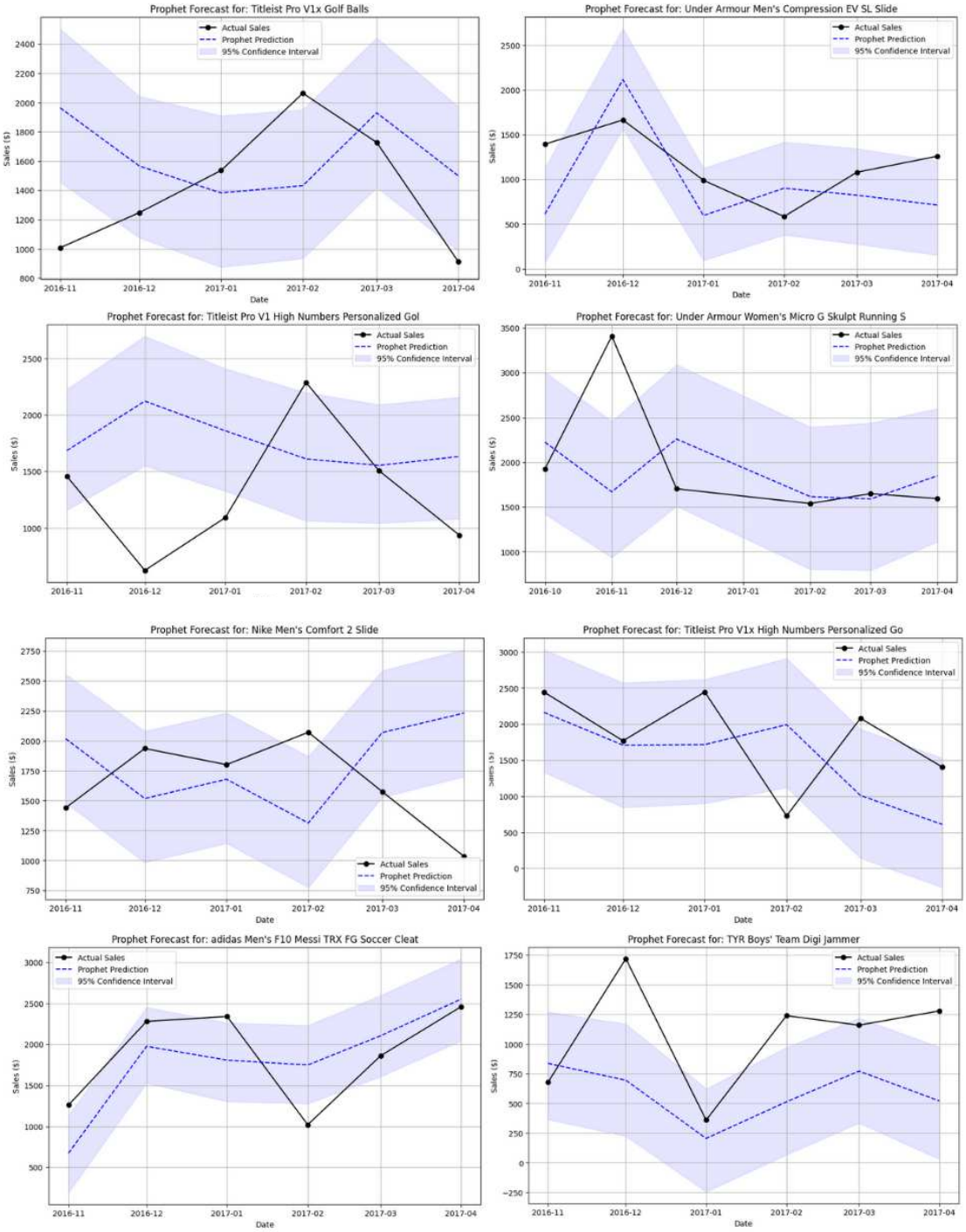

Figure 4.3. Propeht Result Per Product with 95% Confidence Interval

3. High Forecast Uncertainty: The wide average forecast intervals reflect Prophet's conservative uncertainty estimation, which may be a response to the limited signal available at the monthly frequency combined with multicollinearity between regressors. These wide bounds, while statistically valid, reduce the practical utility of the forecasts for decision-making.

4. Data Complexity vs. Model Simplicity: Prophet is deliberately designed to be simple, interpretable, and robust to missing data or outliers. However, in this context, the underlying sales patterns are driven by complex interplays of price sensitivity, promotional timing, and seasonality—dynamics that a simple additive model may not fully capture.

Figure 4.3 present the forecast plots produced by the Facebook Prophet model for selected products in the test set. Each plot illustrates the historical monthly sales (black dots), the model's fitted forecast (blue line), and the 95% confidence interval (shaded region). Across the products, the model captures general trend directions but struggles to precisely follow observed fluctuations. In several cases, such as Plot 3 and Plot 5, the predicted values significantly diverge from actual sales points, particularly around peaks and dips—suggesting that the model was not sufficiently responsive to rapid changes in demand. The wide prediction intervals further highlight Prophet's uncertainty, reflecting its cautious estimation in the presence of irregular patterns or limited seasonal repetition at the monthly level. Overall, while the model provides a smooth and interpretable trend projection, the visual evidence reinforces earlier performance metrics indicating that the Prophet forecasts lacked the granularity and accuracy needed for high stakes forecasting at the product level.

4.5 Comparative Analysis of Models

4.5.1 Evaluation Metrics Summary

The comparative analysis of forecasting models presented in Section 4.6.1 evaluates the performance of four distinct models—Pooled OLS Regression (Baseline), Random Forest, XGBoost, and Facebook Prophet—in predicting product sales. The evaluation is based on a comprehensive set of metrics, including Average Mean Absolute Error (Avg MAE), Average Root Mean Square Error (Avg RMSE), R^2, and Mean Absolute Percentage Error (MAPE). These metrics provide a robust framework for assessing the models' predictive accuracy and explanatory power, with the Baseline model serving as a reference point for comparison. The results indicate varying levels of performance across the models, highlighting the strengths and limitations of each approach in the context of sales forecasting.

Model	Avg MAE	Avg RMSE	R^2	MAPE (%)
Baseline	1320.27	3143.53	0.209	38.55%
Random Forest	**439.157431**	**529.413521**	**0.651**	35.65%
XGBoost	472.756898	577.296013	0.615	35.96%
Facebook Prophet	455.944728	615.988553	-	**24.55%**

Table 4.2: Evaluation Metrics Table for All models

Among the models evaluated, **Random Forest and XGBoost clearly outperform the Baseline model across all error metrics**. Random Forest achieves the lowest average MAE (**439.16**) and RMSE (**529.41**), demonstrating its strong ability to capture sales patterns across products and time while minimizing forecast errors. XGBoost also performs competitively, with an average MAE of **472.76** and RMSE of **577.30**, confirming its effectiveness in producing accurate sales forecasts in a retail context.

The Baseline (Pooled OLS) model records substantially higher errors, with an average MAE of **1320.27** and RMSE of **3143.53**, reflecting its limitations in handling the nonlinearities and product-level variability present in real-world sales data.

Facebook Prophet offers a balanced performance, with an average MAE of **455.94** and RMSE of **615.99**, while achieving the **lowest MAPE (24.55%)** among all models, highlighting its strength in capturing trend and seasonality when evaluated with scale-independent metrics.

Overall, these results underscore the **superiority of tree-based machine learning models, particularly Random Forest, in delivering robust and accurate forecasts** for SKU-level retail sales. At the same time, they demonstrate the practical value of models like Prophet for interpretable and seasonality-aware forecasting and emphasize the importance of using multiple evaluation metrics to comprehensively assess model performance.

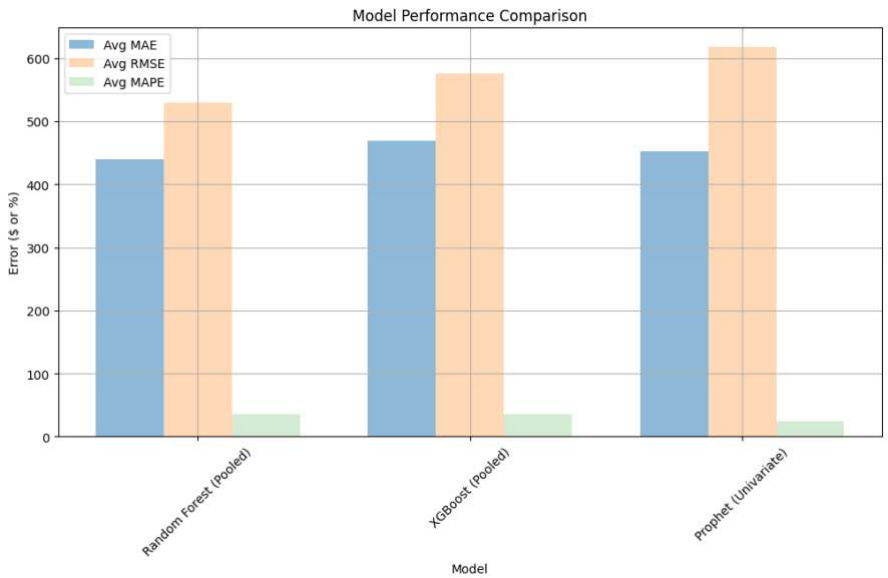

Figure 4.4: ML Model Evaluation Metrics Comparison

4.5.2 Feature Importance Comparison

Feature importance plots (Figures 4.5 and 4.6) reveal differences in how the two models prioritize input variables. In the Random Forest model, Lagged Sales dominate the prediction, accounting for nearly 80% of the total feature importance. Other predictors such as Order Item Discount and Order Item Product Price contribute minimally. In contrast, the XGBoost model distributes importance more evenly: Lagged Sales still remain the most influential feature (≈49%), but Product Price, Discount, and time-based features such as month_sin and month_cos also receive noticeable weight. This suggests that XGBoost may be better at leveraging a broader set of inputs, although Random Forest produces slightly higher accuracy in this case.

