T.C.
İSTANBUL KÜLTÜR UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

DATA-DRIVEN MACHINE LEARNING APPROACHES TO DEMAND FORECASTING IN SUPPLY CHAIN MANAGEMENT

MASTER'S THESIS

Safa ALREFAAI

1700005186

Department: Industrial Engineering

Program: Engineering Management

Supervisor: Prof. Dr. Murat ERMIŞ

SEPTEMBER 20225

T.C.
İSTANBUL KÜLTÜR UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

DATA DRIVEN MACHINE LEARNING APPROACHES TO DEMAND FORECASTING IN SUPPLY CHAIN MANAGEMENT

MASTER'S THESIS

Safa ALREFAAI

1700005186

Department: Industrial Engineering

Program: Engineering Management

Supervisor:	**Prof. Dr. Murat ERMİŞ**
Members of Examining Committee:	**Prof. Dr. Ö. Koray ŞAHİNGÖZ**
	Assist. Prof. Dr. Okay IŞIK

SEPTEMBER 2025

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who has contributed to the successful completion of this study. First and foremost, I extend my heartfelt thanks to my supervisor, Prof.Dr. Murat Ermis, whose unwavering support, expert guidance, and insightful feedback have been instrumental throughout this research. His patience, encouragement, and constructive advice have significantly shaped my work, and I am incredibly fortunate to have had the opportunity to learn from such a knowledgeable and dedicated mentor.

I would also like to thank my family for their endless love and encouragement. To my parents, I am deeply grateful for your constant belief in me and for being my pillars of strength. Your sacrifices and unwavering faith in my abilities have been a continuous source of motivation throughout this journey. To my siblings, thank you for your understanding, patience, and for always being there to support and uplift me when needed.

Lastly, I would like to take a moment to acknowledge myself for the dedication, hard work, and perseverance that I have put into this academic journey. This accomplishment stands as a reflection of the countless hours of study, research, and personal growth that I have undertaken. I am proud of what I have achieved and look forward to applying the knowledge and skills I have gained to future endeavors.

Once again, thank you to everyone for your support and encouragement. This achievement would not have been possible without you.

11/09/2025 Safa ALREFAAI

TABLE OF CONTENTS

ACKNOWLEDGEMENT ..... **i**
TABLE OF CONTENTS ..... **ii**
LIST OF SYMBOLS ..... **iv**
LIST OF TABLES ..... **v**
LIST OF FIGURES ..... **vii**
ÖZET ..... **xi**
ABSTRACT ..... **xii**
1. INTRODUCTION **1**
2. LITERATURE REVIEW **3**
2.1. Demand Forecasting in Supply Chain Management **3**
2.2. Machine Learning for Demand Forecasting **5**
3. METHODOLOGY **9**
3.1. Data Visualization in Supply Chain Management **10**
3.2. Exploratory Data Analysis (EDA) **11**
3.3. Statistical Models **13**
3.3.1. Moving average (MA) 14
3.3.2. Exponential Smoothing (ES) 14
3.3.3. ARIMA (AutoRegressive Integrated Moving Average) 15
3.3.4. SARIMA (Seasonal AutoRegressive Integrated Moving Average) 16
3.4. Machine Learning Models **17**
3.4.1. Decision Tree (DT) 18
3.4.2. Random Forest (RF) 19
3.4.3. Extreme Gradient Boosting (XGBoost) 20
3.5. Model Evaluation Metrics **20**
3.5.1. The Mean Absolute Error (MAE) 21
3.5.2. RMSE (Root Mean Squared Error) 21
3.5.3. MAPE (Mean Absolute Percentage Error) 21
3.5.4. R^2 (Coefficient of Determination) 21
3.6. Comparative Analysis **22**
3.7. Repeatability and Rigor **22**
4. IMPLEMENTATION AND RESULTS **24**
4.1. Overview of the Dataset (case study) **24**

4.2. Data Visualization in the case study 25
4.3. Exploratory Data Analysis (EDA) in the Case Study 28
4.4. Statistical Models in the Case Study 39
4.4.1. Moving Average 39
4.4.2. Single Exponential Smoothing 40
4.4.3. ARIMA 41
.4.4.4 SARIMA 44
.4.4.5 Evaluation of Statistical Models 44
4.5. Machine Learning in the Case Study 46
4.5.1. Data Preparation 47
4.5.2. Training and Testing Design 48
4.5.3. Model Training 49
4.5.4. Evaluation of Machine Learning Models 50
4.6. Comparative Analysis in the Case Study 52
4.6.1. Comparative Findings 53
5. CONCLUSION 54
REFERENCES 56
APPENDIX 62

LIST OF SYMBOLS

y_t: actual observed value at time t.

$\widehat{y_t}$: predicted value at time t.

$\bar{y}$: mean of the observed values.

n : total number of observations.

LIST OF TABLES

Table 1. Features and attributes. 25

Table 2. The monthly aggregated demand for each product class. 37

Table 3. Statistical model results on the testing set. 46

Table 4. Machine learning model results on the testing set. 51

Table 5. Statistical model results on the testing set for Analgesics class. 69

Table 6. Statistical model results on the testing set for Antibiotics class. 74

Table 7. Statistical model results on the testing set for Antimalarial class. 79

Table 8. Statistical model results on the testing set for Antipiretics class. 83

Table 9. Statistical model results on the testing set for Antiseptics class. 88

Table 10. Statistical model results on the testing set for Mood Stabilizers class..... 92

LIST OF FIGURES

Figure 1. Basic structure and material flow in the Supply Chain. 1

Figure 2. Conceptual Flowchart. 10

Figure 3. Flowchart of Machine Learning. 18

Figure 4. Sales and Demand by Product Class. 26

Figure 5. Monthly Trends in Sales and Demand. 26

Figure 6. Demand by Product Class and Country. 27

Figure 7. Sales by Product and Distributor. 28

Figure 8. The Relationship Between Demand and Price. 29

Figure 9. The Relationship Between Demand and Price along with Country “Germany". 29

Figure 10. The Relationship Between Demand and Price along with Country “Poland". 30

Figure 11. The Relationship Between Demand and Price along with Channel “Hospital". 30

Figure 12. The Relationship Between Demand and Price along with Channel "Pharmacy" 31

Figure 13. The Relationship Between Demand and Price along with Sub-channel “Government". 31

Figure 14. The Relationship Between Demand and Price along with Sub-channel “Institution”. 32

Figure 15. The Relationship Between Demand and Price along with Sub-channel “Private". 32

Figure 16. The Relationship Between Demand and Price along with Sub-channel “Retail". 33

Figure 17. The Relationship Between Demand and Price along with Product Class "Analgesics". 33

Figure 18. The Relationship Between Demand and Price along with Product Class "Antibiotics". 34

Figure 19. The Relationship Between Demand and Price along with Product Class "Antimalarial". 34

Figure 20. The Relationship Between Demand and Price along with Product Class "Antipiretics". 35

Figure 21. The Relationship Between Demand and Price along with Product Class "Antiseptics". 35

Figure 22. The Relationship Between Demand and Price along with Product Class "Mood Stabilizers". 36

Figure 23. Distribution of demand before boxplot. 38

Figure 24. Distribution of demand after boxplot. 38

Figure 25. Moving Average plot for Sum of Demand. 40

Figure 26. Smoothing Plot for Sum of Demand. 41

Figure 27. Augmented Dickey–Fuller (ADF) test before differencing: the series is non-stationary. 42

Figure 28. ACF and PACF plots of the original series. 42

Figure 29. Augmented Dickey–Fuller (ADF) test after first differencing. 42

Figure 30. ACF and PACF plots of the series after first differencing 43

Figure 31. ARIMA model selection based on AICc. 43

Figure 32. Forecast results from ARIMA (1,1,0) 44

Figure 33. Correlation heatmap. 48

Figure 34. Moving Average Plot for Sum of Demand Analgesics. 65

Figure 35. Smoothing Plot for Sum of Demand Analgesics. 66

Figure 36. ADF Test of Demand Analgesics before differencing. 66

Figure 37. ACF & PACF of Demand Analgesics before differencing. 67

Figure 38. ADF Test of Demand Analgesics after differencing. 67
Figure 39. ACF & PACF of Demand Analgesics after differencing. 67
Figure 40. ARIMA model selection based on AICc for Demand Analgesics. 68
Figure 41. Forecast results from ARIMA (0,1,1) for Demand Analgesics. 68
Figure 42. Moving Average Plot for Sum of Demand Antibiotics. 70
Figure 43. Smoothing Plot for Sum of Demand Antibiotics. 71
Figure 44. ADF Test of Demand Antibiotics before differencing. 71
Figure 45. ACF & PACF of Demand Antibiotics before differencing. 72
Figure 46. ADF Test of Demand Antibiotics after differencing. 72
Figure 47. ACF & PACF of Demand Antibiotics after differencing. 72
Figure 48. ARIMA model selection based on AICc for Demand Antibiotics. 73
Figure 49. Forecast results from ARIMA (0,1,0) for Demand Antibiotics. 73
Figure 50. Moving Average Plot for Sum of Demand Antimalarial. 75
Figure 51. Smoothing Plot for Sum of Demand Antimalarial. 75
Figure 52. ADF Test of Demand Antimalarial before differencing. 76
Figure 53. ACF & PACF of Demand Antimalarial before differencing. 76
Figure 54. ADF Test of Demand Antimalarial after differencing. 76
Figure 55. ACF & PACF of Demand Antimalarial after differencing. 77
Figure 56. ARIMA model selection based on AICc for Demand Antimalarial. 77
Figure 57. Forecast results from ARIMA (0,1,1) for Demand Antimalarial. 78
Figure 58. Moving Average Plot for Sum of Demand Antipiretics. 79
Figure 59. Smoothing Plot for Sum of Demand Antipiretics. 80
Figure 60. ADF Test of Demand Antipiretics before differencing. 80
Figure 61. ACF & PACF of Demand Antipiretics before differencing. 81
Figure 62. ADF Test of Demand Antipiretics after differencing. 81
Figure 63. ACF & PACF of Demand Antipiretics after differencing. 81

Figure 64. ARIMA model selection based on AICc for Demand Antipiretics. 82
Figure 65. Forecast results from ARIMA (0,1,1) for Demand Antipiretics. 82
Figure 66. Moving Average Plot for Sum of Demand Antiseptics. 84
Figure 67. Smoothing Plot for Sum of Demand Antiseptics. 84
Figure 68. ADF Test of Demand Antiseptics before differencing. 85
Figure 69. ACF & PACF of Demand Antiseptics before differencing. 85
Figure 70. ADF Test of Demand Antiseptics after differencing. 85
Figure 71. ACF & PACF of Demand Antiseptics after differencing. 86
Figure 72. ARIMA model selection based on AICc for Demand Antiseptics. 86
Figure 73. Forecast results from ARIMA (1,1,0) for Demand Antiseptics. 87
Figure 74. Moving Average Plot for Sum of Demand Mood Stabilizers. 88
Figure 75. Smoothing Plot for Sum of Demand Mood Stabilizers. 89
Figure 76. ADF Test of Demand Mood Stabilizers before differencing. 89
Figure 77. ACF & PACF of Demand Mood Stabilizers before differencing. 89
Figure 78. ADF Test of Demand Mood Stabilizers after differencing. 90
Figure 79. ACF & PACF of Demand Mood Stabilizers after differencing 90
Figure 80. ARIMA model selection based on AICc for Demand Mood Stabilizers. 90
Figure 81. Forecast results from ARIMA (0,1,0) for Demand Mood Stabilizers. 91

Üniversite : **İstanbul Kültür Üniversitesi**

Enstitü : **Lisansüstü Eğitim Enstitüsü**

Anabilim Dalı : **Endüstri Mühendisliği**

Programı : **Mühendislik Yönetimi**

Tez Danışmanı : **Prof. Dr. Murat ERMİŞ**

Tez Türü ve Tarihi : **Yüksek Lisans – Eylül 2025**

ÖZET

TEDARIK ZINCIRI YÖNETIMINDE TALEP TAHMINI IÇIN VERI ODAKLI MAKINE ÖĞRENMESI YAKLAŞIMLARI

Safa ALREFAAI

İlaç tedarik zincirlerinde talep tahmini kritik bir işlevdir; yanlış tahminler, stok yetersizliklerine, aşırı envantere ve finansal kayıplara yol açabilir. Bu tezde, iki yıllık aylık veri kullanılarak bir ilaç vaka çalışmasına istatistiksel yöntemler (Hareketli Ortalama, Tek Üstel Düzeltme, ARIMA) ve makine öğrenmesi modelleri (Karar Ağacı, Rastgele Orman, XGBoost) uygulanmış ve tahmin performansları değerlendirilmiştir. Veri seti 21 ay eğitim ve 3 ay test olarak ikiye ayrılmış, modellerin performansı Ortalama Mutlak Hata (MAE), Kök Ortalama Kare Hatası (RMSE) ve Ortalama Mutlak Yüzde Hatası (MAPE) ile ölçülmüştür.