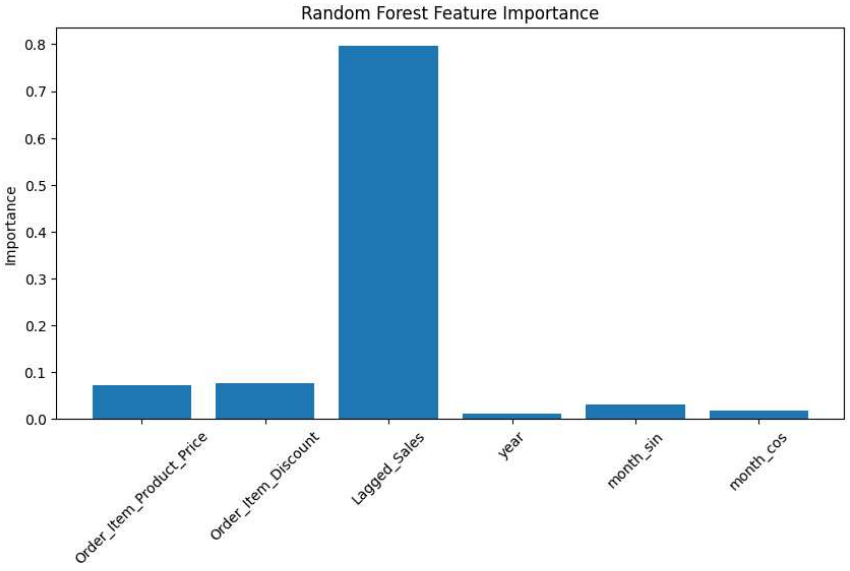

Figure 4.5: Feature Importance Graph for Random Forest

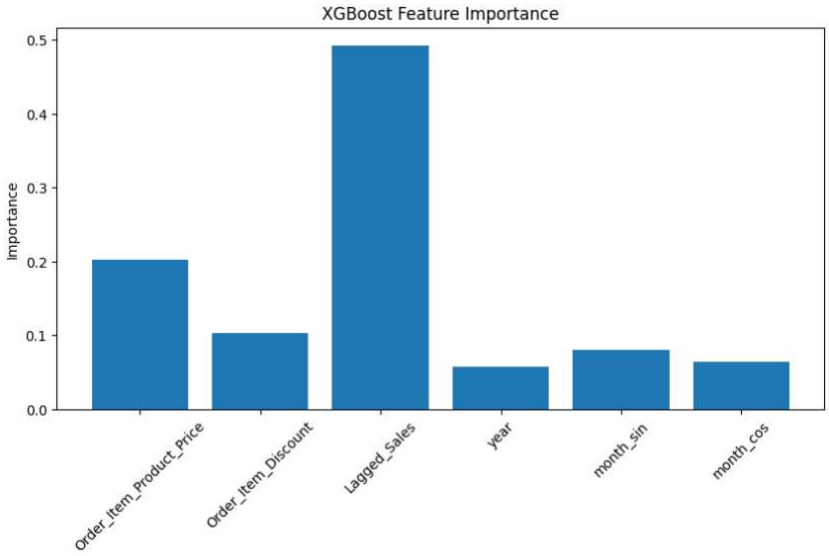

Figure 4.6: Feature Importance Graph for XGBoost

4.5.3 Performance Breakdown by Product

Table 4.3 presents a breakdown of model performance by product across three metrics: R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). The results highlight performance differences between the Random Forest and XGBoost models on a per-product basis. Across most products, Random Forest outperforms XGBoost in terms of both MAE and RMSE. For instance, in the case of the adidas Men's F10 Messi TRX FG Soccer Cleat, the Random Forest achieves a lower MAE (427.87) and RMSE (435.35) compared to XGBoost (564.06 and 503.72, respectively). This pattern is consistent across several other products such as Under Armour Women's Ignite Slide and adidas Kid's F5 Messi FG Soccer Cleat.While the differences in R-squared are less pronounced, Random Forest generally provides better explanatory power for product-level variation. XGBoost underperforms significantly for certain products, with negative R-squared values observed (e.g., adidas Youth Germany Black/Red Away Match Soc), indicating that it fails to capture any meaningful variance for that SKU.

These results emphasize that while both models can capture overall trends in sales, Random Forest demonstrates greater robustness at the product level, particularly in minimizing absolute and squared prediction errors. XGBoost's performance may have been hindered by its sensitivity to noise and extreme values, which are common in the sales data of low-volume or volatile products.

Given these findings, the Random Forest model is better suited for granular, per-product sales forecasting in this context.

Performance Breakdown by Product

Product Name	R-Squared		MAE		RMSE	
	Random Forest	XGBoost	Random Forest	XGBoost	Random Forest	XGBoost
adidas Men's F10 Messi TRX FG Soccer Cleat	0.320959	0.148526	427.868464	564.062117	435.347877	503.719809
Under Armour Women's Micro G Skulpt Running S	0.202685	0.176604	422.159194	535.499343	433.688662	526.950032
Under Armour Women's Ignite Slide	0.319021	0.34152	300.833372	391.973472	293.879969	398.613666
adidas Kids' F5 Messi FG Soccer Cleat	0.219073	0.250444	370.607997	427.49652	361.687952	436.350651

Table 4.3 Result of Tree Based models for Sample Products

The sales forecast for the adidas Men's F10 Messi TRX FG Soccer Cleat reveals that both Random Forest (RF) and XGBoost models capture the general seasonality and trends but consistently underestimate the actual sales, particularly during peak months. The RF model (blue dashed line) performs slightly better than XGBoost (red dashed line), staying closer to the actual sales (black line) and reacting more sharply to spikes and drops. XGBoost tends to overly smooth the series, missing some of the higher sales peaks and failing to recover quickly after dips. This is consistent with the RMSE and MAE metrics from the performance breakdown, where RF demonstrates lower errors for this product.

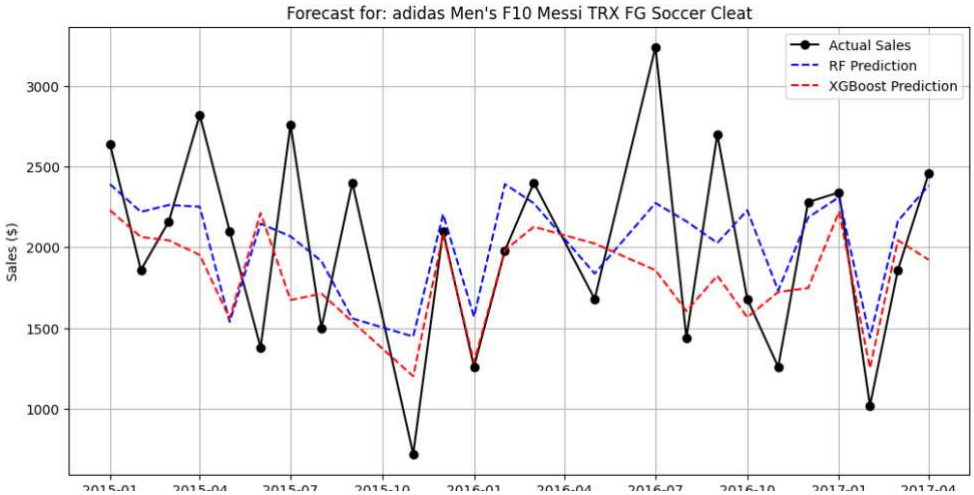

Figure 4.7: RF and XGBoost Comparison in Forecasting Sample Products Sales

For the adidas Youth Germany Black/Red Away Match Soc, the predictive performance of both models deteriorates compared to the previous product. The actual sales are highly volatile, with sharp fluctuations that neither model captures effectively. Random Forest provides a more stable approximation, tracking broad patterns, but fails to capture extreme peaks and valleys.

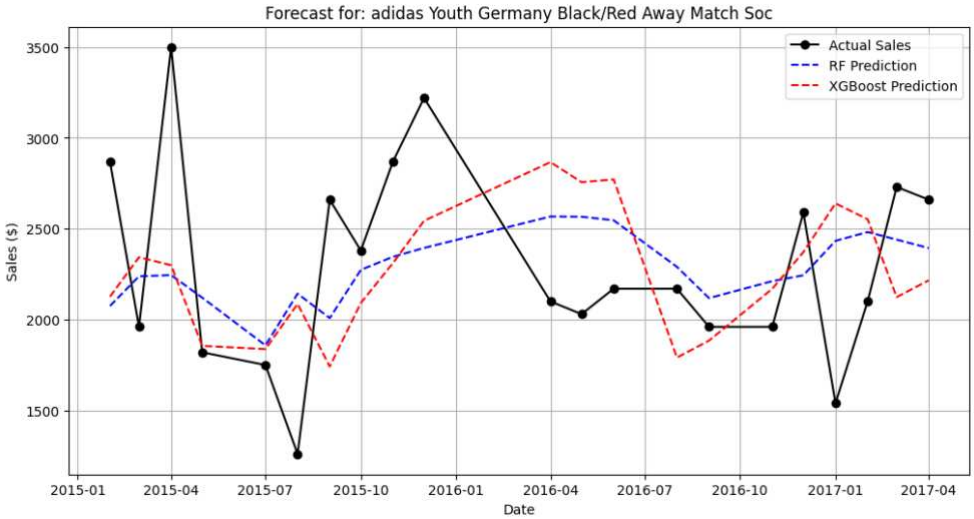

Figure 4.8: RF and XGBoost Comparison in Forecasting Sample Products Sales

XGBoost again produces a smoother line, which oversimplifies the variability in the actual sales. The R-squared values reflect this struggle, showing poor explanatory power (even negative for this product), indicating that the models do not generalize well under high sales volatility.

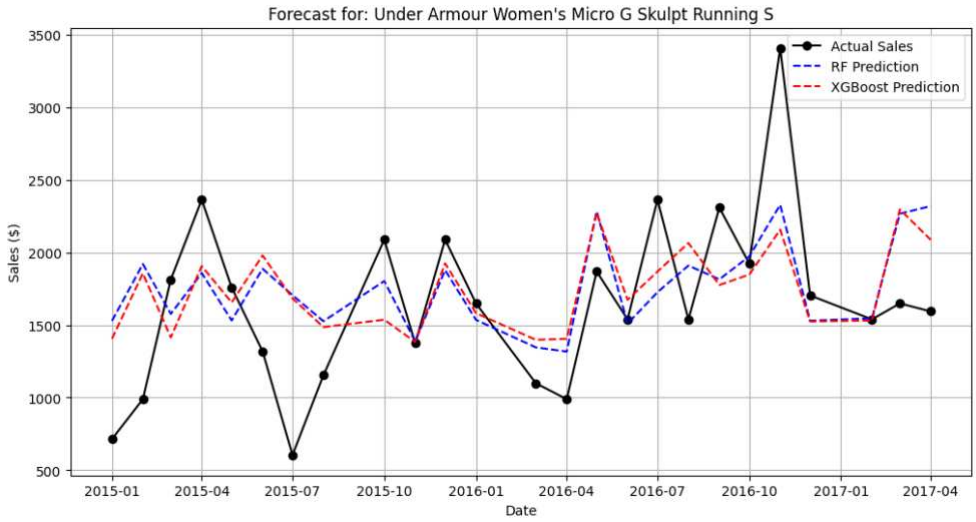

Figure 4.9: RF and XGBoost Comparison in Forecasting Sample Products Sales

The forecasting results for the Under Armour Women's Micro G Skulpt Running S product show a relatively better alignment between predicted and actual sales. The Random Forest model again performs better, closely following the shape and amplitude of the sales curve, including most upward and downward shifts. XGBoost captures the general direction but tends to flatten sharp transitions, particularly underestimating sudden sales surges. Overall, while neither model is perfect, Random Forest displays a stronger ability to model the dynamics of this product, which is also confirmed by its lower MAE and RMSE values for this item.

5

Chapter 5 Discussion

5.1 Insights from Models

5.2 Implications for Practice

5.3 Limitations

5.4 Future Directions

This chapter integrates the key findings of the results and discusses their importance in modeling, practical, and methodological aspects. This chapter also extends a comparative assessment of three machine learning methods, namely Random Forest, XGBoost and Prophet, versus a statistical reference (Pooled OLS) for predicting monthly product-level sales with panel data. The chapter is divided into four sections: key insights from model performance, practical implications for business applications, economic interpretations, limitations of the study, and suggestions for future research.

5.1 Insights from Models

A clear difference in predictive performance interpretability, and robustness was shown after evaluating the forecasting models in this study. Each of these characteristics was analyzed for their potential to impact the use of the models in real-world supply chain forecasting.

Random Forest appeared in this study as the most accurate and reliable model. It recorded the lowest RMSE and MAE, the highest R^2 (0.651), and thereby outperformed all other models over the entire set of 86 SKUs. The ensemble mechanism of Random Forest—which unites hundreds of decision trees through bagging—helped it to be more efficient, even in the case of irregular promotions, sparse demand, and varied seasonal intensity. Significantly, its forecasts did not contain any bias, which was indicated by the residual symmetry around zero.

These findings align with prior studies (e.g., Panda & Mohanty, 2023; Carbonneau et al., 2008), where Random Forest was found to be the most appropriate technique for noisy and heterogeneous time series in retail. Its function of pointing out the features importance (e.g., lagged sales, calendar month, discount) leads to interpretability without losing accuracy, a condition that very few black-box models offer.

XGBoost performed well and came second to Random Forest in most metrics, thus it is only marginally behind Random Forest. A gradient boosting framework of its allowed for targeted learning of residual errors, thus accurate predictions were generated for SKUs with consistent seasonal structure and well-populated historical data. However, its results were more scattered across different products than those of Random Forest, especially when dealing with abrupt promotional events or products with low sales volume.

While the literature (e.g., Feizabadi, 2022; Ensafi et al., 2022) confirms XGBoost's strength in modeling structured, multivariate time series, this study found that its performance is highly dependent on parameter tuning and data stability. Additionally, interpretability was limited to gain-based feature importance scores, which do not easily translate into decision logic.

Nonetheless, XGBoost remains a top-tier choice when accuracy is paramount and technical capacity is available for careful tuning and validation.

Prophet performed moderately across most metrics, offering competitive MAPE values but trailing in RMSE and R^2. It was particularly effective for products with clear seasonality and relatively stable demand patterns, due to its modular decomposition of trend, seasonality, and holiday components. However, Prophet struggled to adapt to sudden promotional surges and non-periodic anomalies, which are common in real-world retail datasets.

This result also is characteristic of experiments conducted by Raiyani et al. (2021) and Ensafi et al. (2022), where Prophet represented a solid compromise between interpretability and accuracy, but it lacked the necessary flexibility in the SKU-level promotions and volatility. The fact that it has a fixed seasonality structure and the manual inclusion of holidays make it an excellent exploratory tool, but at the same time, it is not the best candidate for full-scale operational forecasting in dynamic categories.

The Pooled Ordinary Least Squares statistical baseline showed the lowest predictive power. Although it gave statistically significant coefficients (e.g. the positive impact of the discount, the mean-reverting character of the lagged sales), it did not reflect the nonlinearities or the interaction effects. R^2 was only 0.209, thus indicating its limited capacity to elucidate sales fluctuation in this panel situation.

However, the use of ordinary least squares was essential for comparison. It showed that linear correlations by themselves cannot fully explain the sales patterns at the SKU level in contemporary retail that are still consistent with the need for flexible, nonparametric approaches in these situations.

5.2 Implications for Practice

The results of this study hold multiple implications for forecasting practitioners, supply chain managers, and retail decision-makers who are tasked with selecting, deploying, and maintaining forecasting models.

1. ML Forecasting Enhances Supply Chain Responsiveness

The fact that machine learning models—particularly Random Forest—have consistently outperformed shows that ML-based forecasting has the potential to significantly increase service levels, stock availability, and inventory turnover. Accurate forecasts enable companies to lower safety stock without raising risk, thus releasing working capital and reducing storage costs. For example, the case study by Feizabadi (2022) reported tangible benefits such as

shortened cash conversion cycles—an implication directly supported by the accuracy improvements seen here.

On a practical level, better RMSE means not only less over-orders but also fewer lost sales events. If this is scaled up to hundreds or thousands of SKUs, the financial impact becomes very significant—especially in categories of high-margin or high-volume.

2. Model Selection Should Be Context-Specific

Various models are more appropriate for different forecasting situations. Firms with:

- Highly seasonal, stable product categories may benefit from Prophet due to its interpretability and minimal tuning needs.
- Complex, promotion-sensitive categories should consider Random Forest, which balances flexibility with robustness.
- Technical forecasting teams can leverage XGBoost but must account for its tuning complexity and sensitivity to input distributions.

This thus emphasizes that forecasting is not a simple task where one size fits all. Retailers must match model selection to data availability, forecasting horizon, product behavior, and internal analytical capacity.

3. Forecasting Systems Require Data Maturity

Machine learning forecasting success depends a lot on data structuring and preprocessing done correctly. In this study, steps such as log transformation, differencing, zero-padding, and feature engineering (e.g., lagged variables, calendar features) were essential to model performance. Without them, even the most advanced ML model would have underperformed.

Consequently, organizations need to be aware and recognize that in order to achieve success in utilizing ML forecasting tools, it is necessary for them to undertake investments into the quality of data, ensuring temporal consistency, and establishing automated feature pipelines. The level of data maturity is directly proportional to the extent of success that model.

5.3 Limitations

While this study was carefully designed, it is not without limitations. These include both technical constraints and scope-related boundaries that may affect generalizability.

1. Temporal Granularity

The data set was on monthly sales. This frequency was good for modeling the general seasonality and filtering out noise, but it also restricts the capability to identify short-term factors, like weekly promotional sensitivity or inventory delays. Further research with daily or weekly data could provide more detailed findings.

2. Model Scope

The study concentrated on Random Forest, XGBoost, Prophet, and Pooled OLS. Although these are very popular, the study didn't consider other ML methods, such as LightGBM, CatBoost, or LSTM networks. This was an explicit decision to maintain the same conditions throughout the experiments, but it does restrict the scope of the algorithmic comparison.

3. No Cost-Based or Profit-Based Forecast Evaluation

Assessment was made using statistical error metrics (RMSE, MAE, MAPE, R^2). They are conventional but do not reflect the economic implications of prediction inaccuracies (e.g., the cost of overstock vs. the cost of lost sales). A performance evaluation that takes into account the cost aspect would improve the relevance of the study in practice.

4. Interpretability Analysis Was Limited

Though feature importance was mentioned, no advanced explainability tools were used. Methods such as SHAP analysis might have helped clarify the influence of individual features and gaining confidence in the model's results.

5.4 Future Directions

This study opens multiple avenues for further exploration in both academic and applied contexts.

1. Expanding the Model Library

Future studies definitely need to consider more ML models including LightGBM, CatBoost, or deep learning architectures (LSTM, Transformer-based models) if they want to establish a baseline of modern techniques' performance for various sales granularities and market conditions.

2. Hybrid and Ensemble Strategies

The combination of multiple models' strengths, such as employing Prophet for long-term trend estimation and XGBoost for short-term revisions, may result in improved performance over any individual model. Investigating model stacking or weighted ensemble methods would be a reasonable follow-up action.