Sonuçlar, istatistiksel modellerin makine öğrenmesi modellerinden tutarlı bir şekilde daha iyi performans gösterdiğini ortaya koymuştur. Hareketli Ortalama, Tek Üstel Düzeltme ve ARIMA yaklaşık %18–19 MAPE değerleriyle düşük hata oranları sağlamıştır. Makine öğrenmesi modelleri gecikme değişkenleri olmadan zayıf sonuçlar üretmiş (MAPE > %320), ancak gecikmeler eklendiğinde iyileşme göstermiştir. Rastgele Orman ve XGBoost en iyi performansı göstermelerine rağmen, hata oranları (%234–237 MAPE) istatistiksel yöntemlerden belirgin şekilde daha yüksek kalmıştır. Sınırlı veri koşullarında istatistiksel yaklaşımlar daha güvenilir olsa da, kullanılan kısa tarihsel dönem nedeniyle doğruluk ekonomik açıdan güçlü bir değer sunmamaktadır. Gelecekteki araştırmaların, daha uzun veri setleri ve daha zengin açıklayıcı değişkenler kullanarak tahmin doğruluğunu geliştirmesi ve ilaç tedarik zincirlerinde karar alma süreçlerini güçlendirmesi önerilmektedir.

Anahtar Kelimeler: Makine Öğrenmesi, Tedarik Zinciri Yönetimi, Talep Tahmini, Zaman Serisi Analizi.

University	**: İstanbul Kültür University**
Institute	**: Institute of Graduate Studies**
Department	**: Industrial Engineering**
Program	**: Engineering Management**
Supervisor	**: Prof. Dr. Murat ERMIS**
Degree Awarded and Date	**: Master of Science – September 2025**

ABSTRACT

DATA-DRIVEN MACHINE LEARNING APPROACHES TO DEMAND FORECASTING IN SUPPLY CHAIN MANAGEMENT

Safa ALREFAAI

Demand forecasting is a critical function in pharmaceutical supply chains, where inaccurate estimates may result in shortages, excess inventory, and financial losses. This thesis evaluates forecasting performance using statistical methods (Moving Average, Single Exponential Smoothing, ARIMA) and machine learning models (Decision Tree, Random Forest, XGBoost) applied to a pharmaceutical case study with two years of monthly data. The dataset was divided into 21 months for training and 3 months for testing, and model performance was assessed using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

The results show that statistical models consistently outperformed machine learning models. Moving Average, SES, and ARIMA achieved testing errors of around 18–19% MAPE, supported by stable MAE and RMSE values. Machine learning models performed poorly without lag variables (MAPE > 320%), but showed improvements when lags were introduced. Among them, Random Forest and XGBoost performed best, yet their errors remained substantially higher (MAPE ≈ 234–237%) compared to statistical methods. While statistical approaches were more reliable under limited data conditions, their accuracy still lacks strong economic value due to the short historical period analyzed. Future research should therefore expand the dataset and incorporate richer explanatory variables to improve predictive performance and support decision-making in pharmaceutical supply chains.

Keywords: Machine Learning, Supply Chain Management, Demand Forecasting, Time Series Analysis.

1. INTRODUCTION

Demand forecasting is a critical function in engineering management, particularly within supply chain operations where accurate planning directly influences efficiency, cost-effectiveness, and service reliability. In the pharmaceutical sector, the stakes are even higher: inaccurate demand forecasts may cause life-threatening shortages for patients or costly excess inventories that burden healthcare systems and suppliers. As global supply chains become more complex, forecasting methods have evolved from simple statistical approaches toward advanced data-driven models. This thesis is situated within this evolving landscape, aiming to evaluate how different forecasting techniques can be applied to pharmaceutical demand in order to support managerial decision-making and strengthen operational resilience.

The importance of forecasting is evident across the entire pharmaceutical supply chain. As illustrated in Figure 1, supply chains typically begin with the procurement of raw materials, followed by inventory management, manufacturing, distribution, and delivery to end-users. At every stage, inaccurate forecasts create ripple effects: overestimation leads to excess stocks and higher holding costs, while underestimation results in production bottlenecks, delivery delays, and shortages for patients. Effective forecasting therefore provides the foundation for balancing supply and demand, ensuring resource efficiency and timely access to essential medicines.

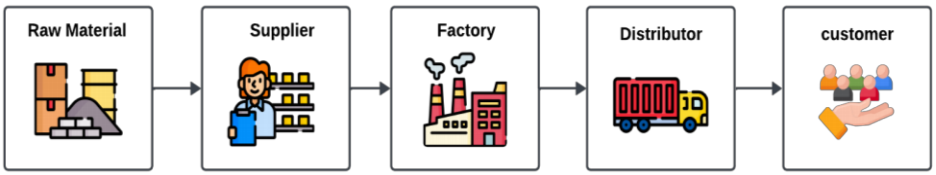


Figure 1. Basic structure and material flow in the Supply Chain.

The aim of this thesis is to investigate and compare the performance of statistical and machine learning models in forecasting pharmaceutical demand through a real case study. To accomplish this aim, four specific objectives were pursued. First, the dataset from the case study was visualized and described to provide an initial understanding of its structure. Second, exploratory data analysis (EDA) was conducted to uncover

hidden patterns, trends, and relationships within the demand data. Third, both statistical models, including Moving Average, Single Exponential Smoothing, ARIMA, and SARIMA, and machine learning models, namely Decision Tree, Random Forest, and XGBoost, were applied under identical conditions to ensure a fair comparison. Finally, the performance of these models was evaluated and compared using a unified error framework consisting of Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

The study addresses three research questions: How effectively can statistical models capture the dynamics of pharmaceutical demand? To what extent do machine learning models improve forecasting accuracy when temporal features such as lag variables are introduced? Which modeling approach provides more reliable forecasts for managerial decision-making in pharmaceutical supply chains?

By answering these questions, the research clarifies the predictive power of each approach and offers evidence-based insights for supply chain managers. The contribution of this thesis lies in providing a structured, side-by-side comparison of statistical and machine learning forecasting methods within a pharmaceutical case study. Academically, it enriches the literature by presenting a rigorous evaluation of different approaches under the same dataset and conditions. From a managerial perspective, it emphasizes the importance of choosing appropriate forecasting methods and incorporating richer datasets to reduce uncertainty, enhance efficiency, and improve resilience in pharmaceutical distribution.

This thesis report is organized as follows: Section 2 presents a literature review on pharmaceutical demand forecasting, covering both traditional statistical methods and modern machine learning techniques. Section 3 details the methodology, including data visualization and exploratory data analysis, model development, evaluation metrics, comparative analysis, and reproducibility. Section 4 presents the implementation and results of the case study, covering the dataset, data visualization and exploratory analysis, statistical models, machine learning models, and comparative evaluation. Finally, Section 5 concludes the study by summarizing the findings, highlighting managerial implications, and suggesting directions for future research.

2. LITERATURE REVIEW

Machine learning has significantly impacted supply chain management (SCM) and demand forecasting. Accurate demand forecasting is crucial for improving sales, reducing costs, and enhancing customer satisfaction. Machine learning techniques provide advantages such as identifying hidden patterns in demand and improving forecasting accuracy, particularly when using exogenous variables. For example, models like Neural Network-ARMAX have shown superior performance in managing demand, effectively reducing overstock and out-of-stock issues. However, challenges like handling data biases, managing uncertainty, and ensuring data quality persist. Machine learning methods, such as Random Forest, offer greater precision compared to traditional forecasting techniques, but selecting the right model and data preprocessing remains critical for optimal outcomes.

2.1. Demand Forecasting in Supply Chain Management

One key objective of supply chain collaboration is enhancing forecast accuracy (Raghunathan, 1999). However, since full collaboration among supply chain partners is not always feasible, it becomes essential to explore forecasting demand even with limited information from other parties. The primary cause of demand distortion in extended supply chain simulations stems from demand signal processing by supply chain members (Forrester, 1997).

Defining demand signal processing as the adjustment each supply chain participant makes to the incoming demand signal before transmitting it to the next member. As the end-customer's demand signal travels up the supply chain, it undergoes increasing distortion. This happens even when all parties use the same forecasting method. For instance, if every supply chain member applies a six-month trend analysis, demand distortion will still occur (Lee et al., 2004).

Research shows that basic forecasting methods, such as moving averages, naïve forecasting, or demand signal processing, can trigger the bullwhip effect

(Dejonckheere et al., 2001). In contrast, autoregressive linear forecasting has been shown to mitigate this effect. Additionally, simulation-based studies suggest that genetic algorithm-driven artificial agents can manage supply chains more cost-effectively than human decision-makers (Chandra & Grabis, 2005).

Traditional statistical forecasting methods have long served as foundational tools in supply chain management due to their interpretability, low computational cost, and historical success in environments with stable and structured demand. Techniques such as Moving Average (MA), Linear Regression, Exponential Smoothing, and Autoregressive Integrated Moving Average (ARIMA) have been used to produce forecasts based on historical time series patterns (Raizada & Saini, 2021). These models are especially well-suited for short- to medium-term forecasting when demand exhibits consistent seasonality or trend components.

Badr and Ahmed (2023) emphasize that classical models like Holt-Winters and ARIMA remain popular among practitioners because they require fewer parameters, are easy to interpret, and can be implemented in spreadsheet-based systems. Shibu and Agarwal (2023) applied Triple Exponential Smoothing in a retail dataset and found it effective in capturing both trend and seasonality with reasonable forecasting accuracy, particularly for fast-moving consumer goods.

Wang et al. (2019) highlighted the relevance of such models in the retail sector, where frequent and incremental data updates are common. Their findings suggest that while these techniques may lack the adaptive capabilities of machine learning models, they still offer reliable baselines for benchmarking more complex systems. In another study, Saglam et al. (2023) used ARIMA to predict electricity demand in Turkey, noting that its runtime performance and robustness to small datasets made it preferable in resource-constrained environments.

However, despite their long-standing use, traditional methods are inherently limited in handling high-dimensional, non-linear, and rapidly changing data a characteristic increasingly common in modern global supply chains. They assume linear relationships, stationarity, and often require significant manual tuning, such as identifying optimal lag values or seasonal differencing. As Younis et al. (2021) point out in their systematic review, such limitations have encouraged the integration of

traditional models with machine learning approaches to form hybrid forecasting architectures, combining the strengths of both domains.

Nevertheless, statistical models remain relevant not only as educational tools and benchmarking standards but also as components in ensemble or hybrid systems, especially when transparency, simplicity, and low latency are required. Their inclusion in demand forecasting workflows can help organizations maintain a balance between interpretability and performance in decision-making processes.

2.2. Machine Learning for Demand Forecasting

This study focuses on RUL estimation using CNN-LSTM-PSO for lithium-ion batteries. It introduces a hybrid deep neural network for improved RUL prediction and utilizes a NASA dataset to verify CNN-LSTM-PSO approach. The CNN-LSTM-PSO model outperforms other machine learning techniques and deep learning approaches, achieving better results in RUL prediction accuracy. Experimental studies confirm the effectiveness of the CNN-LSTM-PSO model (Kara, 2021).

The paper proposes online CNN selection using saliency maps for time series forecasting, employing gradient-based saliency maps to specialize CNN forecasters. It updates RoCs with concept drift detection for adaptive model selection and offers explanations for model selection using saliency maps. Empirical studies show the method's performance and scalability advantages. OS-PGSM method outperforms popular forecasting methods and recent approaches (Saadallah et al., 2021).

Time series forecasting challenges are addressed by ensemble pruning methods. Combining diverse forecasters tackles evolving time series data complexities. Gradient-based saliency maps are used for online ensemble pruning of DNNs. OEP-ROC outperforms popular forecasting methods and state-of-the-art approaches, providing reliable performance, computational efficiency, and useful explanations. Future work includes exploring parameter variations and model families (Saadallah et al., 2022).

PHM is crucial for the reliability and availability of mechanical systems. A novel MCA-BGRU framework for remaining useful life prediction is proposed, combining multi-scale CNN, BGRU, MHSA mechanism, and fully connected layers. Particle

swarm optimization is used to tune hyperparameters for RUL prediction. MCA-BGRU outperforms benchmark methods in RUL prediction accuracy and provides robust and reliable predictions compared to benchmarks (Kara, 2022).

Shibu and Agarwal (2023) discussed the importance of data analysis and visualization in supply chain management for accurate demand forecasting. In their study, they trained and analyzed four prediction models: Seasonal Naïve, Holt-Winters Triple Exponential Smoothing, Seasonal ARIMA, and Linear Regression, all of which showed an error rate not exceeding 27%. The authors concluded that choosing the appropriate forecasting model can significantly enhance business performance.

Elmir et al. (2023) proposed a smart platform-oriented approach that leverages machine learning and time series forecasting models to predict blood demand and optimize the collection and distribution process in the blood supply chain. Their study demonstrated that time series models offer accurate predictions for blood supply needs, with Artificial Neural Network, Logistic Regression, and Support Vector Machines achieving the lowest error measures.

Big data improves supply chain performance through demand forecasting. The positive significant impact of big data on supply chain performance is discussed in another article (Impact of Big Data, 2023).

Filali et al. (2022) proposed a deep learning approach based on Long Short-Term Memory (LSTM) networks for demand forecasting in supply chain management. Their study also examined alternative techniques such as Exponential Smoothing (ETS), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Networks (RNN). The results indicated that the LSTM-based deep learning model outperformed the others, making it the most effective approach for forecasting demand in supply chain operations.