3. Incorporating Business Constraints and Cost Structures

Integrating forecasting with downstream applications—such as inventory optimization, pricing, or logistics planning—can also result in end-to-end decision systems. In this situation, the models have to be tested not only on statistical metrics but on operational and financial outcomes as well.

4. Explainable AI in Forecasting

Future work should prioritize transparency in forecasting. Tools like SHAP, LIME, or surrogate models can help demystify black-box predictions, making ML models more accessible to decision-makers and regulators alike.

5. Forecasting Hierarchies and Reconciliation

To sum up, additional research could also use this structure for hierarchical forecasting—where a single stock keeping unit (SKU)-level, category-level, and regional forecasts are made at the same time and then unified. This would be a more accurate representation of how companies at various levels of aggregation.

References

Aviv, Y. (2003). A time-series framework for supply-chain inventory management. Operations Research, 51(2). https://doi.org/10.1287/opre.51.2.210.12780

Bakker, J., & Pechenizkiy, M. (n.d.-a). Food Wholesales Prediction: What Is Your Baseline?

Baltagi, B. H. (2014). Panel Data and Difference-in-Differences Estimation. In Encyclopedia of Health Economics (pp. 425–433). Elsevier. https://doi.org/10.1016/B978-0-12-375678-7.00720-3

Baryannis, G., Dani, S., & Antoniou, G. (2019). Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. Future Generation Computer Systems, 101, 993–1004. https://doi.org/10.1016/j.future.2019.07.059

Boone, T., Ganeshan, R., Jain, A., & Sanders, N. R. (2019). Forecasting sales in the supply chain: Consumer analytics in the big data era. International Journal of Forecasting, 35(1), 170–180. https://doi.org/10.1016/j.ijforecast.2018.09.003

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research, 184(3), 1140–1154. https://doi.org/10.1016/j.ejor.2006.12.004

Chan, F. T. S., Samvedi, A., & Chung, S. H. (2015). Fuzzy time series forecasting for supply chain disruptions. Industrial Management and Data Systems, 115(3), 419–435. https://doi.org/10.1108/IMDS-07-2014-0199

Chen, T., Yin, H., Chen, H., Wu, L., Wang, H., Zhou, X., & Li, X. (2018). TADA: Trend Alignment with Dual-Attention Multi-task Recurrent Neural Networks for Sales Prediction. Proceedings - IEEE International Conference on Data Mining, ICDM, 2018-November, 49–58. https://doi.org/10.1109/ICDM.2018.00020

Doganis, P., Aggelogiannaki, E., & Sarimveis, H. (2008). A combined model predictive control and time series forecasting framework for production-inventory systems. International Journal of Production Research, 46(24). https://doi.org/10.1080/00207540701523058ï

Douaioui, K., Oucheikh, R., Benmoussa, O., & Mabrouki, C. (2024). Machine Learning and Deep Learning Models for Demand Forecasting in Supply Chain Management: A Critical Review. In Applied System Innovation (Vol. 7, Issue 5). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/asi7050093

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022). Time-series forecasting of seasonal items sales using machine learning – A comparative analysis. International Journal of Information Management Data Insights, 2(1), 100058. https://doi.org/10.1016/J.JJIMEI.2022.100058

Feizabadi, J. (2022). Machine learning demand forecasting and supply chain performance. International Journal of Logistics Research and Applications, 25(2), 119–142. https://doi.org/10.1080/13675567.2020.1803246

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. In Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences (Vol. 379, Issue 2194). Royal Society Publishing. https://doi.org/10.1098/rsta.2020.0209

Maçaira, P. M., Tavares Thomé, A. M., Cyrino Oliveira, F. L., & Carvalho Ferrer, A. L. (2018). Time series analysis with explanatory variables: A systematic literature review. In Environmental Modeling and Software (Vol. 107, pp. 199–209). Elsevier Ltd. https://doi.org/10.1016/j.envsoft.2018.06.004

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS ONE, 13(3). https://doi.org/10.1371/journal.pone.0194889

Mircetic, D., Rostami-Tabar, B., Nikolicic, S., & Maslaric, M. (2022). Forecasting hierarchical time series in supply chains: an empirical investigation. International Journal of Production Research, 60(8), 2514–2533. https://doi.org/10.1080/00207543.2021.1896817

Pacella, M., & Papadia, G. (2021). Evaluation of deep learning with long short-term memory networks for time series forecasting in supply chain management. Procedia CIRP, 99, 604–609. https://doi.org/10.1016/j.procir.2021.03.081

Panda, S. K., & Mohanty, S. N. (2023). Time Series Forecasting and Modeling of Food Demand Supply Chain Based on Regressors Analysis. IEEE Access, 11, 42679–42700. https://doi.org/10.1109/ACCESS.2023.3266275

Perera, H. N., Hurley, J., Fahimnia, B., & Reisi, M. (2019). The human factor in supply chain forecasting: A systematic review. European Journal of Operational Research, 274(2), 574–600. https://doi.org/10.1016/j.ejor.2018.10.028

Praveen, U., Farnaz, G., & Hatim, G. (2019). Inventory management and cost reduction of supply chain processes using AI based time-series forecasting and ANN modeling. Procedia Manufacturing, 38, 256–263. https://doi.org/10.1016/j.promfg.2020.01.034

Putra, R. D. (n.d.-a). Forecasting Sales and Inventory in Supply Chain using Machine Learning Methods MSc Research Project MSc in Data Analytics.

Raiyani, A., Lathigara, A., & Mehta, H. (2021). Usage of time series forecasting model in Supply chain sales prediction. IOP Conference Series: Materials Science and Engineering, 1042(1), 012022. https://doi.org/10.1088/1757-899x/1042/1/012022

Seyedan, M., & Mafakheri, F. (2020). Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. Journal of Big Data, 7(1). https://doi.org/10.1186/s40537-020-00329-2

Shih, H., & Rajendran, S. (2019). Comparison of Time Series Methods and Machine Learning Algorithms for Forecasting Taiwan Blood Services Foundation's Blood Supply. Journal of Healthcare Engineering, 2019. https://doi.org/10.1155/2019/6123745

Sun, J., Zhou, S., Zhan, X., & Wu, J. (2024). Enhancing Supply Chain Efficiency with Time Series Analysis and Deep Learning Techniques. https://doi.org/10.20944/preprints202409.0983.v1

Syntetos, A. A., Babai, Z., Boylan, J. E., Kolassa, S., & Nikolopoulos, K. (2016). Supply chain forecasting: Theory, practice, their gap and the future. In European Journal of Operational Research (Vol. 252, Issue 1, pp. 1–26). Elsevier. https://doi.org/10.1016/j.ejor.2015.11.010

Trusov, M., Bodapati, A. v., & Cooper, L. G. (2006). Retailer promotion planning: Improving forecast accuracy and interpretability. Journal of Interactive Marketing, 20(3–4), 71–81. https://doi.org/10.1002/dir.20068

Wang, G., Gunasekaran, A., Ngai, E. W. T., & Papadopoulos, T. (2016). Big data analytics in logistics and supply chain management: Certain investigations for research and applications. In International Journal of Production Economics (Vol. 176, pp. 98–110). Elsevier B.V. https://doi.org/10.1016/j.ijpe.2016.03.014

Weiss, S. M., & Indurkhya, N. (n.d.). LNAI 5212 - Estimating Sales Opportunity Using Similarity-Based Methods.

York, N., & Chopra, S. (2019). Supply Chain Management Strategy, Planning, and OPeratiOn. https://lccn.loc.gov/2017035661

Appendix

Appendix A

Category	Variables	Type	Unit	Description	Values
Financial Information	Sales	Numerical	USD		From 9.99 to 1999.99
Financial Information	Type	Categorical	-	Type of transaction made	4 unique values including: Cash, Debit, Payment, Transfer
Financial Information	Benefit per order	Numerical	USD	Value in sales	From -4274.98 to 911.8
Financial Information	Sales per customer	Numerical	USD	Total sales per customer made per customer	From 7.49 to 1939.99
Shipping Information	Days for shipping (real)	Numerical	Days	Actual shipping days of the purchased product	From 0 to 6
Shipping Information	Days for shipment (scheduled)	Numerical	Days	Days of scheduled delivery of the purchased product	From 0 to 4
Shipping Information	Delivery Status	Categorical	-	Delivery status of orders	4 unique values including: Advance shipping , Late delivery, Shipping on time, Shipping canceled
Shipping Information	Late_delivery_risk	Categorical	-	Indicates if sending is late (1), it is not late (0).	2 unique values including: 0 and 1
Shipping Information	shipping date (DateOrders)	Date/Time	Date	Exact date and time of shipment	From 01/01/2016 00:22 to 9/30/2017 9:
Shipping Information	Shipping Mode	Categorical	-	Type of shipping service used	4 unique values including: Standard Class, First Class, Second Class, Same Day
Category Information	Category Id	Identifier	-	Unique identifier for product categories	51 unique values including: 73 17 29 24 13 12 9 41 37 38 44 3 18 43 65 62 64 4 2 26 40 76 66 70 6 11 16 36 10 63 5 60 45 59 67 61 68 7 72 69 71 75 35 34 33 32 31 30 74 48 46
Category Information	Category Name	Categorical	-	Description of the product category	50 unique values including "Sporting Goods", "Cleats", "Shop

					By Sport", "Women's Apparel", "Electronics", "Boxing & MMA", "Cardio Equipment", "Trade-In", "Kids' Golf Clubs", "Hunting & Shooting", "Baseball & Softball", "Men's Footwear", "Camping & Hiking", "Consumer Electronics", "Cameras", "Computers", "Basketball", "Soccer", "Girls' Apparel", "Accessories", "Women's Clothing", "Crafts", "Men's Clothing", "Tennis & Racquet", "Fitness Accessories", "As Seen on TV!", "Golf Balls", "Strength Training", "Children's Clothing", "Lacrosse", "Baby", "Fishing", "Books", "DVDs", "CDs", "Garden", "Hockey", "Pet Supplies", "Health and Beauty", "Music", "Video Games", "Golf Gloves", "Golf Bags & Carts", "Golf Shoes", "Golf Apparel", "Women's Golf Clubs", "Men's Golf Clubs", "Toys", "Water Sports", "Indoor/Outdoor Games"
Customer Information	Customer City	Categorical	-	City where the customer made the purchase	344 unique values included in the appendix
Customer Information	Customer Country	Categorical	-	Country where the customer made the purchase	2 unique values including: "Puerto Rico" "EE. UU."
Customer Information	Customer Email	Protected	-	Customer's email	Protected due to Legal Reasons
Customer Information	Customer Fname	Categorical	-	Customer name	These values wont be added to the analysis to keep the privacy of customer and integrity

Customer Information	Customer Id	Identifier	-	Unique customer identification number	20652 unique values included in the appendix
Customer Information	Customer Lname	Categorical	-	Last name of the customer	These values wont be added to the analysis to keep the privacy of customer and integrity
Customer Information	Customer Password	Protected	-	Masked customer key	Protected due to Legal Reasons
Customer Information	Customer Segment	Categorical	-	Types of Customers	3 unique values including: Consumer, Home Office, Corporate
Customer Information	Customer State	Categorical	-	State to which the store where the purchase is registered belongs	46 unique values including: "PR", "CA", "NY", "FL", "MA", "IL", "MT", "PA", "MI", "TX", "DE", "GA", "MD", "OH", "HI", "NJ", "WI", "AZ", "CO", "MN", "NC", "NM", "OR", "SC", "VA", "UT", "WA", "KY", "WV", "RI", "CT", "LA", "TN", "DC", "ND", "MO", "IN", "ID", "NV", "KS", "AR", "OK", "AL", "IA", "95758", "91732"
Customer Information	Customer Street	Categorical	-	Street to which the store where the purchase is registered belongs	7458 unique values included in the appendix
Customer Information	Customer Zipcode	Categorical	-	Customer Zipcode	996 unique values included in the appendix
Department Information	Department Id	Identifier	-	Department code of store	11 unique values including: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Department Information	Department Name	Categorical	-	Department name of store	11 unique values including : Fitness, Apparel, Golf, Footwear, Outdoors, Fan Shop, Technology, Book Shop, Discs Shop, Pet Shop, Health and Beauty.
Market Information	Latitude	Numerical	Coordinate	Latitude corresponding to location of store	from -33.93755 to 48.78193

Market Information	Longitude	Numerical	Coordinate	Longitude corresponding to location of store	From -158.026 to 115.2631
Market Information	Market	Categorical	-	Market to where the order is delivered	5 unique values including: Pacific Asia, USCA, Africa, Europe, LATAM
Order Information	Order City	Categorical	.	Destination city of the order	3597 unique values included in the appendix
Order Information	Order Country	Categorical	.	Destination country of the order	164 unique values included in the appendix
Order Information	Order Customer Id	Identifier	.	Customer order code	20652 unique values included in the appendix
Order Information	order date (DateOrders)	Date/Time	Date	Date on which the order is made	From 01/01/2015 00:00 to 9/30/2017 9:58
Order Information	Order Id	Identifier	-	Order code	65752 unique values included in the appendix
Order Information	Order Item Cardprod Id	Identifier	-	Product code generated through the RFID reader	118 unique values included in the appendix
Order Information	Order Item Discount	Numerical	USD	Order item discount value	From 0 to 500
Order Information	Order Item Discount Rate	Numerical	Percentage	Order item discount percentage	From 0 to 0.25
Order Information	Order Item Id	Identifier	-	Order item code	180519 unique values included in the appendix
Order Information	Order Item Product Price	Numerical	USD	Price of products without discount	From 9.99 to 1999.99
Order Information	Order Item Profit Ratio	Numerical	Percentage	Order Item Profit Ratio	From -2.85 to 0.5
Order Information	Order Item Quantity	Numerical	Integer	Number of products per order	From 1 to 5
Order Information	Order Item Total	Numerical	Integer	Total amount per order	From 7.49 to 1939.99
Order Information	Order Profit Per Order	Numerical	USD	Order Profit Per Order	From -4274.98 to 911.8
Order Information	Order Region	Categorical	-	Region of the world where the order is deliveredAfrica ,Western Europe ,Northern ,	23 unique values including: Southeast Asia, South Asia, Oceania, Eastern Asia, West Asia, West of USA, US Center, West Africa,

				Caribbean , South America ,East Africa ,Southern Europe , East of USA ,Canada ,Southern Africa , Central Asia , Europe , Central America, Eastern Europe , South of USA	Central Africa, North Africa, Western Europe, Northern Europe, Central America, Caribbean, South America, East Africa, Southern Europe, East of USA, Canada, Southern Africa, Central Asia, Eastern Europe, South of USA
Order Information	Order State	Categorical	-	State of the region where the order is delivered	1089 unique values included in the appendix
Order Information	Order Status	Categorical	-	Order status	9 unique values including: complete, pending, closed, pending_payment, canceled, processing, suspected_fraud, on_hold, payment_review
Order Information	Order Zipcode	Categorical	-	Postal code of the order destination	610 unique values included in the appendix
Product Information	Product Card Id	Identifier	-	Product code	118 unique values included in the appendix
Product Information	Product Category Id	Identifier	-	Product category code	51 unique values included in the appendix
Product Information	Product Description	Categorical	-	Description of the product features and details	These values wont be added to the analysis
Product Information	Product Image	Not Variable	-	Link of visit and purchase of the product	These values wont be added to the analysis
Product Information	Product Name	Categorical	-	Name of the product	118 unique values included in the appendix
Product Information	Product Price	Numerical	USD	Price of the product	From 9.99 to 1999.99
Product Information	Product Status	Categorical	-	Availability status of the product	Missing Value

Appendix B

```
import pandas as pd
import numpy as np
from statsmodels.tsa.stattools import adfuller
from statsmodels.regression.linear_model import OLS
import statsmodels.api as sm
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
from linearmodels.panel import PanelOLS, RandomEffects
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
file_path = r"C:\Users\beign\Desktop\Thesis\02.Data\DataCoSupplyChainDataset.csv"
try:
    df = pd.read_csv(file_path, encoding='cp1252')
except FileNotFoundError:
    print(f"Error: The file was not found at {file_path}")
        df = pd.DataFrame()
if not df.empty:
    cols_to_drop = [
        "Type", "Sales per customer", "Delivery Status", "Category Id", "Category Name",
"Customer City",
        "Customer Country", "Customer Email", "Customer Fname", "Customer Id",
        "Customer Lname", "Customer Password", "Customer State", "Customer Street",
"Customer Segment",
        "Customer Zipcode", "Department Id", "Department Name", "Latitude", "Longitude",
"Market",
        "Order City", "Order Customer Id", "Order Id", "Order Item Cardprod Id",
        "Order Item Id", "Order Item Product Price", "Order Item Total", "Order Region", "Order
State", "Order Status",
        "Order Zipcode", "Product Card Id", "Product Category Id", "Product Description",
        "Product Image", "Product Status", "Shipping Mode",
    ]
```

```
df = df.drop(columns=cols_to_drop, errors='ignore')


print("Missing values before dropping:\n", df.isnull().sum())
df = df.dropna()
print("\nDataset shape after dropping NA:", df.shape)
date_column = 'order date (DateOrders)'
try:
    df[date_column] = pd.to_datetime(df[date_column], errors='coerce')
    df['month'] = df[date_column].dt.to_period('M')
    print(f"\nDate column '{date_column}' successfully converted to datetime.")
except Exception as e:
    print(f"Error parsing dates in '{date_column}': {e}")
    raise
df = df[df[date_column] <= '2017-09-30']
print(f'Dataset shape after filtering (up to Sep 2017): {df.shape}")
columns_to_check = ['Sales', 'Order Item Discount', 'Product Price']
print("\nVisualizing potential outliers before removal...")
plt.figure(figsize=(12, 6))
for i, col in enumerate(columns_to_check, 1):
    plt.subplot(1, len(columns_to_check), i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
    plt.ylabel(col)
plt.tight_layout()
plt.show()
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

```
    for col in columns_to_check:
        df = remove_outliers_iqr(df, col)
    print("\nDataset shape after outlier removal:", df.shape)
    dependent_var = 'Sales'
    correlations = df.corr(numeric_only=True)[dependent_var]
    print("\nCorrelations with", dependent_var, ":\n", correlations)
    df.to_csv(r"C:\Users\beign\Desktop\Thesis\02.Data\preprocessed_data.csv", index=False)
    print("\nExported preprocessed DataFrame to 'preprocessed_data.csv'")
    df_aggregated = df.groupby(['Product Name', 'month']).agg({
        'Sales': 'sum',
        'Order Item Discount': 'mean',
        'Product Price': 'mean'
    }).reset_index()
    df_aggregated['prev_month'] = df_aggregated['month'] - 1
    df_aggregated = df_aggregated.merge(
        df_aggregated[['Product Name', 'month', 'Sales']],
        left_on=['Product Name', 'prev_month'],
        right_on=['Product Name', 'month'],
        how='left',
        suffixes=('', '_lagged')
    )
    df_aggregated = df_aggregated.rename(columns={'Sales_lagged': 'lagged_sales'})
    df_aggregated = df_aggregated.drop(columns=['month_lagged', 'prev_month'])
    df_panel = df_aggregated.set_index(['Product Name', 'month'])[['Sales', 'Order Item
Discount', 'Product Price', 'lagged_sales']].dropna()
    df_panel_reset = df_panel.reset_index()
    df_panel_reset.to_csv(r"C:\Users\beign\Desktop\Thesis\02.Data\updated_panel_data.csv",
index=False)
    print("Exported updated panel DataFrame to 'updated_panel_data.csv'")
    total_sales = df.groupby('month')['Sales'].sum()
    def adf_test(series, name):
        try:
```