Singha and Panse (2022) analyzed various machine learning models—such as Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) for supply chain demand forecasting. Their study utilized the ‘Store Item Demand Forecasting’ dataset from Kaggle to compare model effectiveness. However, they did not explicitly state which model achieved the best performance.

Deep Learning methods like ARIMA (Auto-Regressive Integrated Moving Average) and LSTM (Long Short-Term Memory) are utilized for accurate demand forecasting in Supply Chain Management 4.0, enhancing decision-making and addressing demand fluctuations effectively. LSTM is the most efficient method for demand forecasting. Demand forecasting is crucial for Supply Chain Management 4.0. Deep learning methods (ARIMA and LSTM) improve accuracy (Terrada et al., 2022).

Saglam et al. (2023) compared different methods for estimating electricity demand in Turkey's mainland, including Medium Neural Networks (MNN), Whale Optimization Algorithm (WOA), and Support Vector Machine (SVM). Among these techniques, the MNN model demonstrated the strongest correlation between forecasted and actual electricity demand. Based on their analysis, the MNN method was identified as the most effective model for this forecasting task.

Yoon et al. (2023) proposed a hybrid model that integrates K-means clustering, Least Absolute Shrinkage and Selection Operator (LASSO) regression, and Long Short-Term Memory (LSTM) deep learning to enhance demand forecasting accuracy across various industries. Their approach demonstrated improved forecasting performance and was shown to support better management and decision-making, particularly in the retail sector.

Various AI methods are employed in demand prediction techniques found in the literature. Among them, Deep Learning (DL) demonstrates superior forecasting performance compared to other methods. While forecasting based solely on historical manufacturing demand data is feasible, its accuracy improves significantly when multiple factors are considered (Dou et al., 2021). To enhance manufacturing planning and inventory management, researchers have proposed exponential forecasting frameworks for sales prediction. Therefore, our objective is to forecast future demands based on past manufacturer orders and retailers' needs, considering customer requirements, and providing predictions in advance to offset manufacturer lead times (Younis et al., 2021).

Several Machine Learning (ML) and DL methods have been applied to improve demand forecasting accuracy. compared a multi-layer Long Short-Term Memory (LSTM) model with other time-series forecasting techniques, such as K-Nearest

Neighbors (KNN), Exponential Smoothing (ETS), Auto-Regressive Integrated Moving Average (ARIMA), Support Vector Machines (SVM), Recurrent Neural Networks (RNN), and Artificial Neural Networks (ANN). The findings indicated that LSTM outperformed these methods in terms of prediction accuracy (Abbasimehr et al., 2020)

Similarly, LSTM and Grey Model (GM) used to forecast future values based on historical data and total industrial value, using data from the Statistical Yearbook of GD (2005-2020). Their experiments showed that LSTM delivered excellent results, while the GM model, despite being a classic auto-regression model, performed significantly worse when considering multiple (Dou et al., 2021).

Raizada and Saini (2021) conducted a comparative analysis of various supervised machine learning algorithms including Support Vector Machine (SVM), Random Forest Regression, K-Nearest Neighbors (KNN), and Extra Tree Regression to forecast sales across 45 Walmart retail outlets in India. Their findings indicated that Extra Tree Regression yielded the highest efficiency for the given dataset, although its prediction accuracy varied depending on the variance within the training data.

Wang et al. (2019) compared classical forecasting models with emerging techniques to predict the sales of both perishable and non-perishable products from a large grocery retailer. Their findings indicated that Support Vector Machine (SVM), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) models were particularly effective for forecasting perishable goods, while ARIMA outperformed others in terms of runtime efficiency. For non-perishable items, LSTM proved to be the most efficient model due to its advanced predictive capabilities.

Overall, demand forecasting methods vary in accuracy depending on the models used. Employing multiple forecasting models can highlight differences in predictions, indicating the need for further research or improved data inputs.

3. METHODOLOGY

The primary objective of this research is to design and evaluate a range of forecasting models for predicting demand in supply chain management. In contrast to hybrid frameworks that integrate multiple techniques into a single system, this study will apply each model independently. This approach will ensure that the comparative analysis remains transparent, unbiased, and focused on the distinct predictive capabilities of each method.

Both statistical forecasting models and machine learning models will be implemented to capture different dimensions of demand behavior. Statistical models will be employed to capture the temporal structure inherent in the demand series, while machine learning methods will be applied to exploit the relationships between demand and explanatory features. The goal is to assess the relative strengths of these two paradigms and determine which is more effective in forecasting demand within the given context.

The methodological design will follow a structured sequence of steps as illustrated in Figure 2 The process will begin with data visualization, where Power BI dashboards will be used to provide an interactive overview of the dataset. Following visualization, an exploration data analysis (EDA) will be performed to examine the dataset and identify potential relationships. As part of this step, scatter plots and correlation measures will be used to assess associations between the target and explanatory variables, and the data will be aggregated across temporal and categorical groupings to highlight variability patterns. Additionally, box plots will be employed to detect and remove outliers to ensure that both statistical and machine learning models are trained in clean and representative data. This will be followed by the implementation of statistical models (Moving Average, Exponential Smoothing, ARIMA, SARIMA), and the development of machine learning models (Decision Tree, Random Forest, and XGBoost). The next stage will involve the evaluation of model performance using

standardized error metrics (MAE, RMSE, MAPE, and R^2). Finally, a comparative analysis will be conducted to benchmark the models against each other.

This sequential structure provides a comprehensive investigation framework. It will begin with visualization and exploration techniques to build an understanding of the data, proceed with predictive models that utilize either time-dependent components or feature-driven structures, and conclude with a consistent evaluation that ensures fairness and clarity in the comparison of results.

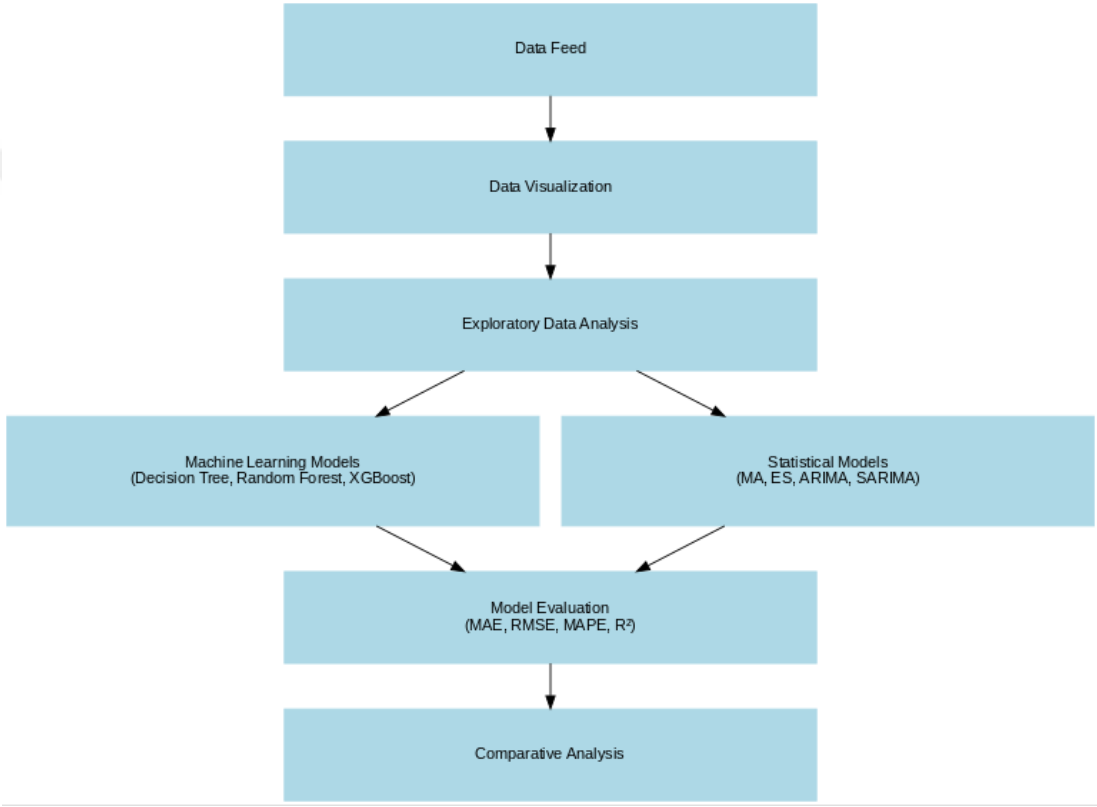


Figure 2. Conceptual Flowchart.

3.1. Data Visualization in Supply Chain Management

Data visualization is a critical component of supply chain management, as it translates raw and complex datasets into actionable insights. By converting numerical and categorical information into charts, dashboards, and maps, organizations can identify demand patterns, inefficiencies, and risks more effectively. This enhances decision-making and strategic planning by providing managers with a comprehensive overview of supply chain activities.

The visualization process begins with data collection before moving into the stages of visualization, reporting, and sharing. This flow highlights the role of visual tools as a bridge between raw data and practical decisions.

Visualization applications in supply chains are diverse. In inventory management, dashboards track stock levels across locations and highlight shortages or overstock situations. In logistics, route maps and shipment tracking diagrams reveal delivery inefficiencies. Visualization also supports supplier evaluation through scorecards and comparison graphs, while demand forecasting benefits from time-series and scatter plots that highlight trends and seasonal changes. In risk management, scenario dashboards help organizations anticipate disruptions and prepare contingency plans.

A wide range of tools support these applications. Spreadsheet software remains useful for basic charts, while advanced platforms such as Power BI and Tableau allow real-time, interactive monitoring of key performance indicators. These tools integrate with enterprise systems to provide up-to-date insights into procurement, production, and distribution. Programming environments such as Python and R also offer libraries like Matplotlib and Seaborn for customized visuals and predictive modeling.

Despite these benefits, visualization has its challenges. Its effectiveness depends on the quality of the underlying data, seamless integration with existing systems, and the data literacy of end-users. Poor-quality or inconsistent data can mislead managers, while limited integration with legacy systems may hinder real-time insights. Moreover, even advanced dashboards are of little use without proper interpretation.

In summary, data visualization enhances agility, collaboration, and responsiveness in supply chain management. By transforming raw datasets into clear, meaningful insights, organizations can make better decisions, strengthen internal communication, and improve resilience against uncertainty.

3.2. Exploratory Data Analysis (EDA)

Prior to the implementation of forecasting models, an exploration data analysis (EDA) will be conducted to investigate the statistical and visual characteristics of the dataset. This step will be essential for gaining an initial understanding of the data and for providing justification for the subsequent methodological choices. Scatter plots will be generated to visually assess the relationship between the target variable, demand

(y), and the explanatory features (X_i) By examining these visual patterns, it will be possible to identify whether meaningful associations exist between demand and the independent variables.

To complement the visual inspection, the correlation coefficient will be employed to quantify the degree of linear association between demand and the explanatory features.

$$Correl(X,Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \ \sum(y-\bar{y})^2}} \quad (1)$$

Where:

x: An individual value of variable X.

y: An individual value of variable Y.

$\bar{x}$: The Mean (average) of all values of X

$\bar{y}$: The Mean (average) of all values of Y

Σ: Summation over all data points.

Numerator $\sum(x-\bar{x})(y-\bar{y})$:Measures the covariance between X, Y.

Denominator $\sqrt{\sum(x-\bar{x})^2 \ \sum(y-\bar{y})^2}$: Product of the standard deviations of X and Y.

This statistical measure will provide a numerical assessment of the strength and direction of associations, supporting the interpretation of the visual analysis. The primary motivation for applying this measure will be to determine the extent to which the predictor variables influence demand, thereby offering evidence to support the application of supervised learning models. At the same time, this step will ensure that the demand series exhibits sufficient temporal structure, which is a prerequisite for applying time series–based statistical forecasting methods.

the analysis will include the aggregation of observations across temporal and categorical groupings. This procedure will allow for the examination of monthly demand variability across categories, thereby providing a clearer view of structural differences and trends before model development.

To further refine the dataset, box plots will be employed to identify and handle outliers. The box plot is a graphical representation of the distribution of a dataset that highlights

the median, quartiles, and extreme values. Outliers will be defined using the interquartile range (IQR) method:

$$\begin{aligned} \text{IQR} &= Q_3 - Q_1 \\ \text{Lower Bound} &= Q_1 - 1.5 \times \text{IQR} \\ \text{Upper Bound} &= Q_3 + 1.5 \times \text{IQR} \end{aligned} \tag{2}$$

Where:

Q_1: The first quartile (25th percentile).

Q_3: The third quartile (75th percentile)

IQR :The spread between Q_3 and Q_1

Any data point below the lower bound or above the upper bound will be considered an outlier. These observations will be removed to ensure that both the statistical forecasting models and the machine learning algorithms are trained on representative data without undue influence from extreme values.

In this way, the exploratory data analysis provided both the conceptual foundation and the methodological direction for the forecasting framework developed in this study.

3.3. Statistical Models

A set of statistical forecasting models will be employed to capture the temporal dynamics and trend structures inherent in demand series. Statistical forecasting approaches rely primarily on historical demand values, making them particularly suitable for identifying seasonality, trends, and autocorrelation patterns.

The models implemented in this study will include Moving Average (MA), which smooths short-term fluctuations to reveal underlying trends; Exponential Smoothing (ES), which assigns exponentially decreasing weights to past observations; and the ARIMA family of models, which incorporates autoregressive and moving average components along with differencing to achieve stationarity. To further account for seasonal variations, the SARIMA model will also be applied, extending ARIMA by introducing seasonal terms.

By applying these models sequentially, the study will ensure comprehensive coverage of both simple smoothing techniques and more advanced autoregressive structures. This approach will provide a balanced foundation for evaluating the strengths and limitations of statistical forecasting in comparison to machine learning models.

3.3.1. Moving average (MA)

Moving average is a basic forecasting technique that calculates the forecast as the average of the most recent k observations. It smooths short-term fluctuations by giving equal weight to each of the latest data points. In practice, the value of k is typically chosen empirically by testing different window sizes and selecting the one that minimizes forecasting errors.

$$\hat{y}_{t+1} = \frac{1}{k}\sum_{i=0}^{k-1} y_{t-i} \tag{3}$$

Where:

$\hat{y}_{t+1}$ = forecast for the next time period.

y_{t-i} = the i-th most recent actual observation.

k = the number of periods included in the moving average (window size).

3.3.2. Exponential Smoothing (ES)

Exponential Smoothing is a time series forecasting technique that assigns exponentially decreasing weights to past observations. This means recent data points have a stronger influence on the forecast compared to older observations. It is particularly effective for short-term forecasting when the data does not exhibit strong seasonality or trend, The optimal value of α is usually chosen by minimizing error measures such as MAE, MSE, RMSE, or MAPE.

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha)\hat{y_t} \tag{4}$$

where:

$\hat{y}_{t+1}$= forecast for the next time period.

$\widehat{y_t}$ = forecats observed value at time t.

y_t = actual observed value at time t.

α (0< α <1) = smoothing parameter that controls the rate at which the influence of past observations decays.

3.3.3. ARIMA (AutoRegressive Integrated Moving Average)

ARIMA (AutoRegressive Integrated Moving Average) model is a classical statistical approach for time series forecasting. It combines three components: AR (p): the autoregressive part, which uses past observations to predict the current value. I (d): the integrated part, which applies differencing to remove trends and achieve stationarity. MA (q): the moving average part, which incorporates past forecast errors to improve predictions. The general form of the ARIMA (p, d, q) model is given as:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \quad (5)$$

Where:

y_t = value of the time series at time t.

c = constant term.

ϕ_i = autoregressive coefficients.

θ_j = moving average coefficients.

ϵ_t = white noise error term.

before applying ARIMA, it is necessary to verify whether the time series is stationary, since stationarity is a fundamental requirement for the model. This is assess using the Augmented Dickey–Fuller (ADF) test, where a statistically significant result (p-value < 0.05) indicates that the data can be considered stationary. represented as:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p} \delta_i \, \Delta y_{t-i} + \varepsilon_t \quad (6)$$

Where:

Δy_t = first difference of the series at time t.

α = intercept.

βt = deterministic time trend.

γ = coefficient testing stationarity (γ <0 suggests stationarity)

y_{t-1} = lagged level of the series.

δ_i = coefficients for lagged differences.

ε_t = error term.

Once stationarity is confirmed, different combinations of parameters (p, d, q) are compared. The optimal model is then selected based on information criteria, particularly the Akaike Information Criterion corrected (AICc). The model with the lowest AICc value is chosen as it achieves the best trade-off between goodness of fit and model complexity.

$$AIC = 2k - 2\ln(L) \tag{7}$$

Where:

k = number of estimated parameters.

L = maximum value of the likelihood function.

$$AIC = AIC + \frac{2k(k+1)}{n-k-1} \tag{8}$$

Where:

AIC = Akaike Information Criterion

n = sample size.

k = number of estimated parameters.

3.3.4. SARIMA (Seasonal AutoRegressive Integrated Moving Average)

the SARIMA (Seasonal AutoRegressive Integrated Moving Average) model extends ARIMA by incorporating seasonal components. It is denoted as SARIMA (p, d, q) (P, D, Q) s, where the seasonal terms (P, D, Q)capture autoregressive, differencing, and moving average effects at a seasonal period s. While the general procedure for testing stationarity and selecting the optimal model using the Augmented Dickey–Fuller test and AICc has already been detailed in the ARIMA subsection, the SARIMA framework applies the same principles with the added capacity to

model periodic seasonal fluctuations. In practice, the seasonal differencing order the seasonal differencing order D is determined by applying the Augmented Dickey–Fuller (ADF) test on the original and seasonally differenced series to assess stationarity. This makes SARIMA particularly suitable for time series that exhibit recurring seasonal patterns, such as monthly or quarterly demand data.

$$\Phi_P(B^s)\,\phi_p(B)\,(1-B)^d(1-B^s)^D y_t = c + \Theta_Q(B^s)\,\theta_q(B)\,\epsilon_t \quad (9)$$

where:

y_t= value of the time series at time t.

c= constant term.

ϵ_t = white noise error term.

$\phi_p(B)$= non-seasonal autoregressive (AR) polynomial of order p.

$\theta_q(B)$ = non-seasonal moving average (MA) polynomial of order q.

$(1-B)^d$ = non-seasonal differencing of order d.

$\Phi_P(B^s)$ = seasonal autoregressive (SAR) polynomial of order P.

$\Theta_Q(B^s)$ = seasonal moving average (SMA) polynomial of order Q.

$(1-B^s)^D$ = seasonal differencing of order Q.

s = length of the seasonal cycle (e.g., 12 for monthly data).

3.4. Machine Learning Models

Machine learning methods will be adopted to capture complex, non-linear, and high-dimensional relationships between explanatory variables and demand. These models are chosen because of their flexibility in modeling interactions and their ability to uncover predictive patterns that traditional linear models may fail to capture.

Each model will be applied independently to maintain transparency and avoid bias in the comparative analysis. Three supervised algorithms will be implemented: Decision Tree, which provides a simple yet interpretable baseline; Random Forest, which reduces variance by aggregating multiple trees through bagging; and XGBoost, which employs gradient boosting to achieve enhanced accuracy and robustness. Figure 3 illustrates the flowchart of the machine learning modeling process.

This design will ensure that the predictive potential of each machine learning technique is assessed individually, establishing a clear and fair comparison with statistical forecasting models.

Figure 3. Flowchart of Machine Learning.

3.4.1. Decision Tree (DT)

A Decision Tree is a supervised learning model that predicts the target by splitting the dataset into regions based on the values of input features. At each step, the algorithm chooses the feature and threshold that best separate the data in terms of prediction accuracy. The underlying idea is to divide the feature space into smaller regions, where each region corresponds to a prediction. When a sample falls into a region, the tree predicts the average demand of the training data within that region.

An important aspect of Decision Trees is the use of hyperparameters, which control the structure and complexity of the model. The parameter max_depth specifies the maximum number of levels the tree can grow, thereby limiting its complexity and reducing the risk of overfitting. The parameter min_samples_split defines the minimum number of observations required to further split a node, ensuring that divisions occur only when there is sufficient evidence.

The values for these hyperparameters are determined through a trial-and-error approach, where different combinations are tested and their outcomes compared. This iterative process allows for the identification of parameter settings that achieve a balance between prediction accuracy and model generalization.

$$\hat{y}(X) = \sum_{m=1}^{M} \hat{c}_m \, I(X \in R_m) \tag{10}$$

where:

R_m = region m created by the splits.

$\hat{c}_m$ = mean demand value in that region.

$I(X \in R_m) = 1$ if X belongs to region R_m, otherwise 0.

3.4.2. Random Forest (RF)

Random Forest is an ensemble learning method that combines the predictions of many Decision Trees. Instead of relying on a single tree, which may easily overfit, the algorithm builds multiple trees using random subsets of both data and features. The individual predictions are then averaged to produce the final forecast. The core idea is to combine many weak learners (trees) to create a stronger and more stable model. This ensemble approach reduces variance, limits overfitting, and generally improves predictive performance compared to a single tree.

The behavior of the Random Forest is largely influenced by its hyperparameters. The parameter n_estimators defines the number of trees included in the forest, with a larger number typically leading to more stable predictions at the expense of higher computational cost. The parameter max_depth specifies the maximum depth of each tree, controlling the complexity of the model and preventing overfitting.

The values for these hyperparameters are determined through a trial-and-error approach, where different combinations are tested and their outcomes compared. This iterative process allows for the identification of parameter settings that achieve a balance between prediction accuracy and model generalization.

$$\hat{y}(X) = \frac{1}{B} \sum_{b=1}^{B} f_b(X) \tag{11}$$

where:

B = number of trees.

$f_b(X)$= prediction of the b_{th} tree.

3.4.3. Extreme Gradient Boosting (XGBoost)

XGBoost is a tree-based ensemble method, but unlike Random Forest, it builds trees sequentially rather than independently. Each new tree is trained to correct the errors (residuals) made by the previous ones. This iterative process allows the model to gradually improve its predictions. In addition, XGBoost includes a regularization term in its objective function, which helps reduce overfitting and enhances generalization. The key idea is to learn from errors step by step, making each new tree improve upon the weaknesses of the prior ones. As a result, XGBoost is widely recognized for its accuracy, efficiency, and strong performance in forecasting tasks.

The main hyperparameters guiding the model's behavior are: n_estimators, which specifies the number of boosting rounds (i.e., how many trees are added sequentially); max_depth, which controls the depth of each tree and therefore the complexity of the model;learning_rate, which scales the contribution of each tree, ensuring that improvements are made gradually rather than all at once.

The values for these hyperparameters are determined through a trial-and-error approach, where different combinations are tested and their outcomes compared. This iterative process allows for the identification of parameter settings that achieve a balance between prediction accuracy and model generalization.

$$\mathcal{L} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_l^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{12}$$

where:

$l\,(\cdot)$ = loss function.

$f_t(x_i)$ = new tree at iteration t.

$\Omega(f_t)$ = penalty terms that control tree complexity.

3.5. Model Evaluation Metrics

To ensure a fair and consistent comparison between all forecasting models, a set of standardized error metrics will be employed. These metrics will evaluate the accuracy of predictions by quantifying the deviation between the forecasted and actual values. Each measure will capture a different aspect of model performance, thereby providing a comprehensive basis for evaluation.

3.5.1. The Mean Absolute Error (MAE)

represents the average of the absolute differences between predicted and actual values. It is a straightforward measure of overall forecasting accuracy, with lower values indicating better model performance. MAE is particularly useful because it treats all errors equally without giving additional weight to larger deviations.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{13}$$

3.5.2. RMSE (Root Mean Squared Error)

The Root Mean Squared Error (RMSE) is the square root of the average of squared differences between predicted and actual values. Unlike MAE, RMSE penalizes larger errors more heavily, making it sensitive to outliers. It is widely used as it provides a clear interpretation of error magnitude in the same unit as the target variable.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{14}$$

3.5.3. MAPE (Mean Absolute Percentage Error)

The Mean Absolute Percentage Error (MAPE) expresses accuracy as a percentage by calculating the average of absolute percentage errors between predicted and actual values. This measure allows for relative comparisons across models and datasets. However, it can be sensitive when actual values are close to zero.

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{15}$$

3.5.4. R^2 (Coefficient of Determination)

The coefficient of determination, denoted as R^2, measures the proportion of variance in the dependent variable that can be explained by the independent variables in the model. It provides an indication of the model's goodness of fit, with values closer to 1 indicating stronger explanatory power. A value of 0 suggests that the model does not

explain any of the variance, while negative values indicate that the model performs worse than a simple mean-based prediction.

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \widehat{y_t})^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \quad (16)$$

3.6. Comparative Analysis

The comparative analysis will be conducted to evaluate the relative performance of the two families of forecasting models. Statistical models are expected to exploit temporal structures such as trend components and autocorrelations within the demand series. In contrast, machine learning models, which rely on feature-based learning, are designed to capture non-linear relationships and complex interactions among predictors.

To ensure fairness and consistency, the comparison will be based on the same testing sets and evaluated using three standardized error metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). These measures directly quantify the deviation between actual and forecasted demand, providing a robust and interpretable basis for assessing accuracy.

By applying both approaches independently and comparing their performance through MAE, RMSE, and MAPE, the study will highlight the distinct advantages and limitations of each paradigm. This design ensures an unbiased evaluation framework and makes it possible to identify the conditions under which statistical methods or machine learning models may serve as the more effective forecasting strategy.

3.7. Repeatability and Rigor

The methodology will be designed with a strong emphasis on reproducibility and methodological rigor. Each stage of the process will be explicitly defined through standard formulations and consistent notation, allowing the entire workflow to be replicated on different datasets. This will ensure that the approach is not limited to a single case study but can be adapted and generalized to a wide range of demand forecasting problems in supply chain management. By maintaining clarity,

consistency, and transparency, the framework will provide both scientific credibility and practical applicability.