```
        result = adfuller(series, autolag='AIC', maxlag=1)
        print(f"\nADF Test for {name}:")
        print(f"  ADF Statistic: {result[0]:.4f}")
        print(f"  p-value: {result[1]:.4f}")
        print(f"  Result: {'Stationary' if result[1] < 0.05 else 'Non-stationary'}")
    except ValueError as e:
        print(f"ADF Test for {name} failed: {str(e)}")
adf_test(total_sales, "Entire Dataset Sales (Original)")
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(total_sales.index.astype(str), total_sales.values, marker='o')
plt.title('Total Sales Over Time (Original)')
plt.xticks(rotation=45)
plt.xlabel('Month')
plt.ylabel('Total Sales')
rolling_var = total_sales.rolling(window=3, min_periods=1).var()
plt.subplot(1, 2, 2)
plt.plot(total_sales.index.astype(str), rolling_var, marker='o', color='orange')
plt.title('Variance of Total Sales (Rolling Window)')
plt.xticks(rotation=45)
plt.xlabel('Month')
plt.ylabel('Variance')
plt.tight_layout()
plt.show()
if (total_sales <= 0).any():
    total_sales = total_sales + 1
total_sales_log_diff = np.log(total_sales).diff().dropna()
adf_test(total_sales_log_diff, "Entire Dataset Sales (Log-Transformed and Differenced)")
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(total_sales_log_diff.index.astype(str), total_sales_log_diff.values, marker='o')
plt.title('Log-Transformed and Differenced Sales')
```

```
    plt.xticks(rotation=45)
    plt.xlabel('Month')
    plt.ylabel('Log-Differenced Sales')
    rolling_var_log_diff = total_sales_log_diff.rolling(window=3, min_periods=1).var()
    plt.subplot(1, 2, 2)
    plt.plot(total_sales_log_diff.index.astype(str), rolling_var_log_diff, marker='o',
color='orange')
    plt.title('Variance of Log-Differenced Sales (Rolling Window)')
    plt.xticks(rotation=45)
    plt.xlabel('Month')
    plt.ylabel('Variance')
    plt.tight_layout()
    plt.show()
    if (df_panel['Sales'] <= 0).any() or (df_panel['lagged_sales'] <= 0).any():
        df_panel['Sales'] = df_panel['Sales'] + 1
        df_panel['lagged_sales'] = df_panel['lagged_sales'] + 1
    df_panel['log_sales'] = np.log(df_panel['Sales'])
    df_panel['log_lagged_sales'] = np.log(df_panel['lagged_sales'])
    df_panel['diff_log_sales'] = df_panel.groupby(['Product Name'])['log_sales'].diff()
    df_panel['diff_log_lagged_sales'] = df_panel.groupby(['Product
Name'])['log_lagged_sales'].diff()
    df_panel_transformed = df_panel.dropna(subset=['diff_log_sales',
'diff_log_lagged_sales']).copy()
    group_counts = df_panel_transformed.groupby(['Product Name']).size()
    groups_to_keep = group_counts[group_counts >= 3].index
    df_panel_transformed =
df_panel_transformed[df_panel_transformed.index.droplevel('month').isin(groups_to_keep)]
    df_panel_transformed = df_panel_transformed.reset_index()
    df_panel_transformed['entity'] = df_panel_transformed['Product Name']
    df_panel_transformed['month'] = df_panel_transformed['month'].dt.to_timestamp()
    df_panel_transformed = df_panel_transformed.set_index(['entity', 'month'])
    df_panel_transformed = df_panel_transformed.sort_index()
    y = df_panel_transformed['diff_log_sales']
```

```
    X = df_panel_transformed[['Order Item Discount', 'diff_log_lagged_sales']]
    X = sm.add_constant(X)
    print("\n--- [1] Pooled OLS Model & Diagnostics ---")
    pooled_ols = OLS(y, X).fit(cov_type='HC1')
    print("\nPooled OLS Model Results:\n", pooled_ols.summary())
    bp_test = het_breuschpagan(pooled_ols.resid, X)
    print("\nBreusch-Pagan Test (Heteroskedasticity):")
    print(f"  LM Statistic: {bp_test[0]:.4f}, p-value: {bp_test[1]:.4f}")
    dw_stat = durbin_watson(pooled_ols.resid)
    print(f"\nDurbin-Watson Statistic (Autocorrelation): {dw_stat:.4f}")
    print("\n--- [2] Fixed Effects (Within) Model ---")
    X_fe = df_panel_transformed[['Order Item Discount', 'diff_log_lagged_sales']]
    fe_model = PanelOLS(y, X_fe, entity_effects=True).fit()
    print("\nFixed Effects Model Results:\n", fe_model)
    print("\n--- [3] Random Effects Model ---")
    re_model = RandomEffects(y, X).fit()
    print("\nRandom Effects Model Results:\n", re_model)
    print("\n--- [4] Model Specification Tests ---")
    N = len(df_panel_transformed.index.get_level_values('entity').unique())
    NT = len(df_panel_transformed)
    K = X_fe.shape[1]
    ssr_pooled = np.sum(pooled_ols.resid ** 2)
    ssr_fixed = np.sum(fe_model.resids ** 2)
    if (NT - N - K) > 0 and (N - 1) > 0:
        f_limer_stat = ((ssr_pooled - ssr_fixed) / (N - 1)) / (ssr_fixed / (NT - N - K))
        f_limer_p_value = 1 - stats.f.cdf(f_limer_stat, N - 1, NT - N - K)
        print("\nF-Limer Test (Fixed Effects vs Pooled OLS):")
        print(f"  F-Statistic: {f_limer_stat:.4f}, p-value: {f_limer_p_value:.4f}")
        print(f"  Result: {'Fixed Effects is preferred.' if f_limer_p_value < 0.05 else 'Pooled OLS
may be sufficient.'}")
    else:
        print("\nF-Limer Test could not be computed due to insufficient data.")
```

```
    try:
        hausman_results = fe_model.compare(re_model)
        print("\nHausman Test (Fixed Effects vs Random Effects):")
        print(hausman_results)
    except Exception as e:
        print(f"\nCould not perform Hausman test: {e}")
    print("\n--- [5] Final Model Selection & Robust Errors ---")
    fe_robust_model = PanelOLS(y, X_fe, entity_effects=True).fit(cov_type='clustered',
cluster_entity=True)
    print("\nFixed Effects Model with Clustered (Robust) Standard Errors:\n",
fe_robust_model)
    def print_evaluation_metrics(model, y_true, X_df, model_name="Model"):
        """Calculates and prints key regression metrics."""
        print(f"\n--- Evaluation Metrics for {model_name} ---")
        residuals = model.resid if hasattr(model, 'resid') else model.resids
       avoid AttributeError
        if model.__class__.__name__ == 'RegressionResultsWrapper': # For statsmodels OLS
            r_squared = model.rsquared
            f_stat = model.fvalue
            f_pvalue = model.f_pvalue
            print(f"R-squared:          {r_squared:.4f}")
            print(f"F-statistic:        {f_stat:.4f} (p-value: {f_pvalue:.4f})")
        else:
            r_squared = model.rsquared_within
            f_stat = model.f_statistic.stat
            f_pvalue = model.f_statistic.pvalue
            print(f"R-squared (Within): {r_squared:.4f}")
            print(f"F-statistic:        {f_stat:.4f} (p-value: {f_pvalue:.4f})")
        mae = np.mean(np.abs(residuals))
        rmse = np.sqrt(np.mean(residuals**2))
        y_true_mape = y_true[y_true != 0]
        residuals_mape = residuals.loc[y_true_mape.index]
```

```
        if not y_true_mape.empty:
            mape = np.mean(np.abs(residuals_mape / y_true_mape)) * 100
            print(f"MAPE:              {mape:.4f}%")
        else:
            print("MAPE:              Not computable (y_true contains only zeros)")
        print(f"MAE:               {mae:.4f}")
        print(f"RMSE:              {rmse:.4f}")
        naive_mae_denominator = y_true.groupby(level='entity').diff().abs().mean()
        if naive_mae_denominator > 0:
            mase = mae / naive_mae_denominator
            print(f"MASE:              {mase:.4f}")
        else:
            print("MASE:              Not computable (naive model has zero error)")
    print_evaluation_metrics(pooled_ols, y, X, model_name="Pooled OLS Model")
print("\n--- [6] Raw Sales Pooled OLS Model ---")
df_panel_raw = df_panel.reset_index()
df_panel_raw_clean = df_panel_raw.dropna(subset=['Sales', 'Order Item Discount',
'lagged_sales'])
X_raw = df_panel_raw_clean[['Order Item Discount', 'lagged_sales']]
X_raw = sm.add_constant(X_raw)
y_raw = df_panel_raw_clean['Sales']
ols_raw_model = sm.OLS(y_raw, X_raw).fit()
print(ols_raw_model.summary())
y_pred_raw = ols_raw_model.predict(X_raw)
residuals_raw = y_raw - y_pred_raw
mae_raw = np.mean(np.abs(residuals_raw))
rmse_raw = np.sqrt(np.mean(residuals_raw ** 2))
mape_raw = np.mean(np.abs(residuals_raw / y_raw)) * 100
print("\n--- Evaluation Metrics for Raw Sales OLS Model ---")
print(f"MAE:       {mae_raw:.4f}")
print(f"RMSE:      {rmse_raw:.4f}")
print(f"MAPE:      {mape_raw:.2f}%")
```