4. IMPLEMENTATION AND RESULTS

This section presents the implementation and results of demand forecasting in supply chain management using statistical approaches and machine learning. The workflow was carried out in several steps. First, the dataset was introduced and explained in detail, with its structure summarized in Table 1 Second, data visualization, where Power BI dashboards were used to provide an interactive overview of case study, third an exploration data analysis (EDA) was conducted. Scatter plots were generated in Excel to examine the relationship between demand and each independent feature, and box plots were applied to detect and remove outliers from the dataset to ensure clean data for subsequent model training fourth. Applied statistical models were developed using Minitab, including Moving Average, Exponential Smoothing, ARIMA, and SARIMA. Fifth, the machine learning models were implemented using Python, where three supervised algorithms Decision Tree, Random Forest, and XGBoost were. Sixth, the performance of all models was calculated using standard evaluation metrics: MAE, RMSE, and MAPE, R^2 based on testing sets. Finally, a comparative analysis is conducted to benchmark the models against each other.

4.1. Overview of the Dataset (case study)

The dataset used in this study was sourced from Kaggle. It consists of 150,920 rows and 18 columns, capturing detailed operational data from a pharmaceutical factory involved in the manufacturing and distribution of medicines to retail customers. Each row in the dataset corresponds to a unique transaction or record within the distribution process, while each column represents a distinct attribute related to the factory's supply chain operations.

Table 1 provides an overview of the dataset's features and attributes, outlining the variables used in the forecasting models and their relevance to the supply chain context.

Table 1. Features and attributes.

Features	Data Type	Explanation
Distributor	object	The name of Distributer.
Customer Name	object	The customer name.
City	object	The city name.
Country	object	The country name.
Latitude	float	The Latitude of the city.
Longitude	float	The Longitude of the city.
Channel	object	The channel of the custmer, it can be Hospital or Pharmacy.
Sub-channel	object	The sub-channel of the customer can be Government or Private or Institution or Retail.
Product Name	object	medicine name.
Product Class	object	medicine class.
Quantity	float	The demand of the customer.
Price	integer	The price of the medicine.
Sales	float	Price x quantity.
Month	object	The month of the bill.
Year	integer	The year of the bill.
Name of Sales Rep	object	The name of the sales representative.
Manager	object	The name of the manager.
Sales Team	object	The name of the sales team.

4.2. Data Visualization in the case study

Data visualization was employed in this study to transform complex pharmaceutical sales and demand data into intuitive and interpretable insights. The dashboards were designed to highlight temporal patterns, product class distributions, and geographical differences, thereby supporting both strategic and operational decision-making. Figures 4–7 summarize the visualizations developed in this study.

Figure 4 illustrates the distribution of sales and demand across six pharmaceutical product classes using pie charts and bar charts. The analysis reveals that Analgesics, Antiseptics, and Mood Stabilizers dominate overall demand and sales, while Antipiretics and antibiotics and antimalarial contribute relatively less. The bar chart provides a detailed comparison at the product level, showing which medicines generate

the highest demand. This figure highlights the heterogeneity of market performance within and across classes.

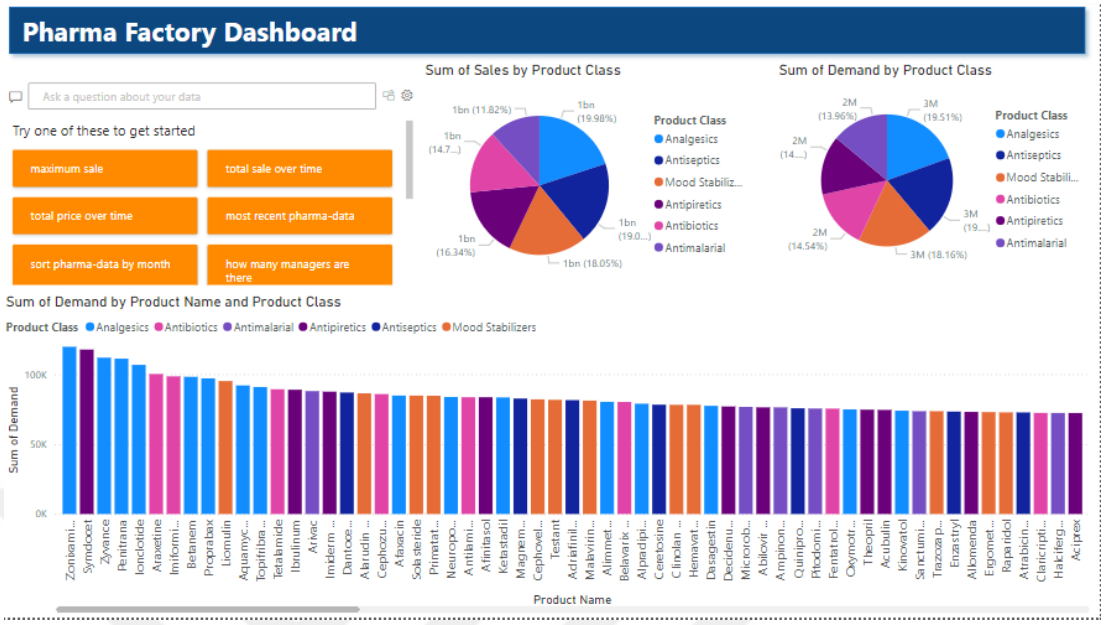


Figure 4. Sales and Demand by Product Class.

Figure 5 presents time-series line charts that compare monthly demand and sales volumes across the years 2017 and 2018. The results indicate seasonal fluctuations, with peaks occurring mid-year and toward the final quarter. Sales trends generally follow demand but with a narrower amplitude, suggesting possible constraints such as pricing strategies or supply limitations. This visualization demonstrates how temporal analysis can reveal demand supply dynamics.

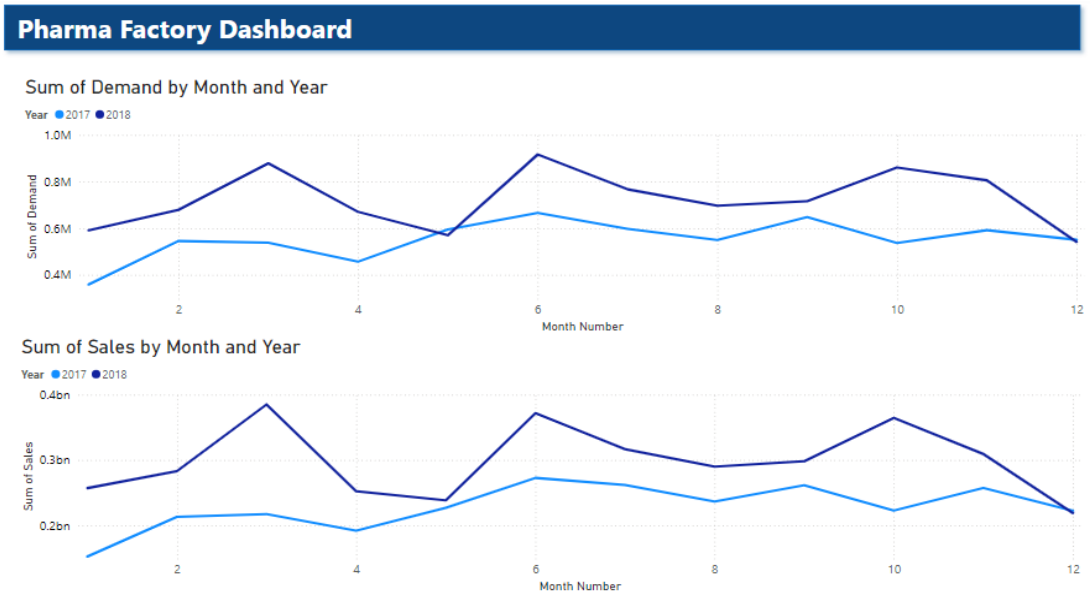


Figure 5. Monthly Trends in Sales and Demand.

Figure 6 provides a disaggregated view of demand trends across six pharmaceutical product classes for the years 2017 and 2018. The line charts show that Analgesics and Antibiotics experience more pronounced fluctuations over time, while Antipiretics and Mood Stabilizers remain comparatively stable. Antimalarial and Antiseptics products exhibit moderate variability, with certain seasonal peaks around mid-year. The geographical map highlights the concentration of demand in Central and Eastern Europe, particularly in Germany and Poland, reflecting regional market significance. Complementary gauge indicators summarize the overall results, with the total demand reaching approximately 15 million units, and the total sales amounting to nearly 6 billion. Together, these insights reveal both the temporal dynamics and spatial concentration of pharmaceutical demand.

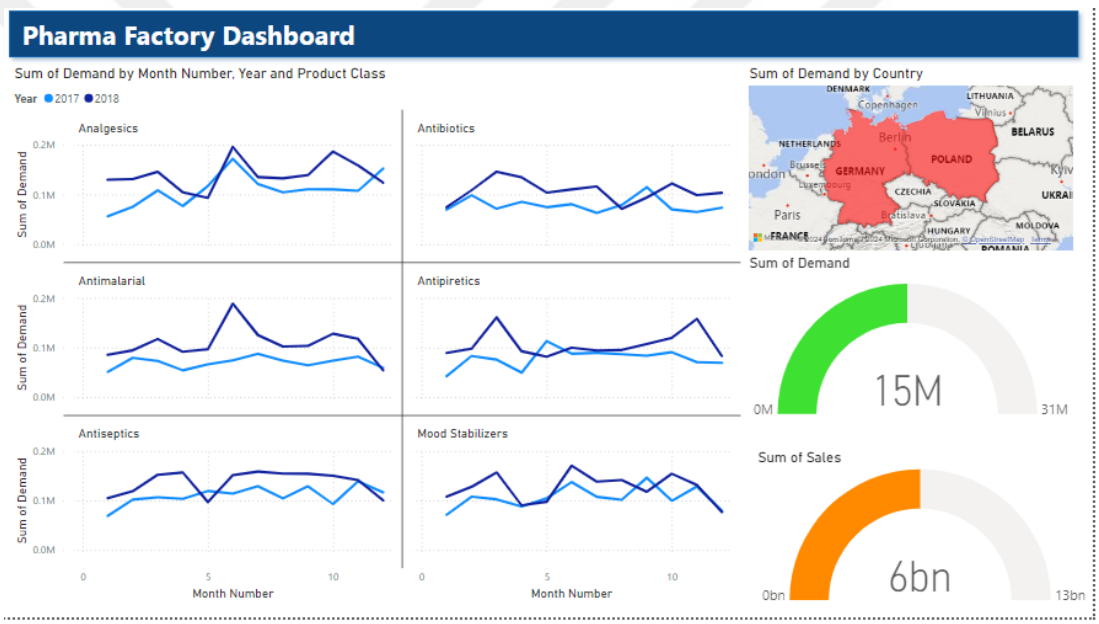


Figure 6. Demand by Product Class and Country.

Figure 7 highlights the top-performing products and distributors by sales, stratified by product class. The ranking indicates that Ionclotide, Cephozumab, and Ketastadil are leading products in terms of revenue generation. This visualization allows stakeholders to identify high-value products and key distributors, enabling better alignment of marketing and distribution strategies.

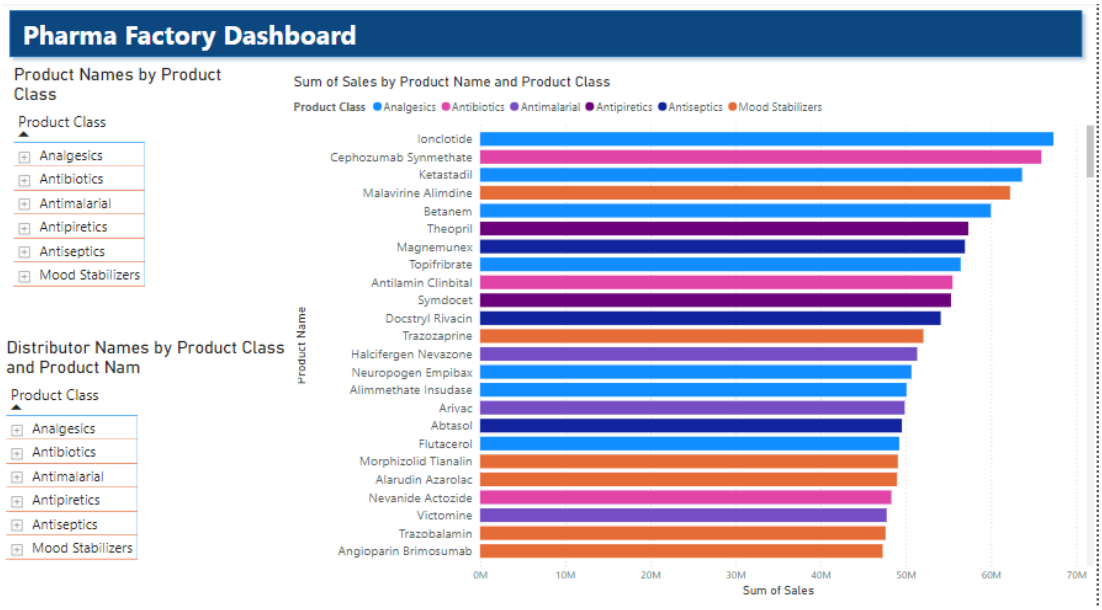


Figure 7. Sales by Product and Distributor.