```
print("\n--- Additional Business Metrics ---")
print(f"MAE (in dollars): ${mae_raw:.2f}")
avg_unit_price = df_panel_raw_clean.groupby(['Product Name', 'month'])['Product Price'].mean()
mean_avg_unit_price = avg_unit_price.mean()
print(f"Average unit sale price per product/month: ${mean_avg_unit_price:.2f}")
total_revenue = df_panel_raw_clean.groupby(['Product Name', 'month'])['Sales'].sum()
mean_revenue_per_month = total_revenue.mean()
print(f"Average revenue per product/month: ${mean_revenue_per_month:.2f}")
```

Appendix C

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
from sklearn.model_selection import TimeSeriesSplit, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from xgboost import XGBRegressor
from statsmodels.tsa.seasonal import seasonal_decompose
warnings.filterwarnings('ignore')
try:
    from prophet import Prophet
    PROPHET_AVAILABLE = True
    print("Prophet library found.")
except ImportError:
    PROPHET_AVAILABLE = False
    print("Warning: Prophet library not found. Skipping Prophet analysis. To install: pip install prophet")

print("\n--- Section 2: Loading and preparing data ---")
# Load the dataset from the specified file path
```

```
file_path = r"C:\Users\beign\Desktop\Thesis\02.Data\DataCoSupplyChainDataset.csv"
try:
    df = pd.read_csv(file_path, encoding='latin1')
    print("Dataset loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at '{file_path}'. Please ensure the path is correct.")
    exit()
df.rename(columns={
    'Days for shipping (real)': 'Real_Days', 'Days for shipment (scheduled)': 'Scheduled_Days',
    'Order Item Discount': 'Order_Item_Discount', 'Order Item Product Price':
'Order_Item_Product_Price',
    'Product Name': 'Product_Name', 'Order Country': 'Order_Country',
    'shipping date (DateOrders)': 'Shipping_Date'}, inplace=True)
df['Shipping_Date'] = pd.to_datetime(df['Shipping_Date'])
df = df[df['Shipping_Date'] < '2017-09-01']
df = df.sort_values(['Product_Name', 'Shipping_Date'])
df['Lagged_Sales'] = df.groupby(['Product_Name'])['Sales'].shift(1)
df = df.dropna(subset=['Lagged_Sales'])
df['Month'] = df['Shipping_Date'].dt.to_period('M')
panel_df = df.groupby(['Product_Name', 'Month']).agg({
    'Order_Item_Discount': 'mean',
    'Order_Item_Product_Price': 'mean',
    'Lagged_Sales': 'mean',
    'Sales': 'sum'}).reset_index()
def remove_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

```
for col in ['Sales', 'Order_Item_Discount', 'Order_Item_Product_Price']:
    panel_df = remove_outliers(panel_df, col)

# Plot 1
monthly_summary = panel_df.copy()
monthly_summary['Month'] = monthly_summary['Month'].dt.to_timestamp()
monthly_summary = monthly_summary.groupby('Month')['Sales'].sum()

plt.figure(figsize=(15, 6))
plt.plot(monthly_summary.index, monthly_summary.values, marker='o', linestyle='-')
plt.title('Total Monthly Sales Over Time (Visual Check for Seasonality)')
plt.xlabel('Date')
plt.ylabel('Total Sales ($)')
plt.grid(True)
plt.show()

# Plot 2
decomposed_df = monthly_summary.to_frame().asfreq('MS')
decomposition = seasonal_decompose(decomposed_df['Sales'], model='multiplicative',
period=12)
fig = decomposition.plot()
fig.set_size_inches(14, 8)
plt.suptitle('Statistical Seasonal Decomposition', y=1.02)
plt.show()

panel_df['Month_dt'] = panel_df['Month'].dt.to_timestamp()
panel_df['year'] = panel_df['Month_dt'].dt.year
panel_df['month_of_year'] = panel_df['Month_dt'].dt.month
panel_df['month_sin'] = np.sin(2 * np.pi * panel_df['month_of_year'] / 12)
panel_df['month_cos'] = np.cos(2 * np.pi * panel_df['month_of_year'] / 12)
print("Feature engineering complete.")
features = [
```

```
    'Order_Item_Product_Price', 'Order_Item_Discount', 'Lagged_Sales',
    'year', 'month_sin', 'month_cos'
]
target = 'Sales'
tscv = TimeSeriesSplit(n_splits=5)
train_indices, test_indices = list(tscv.split(panel_df))[-1]
train = panel_df.iloc[train_indices]
test = panel_df.iloc[test_indices]

X_train = train[features]
y_train = train[target]
X_test = test[features]
y_test = test[target]

param_dist_rf = {
    'n_estimators': [100, 200, 300, 400],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [2, 4, 6]
}

param_dist_xgb = {
    'n_estimators': [100, 200, 300, 400],
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [5, 7, 9],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0]
}

print("\nTraining Random Forest...")
random_rf = RandomizedSearchCV(
    estimator=RandomForestRegressor(random_state=42),
```

```
    param_distributions=param_dist_rf,
    n_iter=50, cv=tscv, n_jobs=-1, random_state=42, verbose=1
)
random_rf.fit(X_train, y_train)
best_rf = random_rf.best_estimator_

print("\nTraining XGBoost...")
random_xgb = RandomizedSearchCV(
    estimator=XGBRegressor(random_state=42, objective='reg:squarederror'),
    param_distributions=param_dist_xgb,
    n_iter=50, cv=tscv, n_jobs=-1, random_state=42, verbose=1
)
random_xgb.fit(X_train, y_train)
best_xgb = random_xgb.best_estimator_
print("\nPooled model training complete.")

def evaluate_pooled_model(model, name, X_test, y_test, params):
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    y_test_mape = y_test.copy()
    y_test_mape[y_test_mape == 0] = 1e-6
    mape = np.mean(np.abs((y_test - y_pred) / y_test_mape)) * 100

    print(f"\n--- Evaluation Results for {name} ---")
    print(f'Best Parameters: {params}")
    print(f'R^2 Score: {r2:.3f}")
    print(f'MAE (Mean Absolute Error): ${mae:.2f}")
    print(f'RMSE (Root Mean Squared Error): ${rmse:.2f}")
    print(f'MAPE (Mean Absolute Percentage Error): {mape:.2f}%")
```

```
    plt.figure(figsize=(15, 5))
    plt.subplot(1, 2, 1)
    plt.scatter(y_test, y_pred, alpha=0.6)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
    plt.title(f'{name}: Actual vs. Forecasted')
    plt.xlabel('Actual Sales ($)'); plt.ylabel('Predicted Sales ($)'); plt.grid(True)

    plt.subplot(1, 2, 2)
    plt.hist(y_test - y_pred, bins=30, edgecolor='k', alpha=0.7)
    plt.title(f'{name}: Distribution of Residuals'); plt.xlabel('Prediction Error ($)');
plt.grid(True)
    plt.show()

    return y_pred, mape

pred_rf, mape_rf = evaluate_pooled_model(best_rf, "Random Forest", X_test, y_test,
random_rf.best_params_)
pred_xgb, mape_xgb = evaluate_pooled_model(best_xgb, "XGBoost", X_test, y_test,
random_xgb.best_params_)

test = test.copy()
test['RF_Pred'] = pred_rf
test['XGB_Pred'] = pred_xgb
test['Month'] = test['Month'].dt.to_timestamp()
test_sorted = test.sort_values('Month')

print("\n--- Performance Breakdown by Product ---")
def calculate_product_metrics(g):
    sales = g['Sales']
    rf_pred = g['RF_Pred']
    xgb_pred = g['XGB_Pred']
    return pd.Series({
```

```
        'R2_RF': r2_score(sales, rf_pred),
        'MAE_RF': mean_absolute_error(sales, rf_pred),
        'RMSE_RF': np.sqrt(mean_squared_error(sales, rf_pred)),
        'R2_XGB': r2_score(sales, xgb_pred),
        'MAE_XGB': mean_absolute_error(sales, xgb_pred),
        'RMSE_XGB': np.sqrt(mean_squared_error(sales, xgb_pred))
    })

product_perf_df = test.groupby('Product_Name').apply(calculate_product_metrics)
top_products_in_test = test.groupby('Product_Name')['Sales'].sum().nlargest(5).index
print("Performance on Top 5 Products in Test Set:")
print(product_perf_df.loc[top_products_in_test])


plt.figure(figsize=(10, 5))
plt.bar(features, best_rf.feature_importances_)
plt.title('Random Forest Feature Importance'); plt.ylabel('Importance'); plt.xticks(rotation=45);
plt.show()


plt.figure(figsize=(10, 5))
plt.bar(features, best_xgb.feature_importances_)
plt.title('XGBoost Feature Importance'); plt.ylabel('Importance'); plt.xticks(rotation=45);
plt.show()


agg_test = test_sorted.groupby('Month')[['Sales', 'RF_Pred', 'XGB_Pred']].sum()
plt.figure(figsize=(15, 7))
plt.plot(agg_test.index, agg_test['Sales'], label='Actual Total Sales', color='black',
linewidth=2.5)
plt.plot(agg_test.index, agg_test['RF_Pred'], label='Random Forest Total Prediction',
color='blue', linestyle='--')
plt.plot(agg_test.index, agg_test['XGB_Pred'], label='XGBoost Total Prediction', color='red',
linestyle='--')
plt.title('Aggregated Forecast vs. Actual Sales'); plt.xlabel('Date'); plt.ylabel('Total Sales ($)')
plt.legend(); plt.grid(True); plt.show()
```

```
top_10_products =
panel_df.groupby('Product_Name')['Sales'].sum().nlargest(10).index.tolist()

print("\n--- Generating Forecast Plots for the Top 10 Products ---")

for product_name in top_10_products:

    single_product_df = test_sorted[test_sorted['Product_Name'] == product_name]

    if single_product_df.empty:

        print(f"Skipping '{product_name}' as it has no data in the test set.")