4.3. Exploratory Data Analysis (EDA) in the Case Study

To explore potential relationships between demand and the explanatory variables, a series of scatter plots were generated. Since price was the only numerical predictor in the dataset, it was placed on the x-axis while demand was plotted on the y-axis. The initial plot examined the overall relationship between price and demand for the entire dataset.

To assess whether categorical features influence this relationship, the same scatter plots were segmented by different variables. These include country (e.g., Germany and Poland), distribution channel (Hospital and Pharmacy), sub-channel (Government, Institution, Private, and Retail), and product class (Analgesics, Antibiotics, Antimalarial, Antipyretics, Antiseptics, and Mood Stabilizers). By examining these plots, it becomes possible to identify whether demand patterns vary across categories and to determine whether price exerts a consistent influence under different conditions. Demand.

Figure 8 presents the overall relationship between price (x-axis) and demand (y-axis). The points are scattered randomly with no visible upward or downward trend, while the fitted regression line remains almost flat. The calculated correlation is zero, confirming that there is no significant linear relationship between price and demand in the dataset.

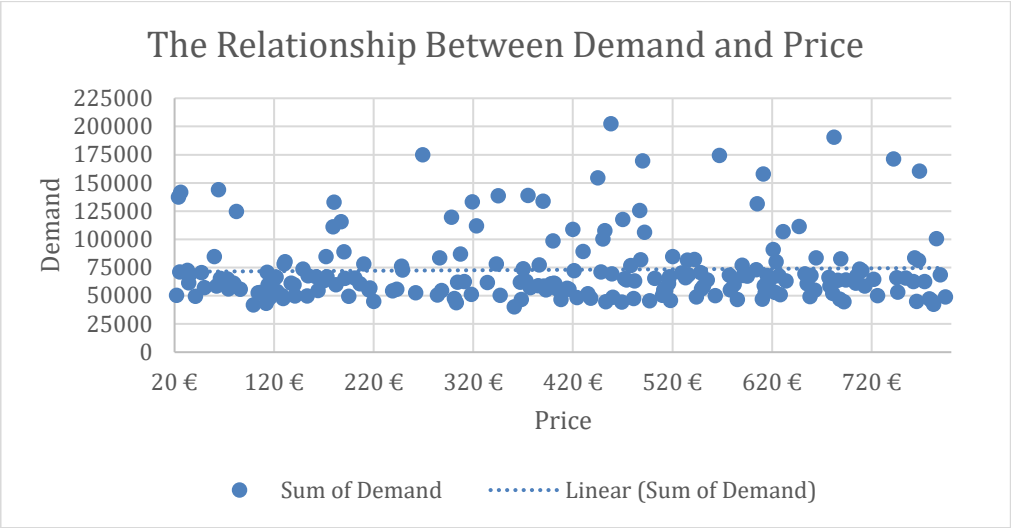


Figure 8. The Relationship Between Demand and Price.

Figure 9 illustrates the relationship between price and demand for Germany. The points appear widely scattered without a clear upward or downward pattern, and the regression line is nearly flat. The calculated correlation is 0.037, indicating that there is no meaningful linear relationship between price and demand in the German market.

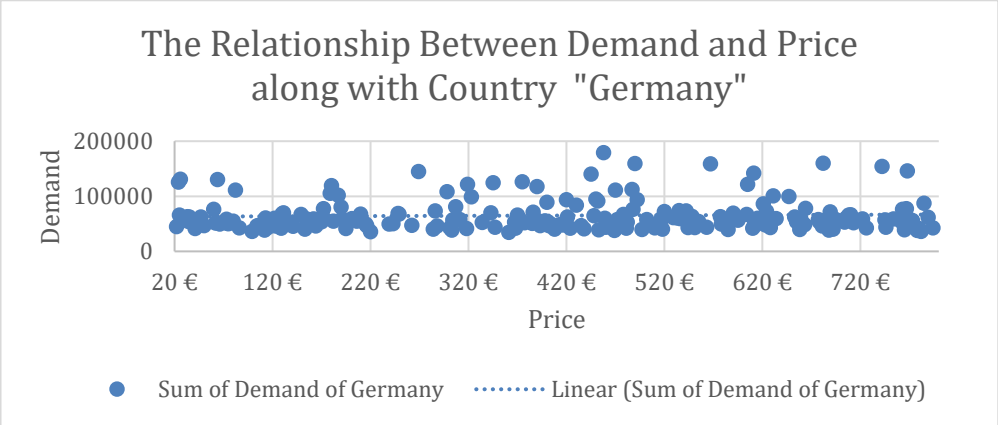


Figure 9. The Relationship Between Demand and Price along with Country “Germany".

Figure 10 presents the same analysis for Poland. The scatter points are again randomly distributed, and the fitted regression line remains almost flat. The correlation is −0.025, suggesting a very weak negative association between price and demand. While higher prices appear slightly related to lower demand, the effect is negligible and does not indicate a substantial dependency.

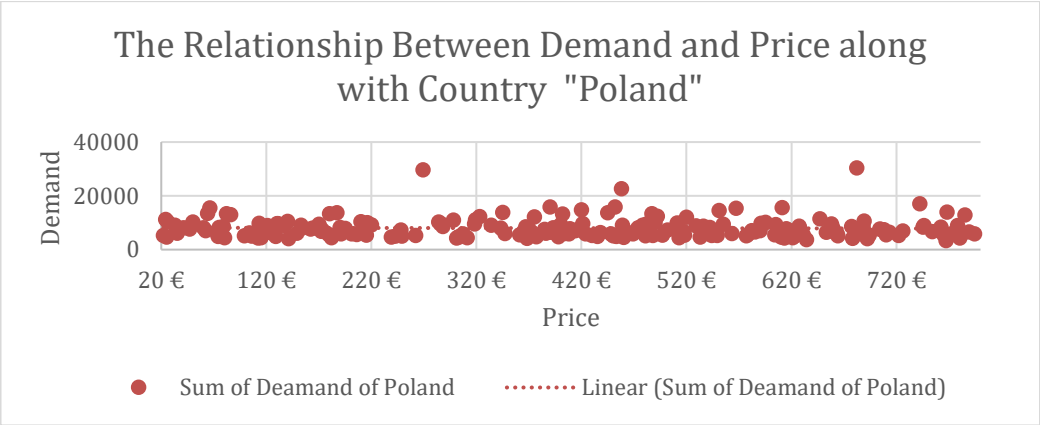


Figure 10. The Relationship Between Demand and Price along with Country “Poland".

Figure 11 illustrates the relationship between price (x-axis) and demand (y-axis) for the Hospital channel. The points are widely scattered without showing a clear upward or downward pattern. The correlation is 0.011, which is very close to zero, indicating almost no linear relationship between price and demand in this channel.

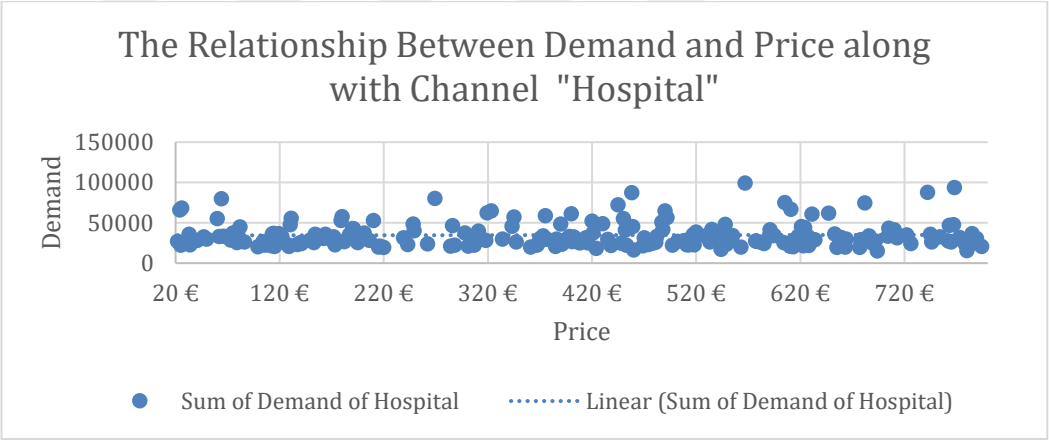


Figure 11. The Relationship Between Demand and Price along with Channel “Hospital".

Figure 12 illustrates the relationship between price (x-axis) and demand (y-axis) for the Pharmacy channel. The points are spread without a clear upward or downward pattern. The correlation is 0.043, which is very close to zero, indicating no linear relationship between price and demand in this channel.

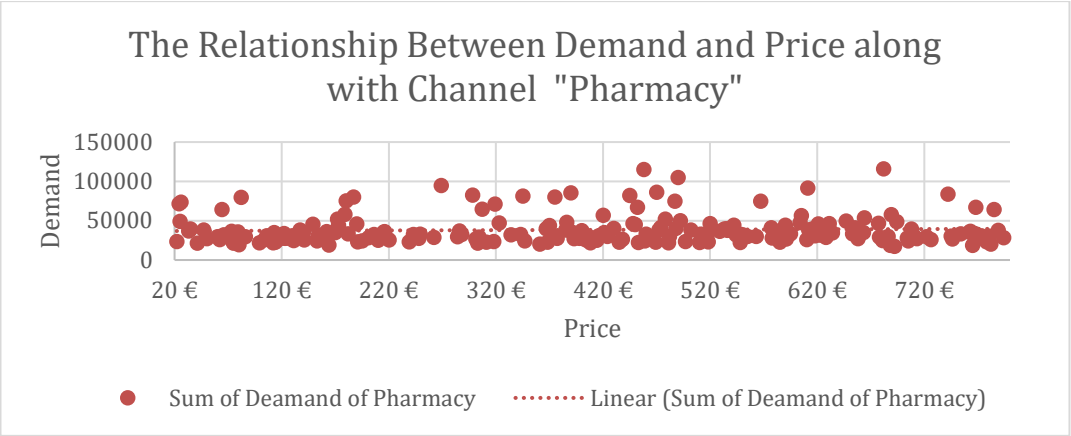


Figure 12. The Relationship Between Demand and Price along with Channel "Pharmacy"

Figure 13 shows the relationship between price and demand for the Government sub-channel. The points are scattered without a clear upward or downward pattern, and the regression line is nearly flat. The correlation is −0.061, indicating a very weak negative relationship. Although higher prices are slightly associated with lower demand, the effect is negligible.

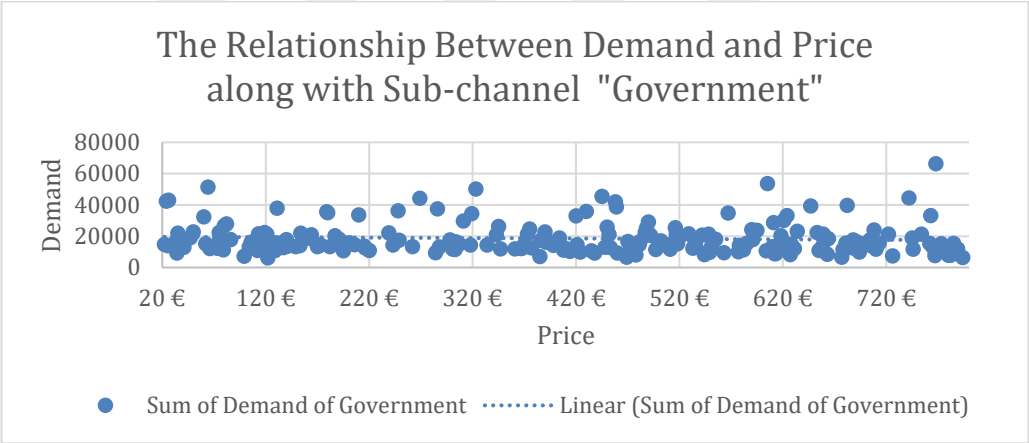


Figure 13. The Relationship Between Demand and Price along with Sub-channel "Government".

Figure 14 illustrates the results for the Institution sub-channel. The points are dispersed without a strong pattern, though the regression line shows a slight upward slope. The correlation is 0.082, suggesting a very weak positive association where higher prices are marginally linked with higher demand. However, the effect is minimal.

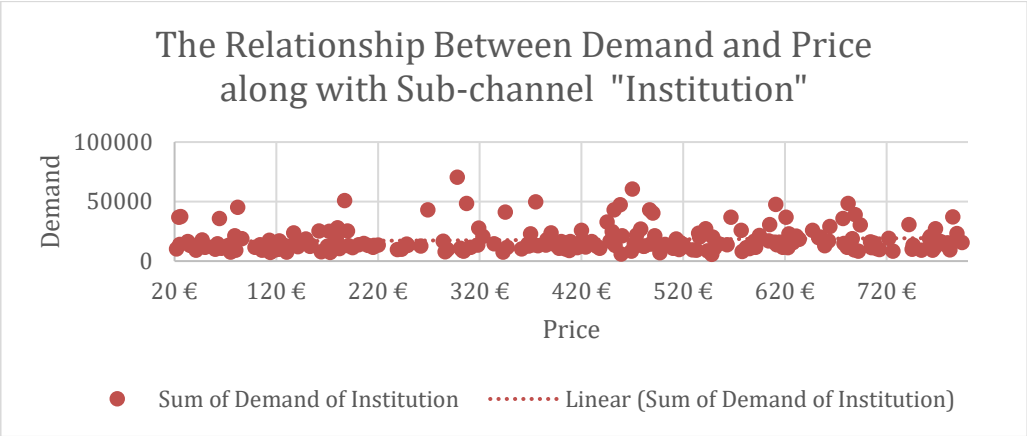


Figure 14. The Relationship Between Demand and Price along with Sub-channel "Institution".

Figure 15 presents the case of the Private sub-channel. The scatter points are spread without a pronounced trend, but the regression line indicates a small upward slope. The correlation is 0.091, again reflecting a very weak positive relationship between price and demand. This suggests a marginal link between higher prices and higher demand, though the effect remains limited.

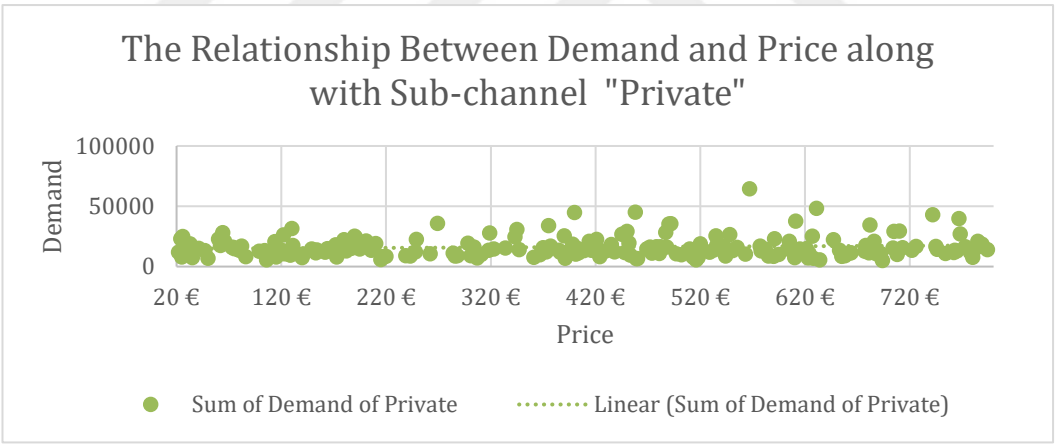


Figure 15. The Relationship Between Demand and Price along with Sub-channel "Private".

Figure 16 displays the scatter plot for the Retail sub-channel. Here, the points appear randomly distributed, and the regression line is essentially flat. The correlation is −0.010, showing a negligible negative relationship, with price changes having virtually no impact on demand in this sub-channel.

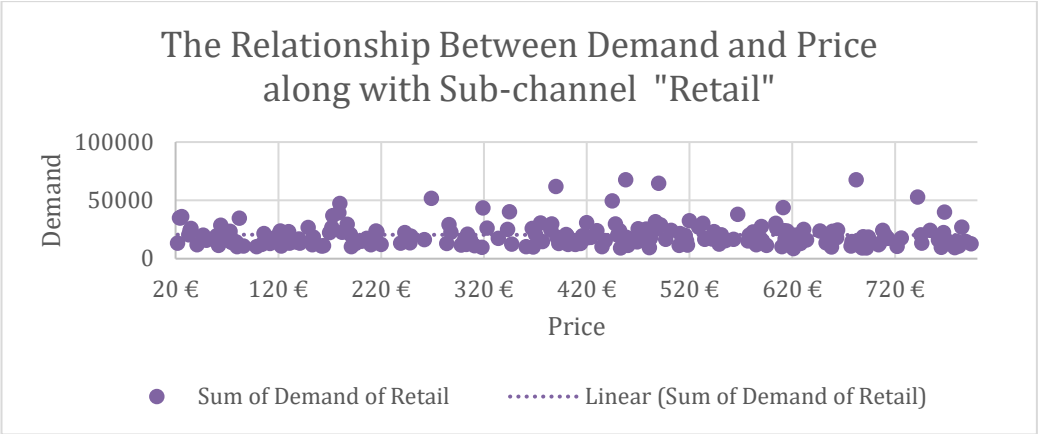


Figure 16. The Relationship Between Demand and Price along with Sub-channel "Retail".

Figure 17 shows the relationship between price and demand for the Analgesics product class. The scatter points are spread without a strong visible trend, and the regression line has a very slight downward slope. The calculated correlation is −0.096, suggesting a very weak negative association between price and demand. This indicates that higher prices are marginally linked to lower demand, though the effect is minimal.

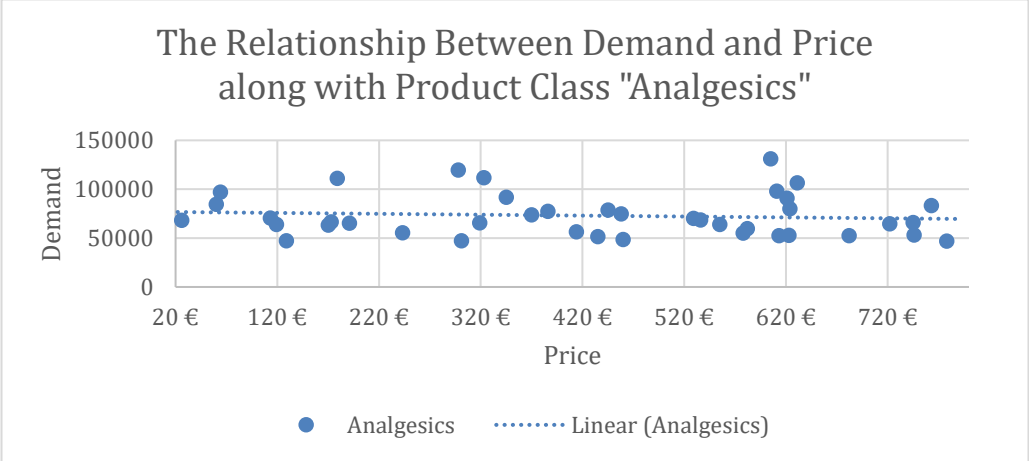


Figure 17. The Relationship Between Demand and Price along with Product Class "Analgesics".

Figure 18 illustrates the results for the Antibiotics class. The points are randomly spread, and the fitted regression line remains nearly flat. The correlation is **0.013**, confirming that there is no meaningful linear relationship between price and demand in this category.

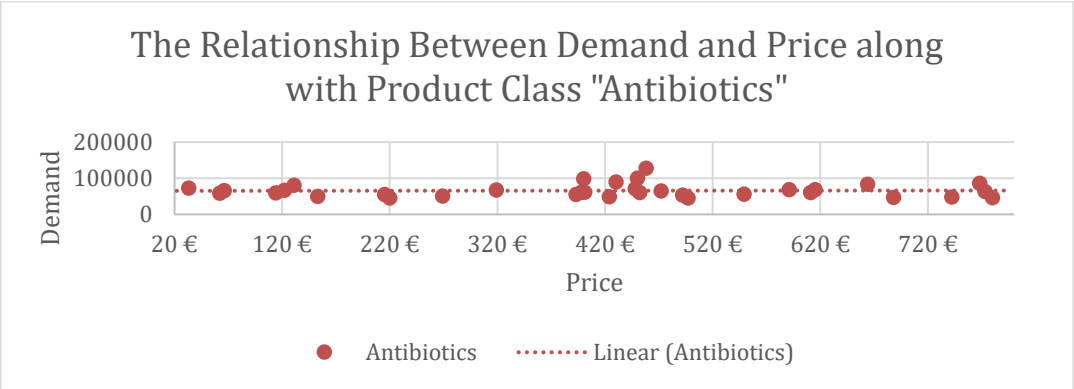


Figure 18. The Relationship Between Demand and Price along with Product Class "Antibiotics".

Figure 19 presents the case of the Antimalarial product class. The scatter points are dispersed without a clear pattern, though the regression line shows a very slight upward slope. The correlation is 0.053, indicating a very weak positive relationship where higher prices are slightly associated with higher demand, but the effect is negligible

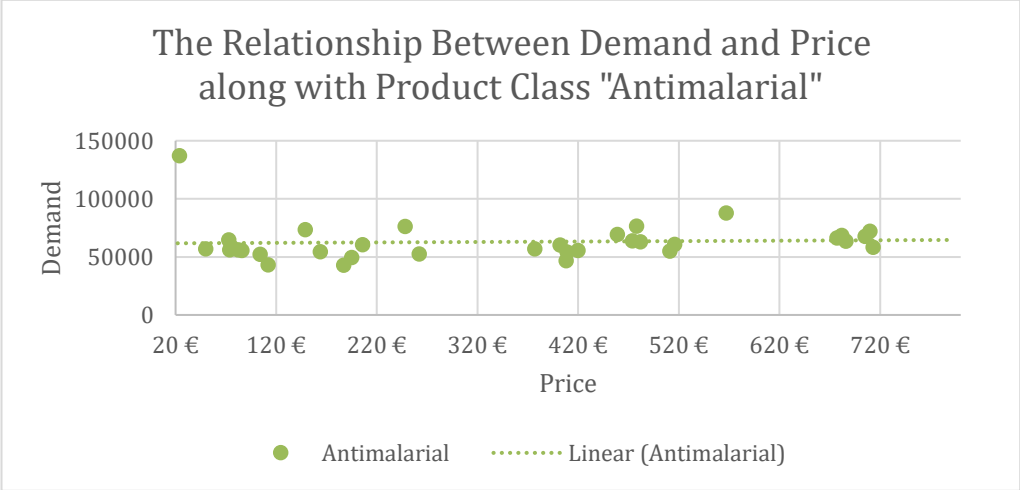


Figure 19. The Relationship Between Demand and Price along with Product Class "Antimalarial".

Figure 20 displays the scatter plot for Antipyretics class. The points are scattered without a strong trend, and the regression line shows a small downward slope. The correlation is −0.096, reflecting a very weak negative relationship, suggesting that higher prices are marginally linked to lower demand, though the effect is minimal.

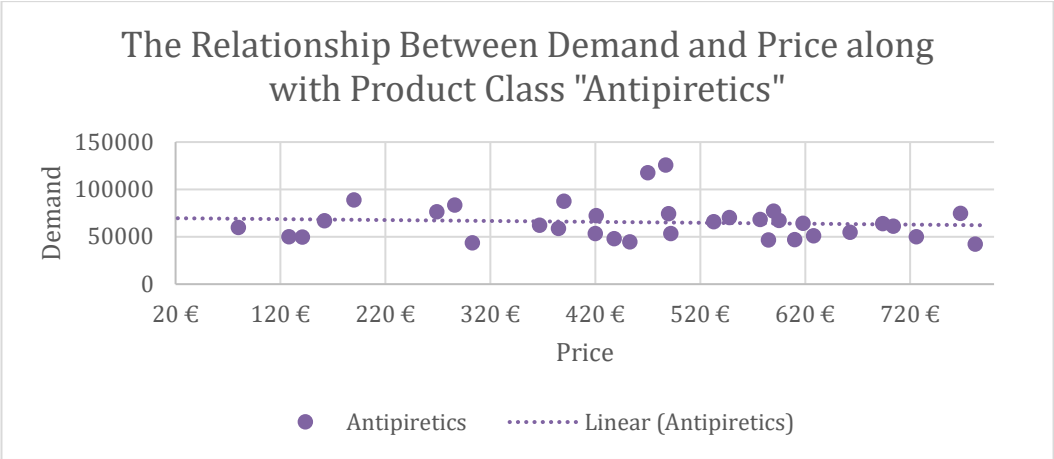


Figure 20. The Relationship Between Demand and Price along with Product Class "Antipiretics".

Figure 21 shows the relationship between price and demand for the Antiseptics class. The scatter points are randomly distributed, and the regression line is nearly flat with a slight downward tendency. The correlation is −0.050, again indicating a very weak negative relationship with negligible impact of price on demand

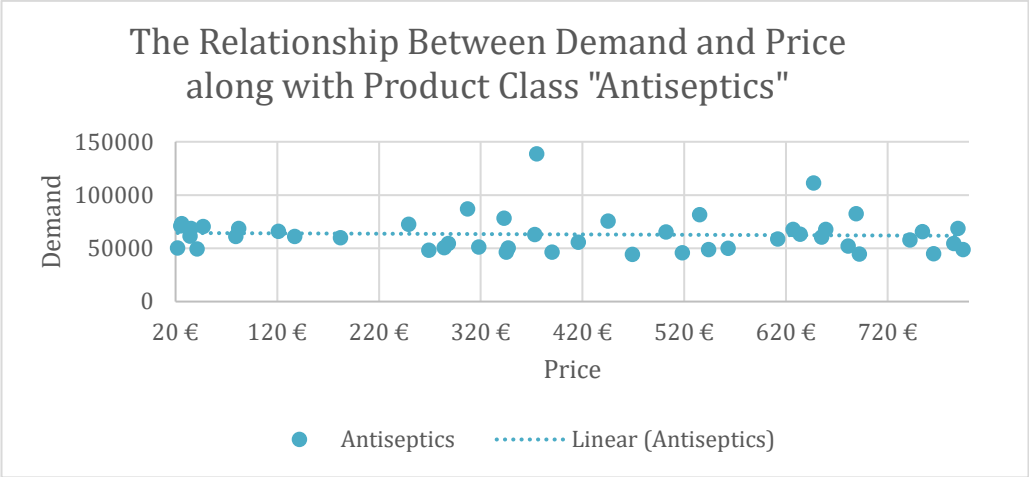


Figure 21. The Relationship Between Demand and Price along with Product Class "Antiseptics".