        continue

    plt.figure(figsize=(12, 6))

    plt.plot(single_product_df['Month'], single_product_df['Sales'], label='Actual Sales',
color='black', marker='o')

    plt.plot(single_product_df['Month'], single_product_df['RF_Pred'], label='RF Prediction',
color='blue', linestyle='--')

    plt.plot(single_product_df['Month'], single_product_df['XGB_Pred'], label='XGBoost
Prediction', color='red', linestyle='--')

    plt.title(f'Forecast for: {product_name}'); plt.xlabel('Date'); plt.ylabel('Sales ($)')

    plt.legend(); plt.grid(True); plt.show()


if PROPHET_AVAILABLE:

    print("\n--- Section 8: Starting Univariate Analysis with Facebook Prophet ---")

    from sklearn.model_selection import TimeSeriesSplit

    prophet_results = []


    for product_name in top_10_products:

        print(f"\nForecasting for: {product_name} with Prophet")

        product_df = panel_df[panel_df['Product_Name'] == product_name].copy()

        prophet_df = product_df[['Month_dt', 'Sales']].rename(columns={'Month_dt': 'ds', 'Sales':
'y'})


        prophet_df = prophet_df.sort_values('ds')

        date_diffs = prophet_df['ds'].diff().dt.days[1:]

        if len(date_diffs) > 0 and not (date_diffs > 27).all() and not (date_diffs < 32).all():
```

```
        print(f"Warning: Irregular or missing dates detected for {product_name}. Prophet may be unreliable.")
        continue

    if len(prophet_df) < 24:
        print(f"Skipping {product_name} due to insufficient data (<24 months).")
        continue

    train_df = prophet_df.iloc[:-6]
    test_df_prophet = prophet_df.iloc[-6:]
    if len(train_df) < 2:
        print(f"Skipping {product_name} due to insufficient training data.")
        continue

    param_grid = {
        'changepoint_prior_scale': [0.01, 0.05, 0.1],
        'seasonality_prior_scale': [0.1, 1.0, 10.0]
    }
    best_params = {'changepoint_prior_scale': 0.05, 'seasonality_prior_scale': 10.0}
    best_mae = float('inf')

    tscv_prophet = TimeSeriesSplit(n_splits=3)
    for cps in param_grid['changepoint_prior_scale']:
        for sps in param_grid['seasonality_prior_scale']:
            mae_scores = []
            for train_idx, val_idx in tscv_prophet.split(train_df):
                train_split = train_df.iloc[train_idx]
                val_split = train_df.iloc[val_idx]
                model = Prophet(
                    yearly_seasonality=True,
                    weekly_seasonality=False,
                    daily_seasonality=False,
```

```
            interval_width=0.95,
            changepoint_prior_scale=cps,
            seasonality_prior_scale=sps
        )
        model.fit(train_split)
        future = model.make_future_dataframe(periods=len(val_split), freq='MS')
        forecast = model.predict(future)
        y_pred = forecast['yhat'][-len(val_split):].values
        y_true = val_split['y'].values
        mae_scores.append(mean_absolute_error(y_true, y_pred))
    avg_mae = np.mean(mae_scores)
    if avg_mae < best_mae:
        best_mae = avg_mae
        best_params = {'changepoint_prior_scale': cps, 'seasonality_prior_scale': sps}

model = Prophet(
    yearly_seasonality=True,
    weekly_seasonality=False,
    daily_seasonality=False,
    interval_width=0.95,
    **best_params
)
model.fit(train_df)
future = model.make_future_dataframe(periods=len(test_df_prophet), freq='MS')
forecast = model.predict(future)

y_pred = forecast['yhat'][-len(test_df_prophet):].values
y_true = test_df_prophet['y'].values

y_lower = forecast['yhat_lower'][-len(test_df_prophet):].values
y_upper = forecast['yhat_upper'][-len(test_df_prophet):].values
```

```
        residuals = y_true - y_pred
        resid_std = np.std(residuals)

        y_true_mape = y_true.copy()
        y_true_mape[y_true_mape == 0] = 1e-6
        mape = np.mean(np.abs((y_true - y_pred) / y_true_mape)) * 100

        prophet_results.append({
            'Product_Name': product_name,
            'MAE': mean_absolute_error(y_true, y_pred),
            'RMSE': np.sqrt(mean_squared_error(y_true, y_pred)),
            'MAPE': mape,
            'Residual_Std': resid_std,
            'Avg_Interval_Width': np.mean(y_upper - y_lower)
        })

        fig, ax = plt.subplots(figsize=(10, 6))
        ax.plot(test_df_prophet['ds'], y_true, label='Actual Sales', color='black', marker='o')
        ax.plot(test_df_prophet['ds'], y_pred, label='Prophet Prediction', color='blue', linestyle='-
-')
        ax.fill_between(test_df_prophet['ds'], y_lower, y_upper, color='blue', alpha=0.1,
label='95% Confidence Interval')
        ax.set_title(f'Prophet Forecast for: {product_name}')
        ax.set_xlabel('Date')
        ax.set_ylabel('Sales ($)')
        ax.legend()
        ax.grid(True)
        plt.show()

        plt.figure(figsize=(10, 6))
        plt.hist(residuals, bins=30, edgecolor='k', alpha=0.7)
        plt.title(f'Prophet Residuals for: {product_name}')
```

```
        plt.xlabel('Prediction Error ($)')
        plt.ylabel('Frequency')
        plt.grid(True)
        plt.show()

    prophet_results_df = pd.DataFrame(prophet_results)
    if prophet_results_df.empty:
        print("No products had sufficient data for Prophet modeling.")
    else:
        print("\nProphet Results Summary:")
        print(prophet_results_df)

valid_top_products = product_perf_df.index.intersection(top_10_products)
print(f"\nComparing models based on the {len(valid_top_products)} top products present in the test set.")

rf_avg_mae = product_perf_df.loc[valid_top_products]['MAE_RF'].mean()
rf_avg_rmse = product_perf_df.loc[valid_top_products]['RMSE_RF'].mean()
xgb_avg_mae = product_perf_df.loc[valid_top_products]['MAE_XGB'].mean()
xgb_avg_rmse = product_perf_df.loc[valid_top_products]['RMSE_XGB'].mean()

summary_data = [
    {'Model': 'Random Forest (Pooled)', 'Avg MAE': rf_avg_mae, 'Avg RMSE': rf_avg_rmse, 'Avg MAPE': mape_rf, 'Avg Residual Std': np.nan, 'Avg Interval Width': np.nan},
    {'Model': 'XGBoost (Pooled)', 'Avg MAE': xgb_avg_mae, 'Avg RMSE': xgb_avg_rmse, 'Avg MAPE': mape_xgb, 'Avg Residual Std': np.nan, 'Avg Interval Width': np.nan}
]

if PROPHET_AVAILABLE and not prophet_results_df.empty:
    prophet_filtered_results = prophet_results_df[prophet_results_df['Product_Name'].isin(valid_top_products)]
    summary_data.append({
        'Model': 'Prophet (Univariate)',
```

```
        'Avg MAE': prophet_filtered_results['MAE'].mean(),
        'Avg RMSE': prophet_filtered_results['RMSE'].mean(),
        'Avg MAPE': prophet_filtered_results['MAPE'].mean(),
        'Avg Residual Std': prophet_filtered_results['Residual_Std'].mean(),
        'Avg Interval Width': prophet_filtered_results['Avg_Interval_Width'].mean()
    })

summary_df = pd.DataFrame(summary_data)
print("\n--- Overall Model Performance Comparison ---")
print(summary_df.to_string(index=False))

plt.figure(figsize=(12, 6))
bar_width = 0.25
index = np.arange(len(summary_df['Model']))
plt.bar(index, summary_df['Avg MAE'], bar_width, label='Avg MAE', alpha=0.5)
plt.bar(index + bar_width, summary_df['Avg RMSE'], bar_width, label='Avg RMSE',
alpha=0.3)
plt.bar(index + 2 * bar_width, summary_df['Avg MAPE'], bar_width, label='Avg MAPE',
alpha=0.2)
plt.title('Model Performance Comparison')
plt.xlabel('Model')
plt.ylabel('Error ($ or %)')
plt.xticks(index + bar_width, summary_df['Model'], rotation=45)
plt.legend()
plt.grid(True)
plt.show()

mae_dollars = product_perf_df[['MAE_RF', 'MAE_XGB']].reset_index()
mae_dollars = mae_dollars.rename(columns={'MAE_RF': 'RF_MAE_$_per_product_month',
'MAE_XGB': 'XGB_MAE_$_per_product_month'})
print("\n--- MAE ($) per Product-Month ---")
print(mae_dollars.head())
```

```
unit_price_df =
panel_df.groupby('Product_Name')['Order_Item_Product_Price'].mean().reset_index()

unit_price_df.columns = ['Product_Name', 'Avg_Unit_Sale_Price']

print("\n--- Average Unit Sale Price per Product ---")

print(unit_price_df.head())

monthly_revenue_df = panel_df.groupby('Product_Name').agg({

   'Sales': 'sum',

   'Month': 'nunique'

}).reset_index()

monthly_revenue_df['Avg_Monthly_Revenue'] = monthly_revenue_df['Sales'] /
monthly_revenue_df['Month']

monthly_revenue_df = monthly_revenue_df[['Product_Name', 'Avg_Monthly_Revenue']]

print("\n--- Average Monthly Revenue per Product ---")

print(monthly_revenue_df.head())

economic_summary_df = mae_dollars.merge(unit_price_df, on='Product_Name', how='left') \
                          .merge(monthly_revenue_df, on='Product_Name', how='left')


print("\n--- Economic Summary Table (First 10 Products) ---")

print(economic_summary_df.head(10))
```

Acknowledgements

I would like to sincerely thank **Professor Enrico Rettore** for his guidance and support throughout this thesis. His thoughtful feedback, and clear advice were essential at every stage of the work. I have learned a great deal from his perspective and approach to research, and I am especially grateful for the time and attention he gave to my questions and drafts.

It has been a privilege to work under his supervision, and I truly appreciate his patience and the confidence he placed in me during this process.