Figure 22 illustrates the results for the Mood Stabilizers product class. The points are spread without a strong visible trend, but the regression line shows a slight upward slope. The correlation is 0.057, reflecting a very weak positive association between price and demand, though the effect remains minimal.

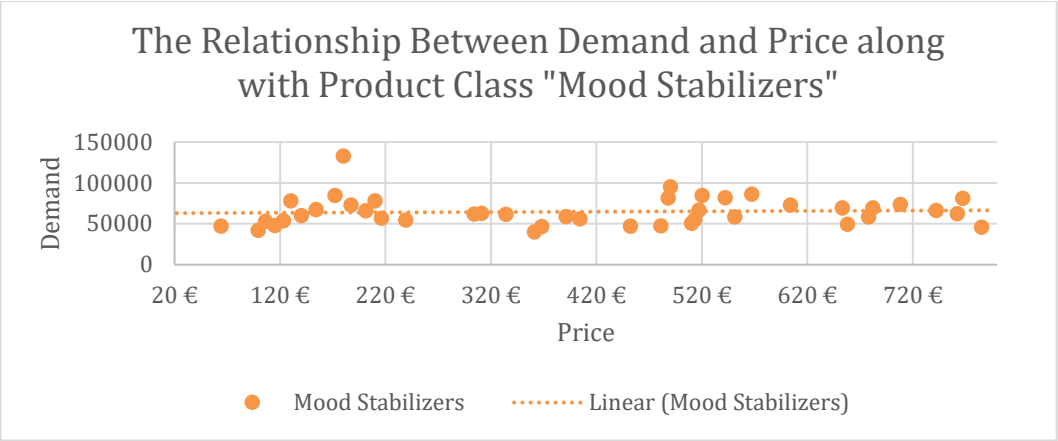


Figure 22. The Relationship Between Demand and Price along with Product Class "Mood Stabilizers".

Table 2 presents the monthly aggregated demand for six pharmaceutical product classes over a 24-month period, from January 2017 to December 2018, along with yearly and overall totals. The results reveal notable differences in demand levels across product categories. Analgesics exhibited the highest overall demand, reaching approximately 2.99 million units, followed very closely by Antiseptics with around 2.96 million units and Mood Stabilizers with 2.78 million units. In contrast, Antibiotics (2.23 million units) and Antipyretics (2.22 million units) recorded moderate but consistent demand, while Antimalarials showed the lowest cumulative demand at 2.14 million units.

A comparison of annual totals indicates a clear upward trend, with total demand increasing from about 6.63 million units in 2017 to 8.69 million units in 2018, reflecting an overall growth of nearly 30 percent. This rise suggests an expansion in pharmaceutical consumption across the study horizon.

Overall, the table underscores the relative dominance of certain product classes particularly Analgesics and Antiseptics while also providing evidence of increasing demand over time. These findings establish a foundation for subsequent forecasting, as they demonstrate both structural differences among product classes and temporal dynamics that justify the application of statistical and machine learning approaches in the predictive analysis.

Table 2. The monthly aggregated demand for each product class.

Product Class	Analgesics	Antibiotics	Antimalarial	Antipiretics	Antiseptics	Mood Stabilizers	Grand Total
Jan-17	55930	69291	50949	42182	68936	71217	358505
Feb-17	75220	98480	79127	82773	101933	107709	545242
Mar-17	108584	71297	72896	75789	106832	102542	537940
Apr-17	76799	85161	54044	49395	103488	87865	456752
May-17	117331	74460	66015	112922	119332	104620	594680
Jun-17	172206	80619	74287	87575	113936	137070	665693
Jul-17	120970	63108	87425	89486	128854	107495	597337
Aug-17	104604	78765	73444	86699	104589	101694	549795
Sep-17	110738	115046	64026	83600	128808	146063	648281
Oct-17	110407	70000	73444	90701	92971	99576	537099
Nov-17	107466	64935	81558	70311	138981	128379	591630
Dec-17	152274	73561	58955	69267	116637	79495	550189
2017 Total	1312529	944723	836170	940700	1325297	1273725	6633143
Jan-18	129757	74249	85501	89094	104497	108008	591106
Feb-18	131355	108813	94379	97722	118967	127521	678757
Mar-18	145871	146251	117251	160971	152007	156409	878760
Apr-18	104650	134850	91370	92539	156552	90408	670369
May-18	93136	104077	96442	81475	97020	97592	569742
Jun-18	196176	110351	188576	99853	151428	170416	916800
Jul-18	135265	116138	125135	94186	158376	138111	767211
Aug-18	132787	70961	102076	95066	154213	141257	696360
Sep-18	139291	93903	103430	107159	154086	117922	715791
Oct-18	186857	122317	127926	119660	150201	154035	860996
Nov-18	158693	98844	117586	157949	141227	131502	805801
Dec-18	124028	103604	53879	82965	100320	77025	541821
2018 Total	1677866	1284358	1303551	1278639	1638894	1510206	8693514
Grand Total	2990395	2229081	2139721	2219339	2964190	2783931	15326657

As part of the exploratory data analysis, box plots were applied to identify and remove outliers. because they did not represent the regular demand pattern and could distort the results. Keeping such extreme values would bias the models and reduce the accuracy of the forecasts. By excluding them, the dataset better reflects typical demand behavior, which provides a more reliable basis for analysis. Figure 23 and Figure 24 illustrate the distribution of demand before and after cleaning. The results show that several extreme values were excluded, resulting in a cleaner dataset that is more suitable for forecasting. After this step, the cleaned dataset was saved in a separate file and used consistently in both the statistical forecasting models and the machine learning models. This refinement ensured that all subsequent analyses were based on representative data without distortion from anomalous points.

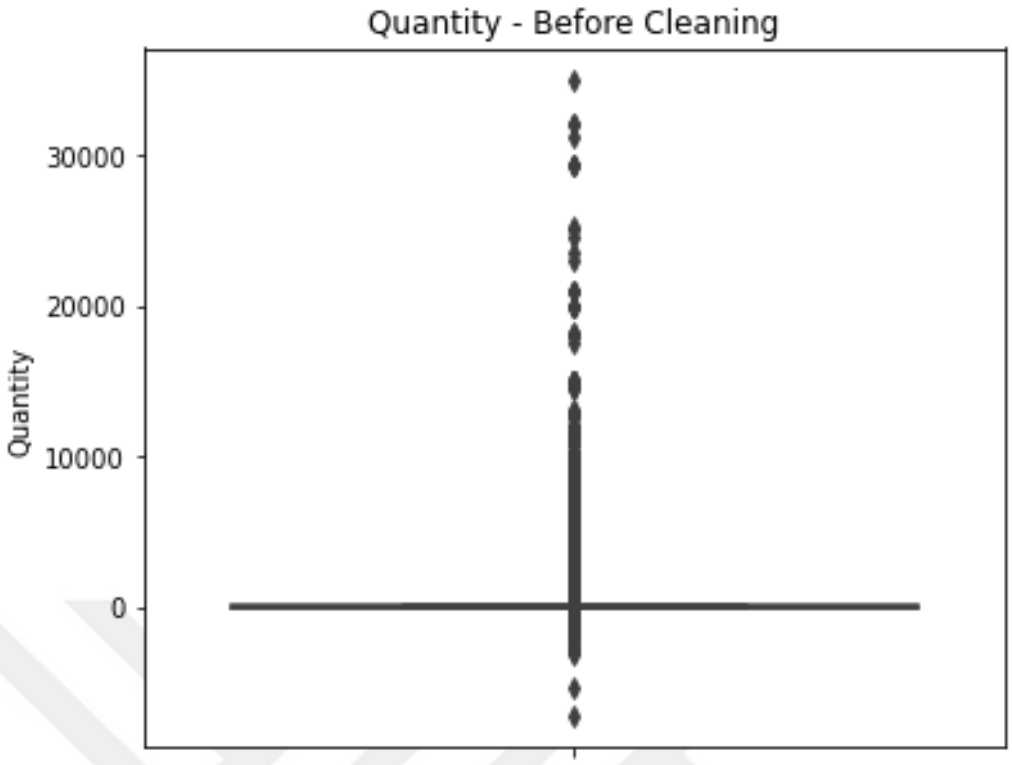


Figure 23. Distribution of demand before boxplot.

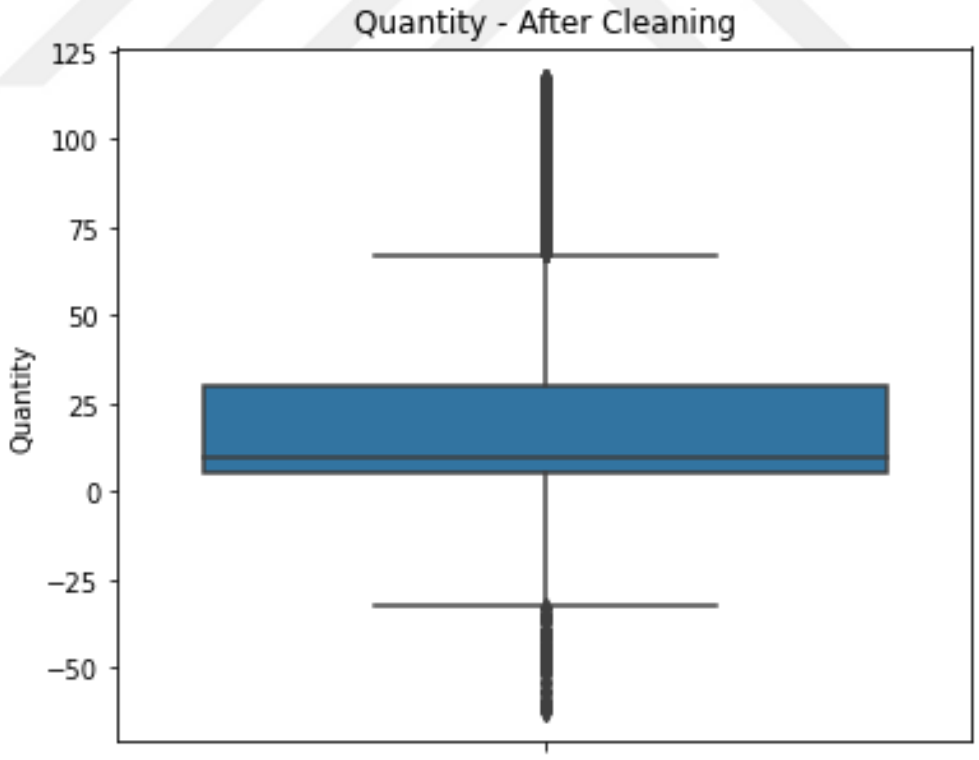


Figure 24. Distribution of demand after boxplot.

4.4. Statistical Models in the Case Study

The initial design planned for two temporal splits to evaluate the statistical models. The first split was intended to use 18 months for training and 6 months for testing; however, this configuration produced execution errors in Minitab due to row requirements and therefore could not be applied. As a result, the analysis proceeded with the second planned split of 21 months for training and 3 months for testing, which ensured compatibility with the software and allowed forecasts to be generated strictly on unseen data. Importantly, the data set used for model training and testing had first been refined through box plot analysis and outlier removal, ensuring that the forecasts were based on clean and representative data.

Multiple statistical techniques were then implemented in Minitab, including Moving Average, Single Exponential Smoothing, and ARIMA. Each model was trained exclusively on the designated training sets, and forecasts were generated for the testing horizons. The accuracy metrics (MAE, RMSE, MAPE, and R^2) were calculated by comparing the forecasted values against the actual observed demand, thereby providing an unbiased assessment of predictive performance.

Note: When attempting to implement SARIMA, the seasonal parameter s =12, which would normally be appropriate for monthly data across two years, could not be specified in Minitab due to software constraints. As a result, SARIMA could not be fully executed under the intended seasonal setting.

4.4.1. Moving Average

As illustrated in the Figure 25 3-month moving average was applied as a baseline forecasting model. By design, this approach smooths out short-term fluctuations and highlights the underlying demand trend. Using the adopted split of 21 months for training (January 2017 – September 2018) and 3 months for testing (October – December 2018), the model produced an MAE of 27,581.67, RMSE of 31,155.28, MAPE of 18.27%, and R^2 of –0.13.

These results indicate that while the moving average provides a simple and interpretable benchmark, its predictive capacity is modest. The negative R^2 suggests that the model performed worse than a naive mean predictor on the test set, reflecting its inability to capture sudden peaks, troughs, or more complex seasonal dynamics.

Overall, the moving average serves as a baseline reference but lacks the sophistication needed for accurate forecasting in volatile demand patterns.

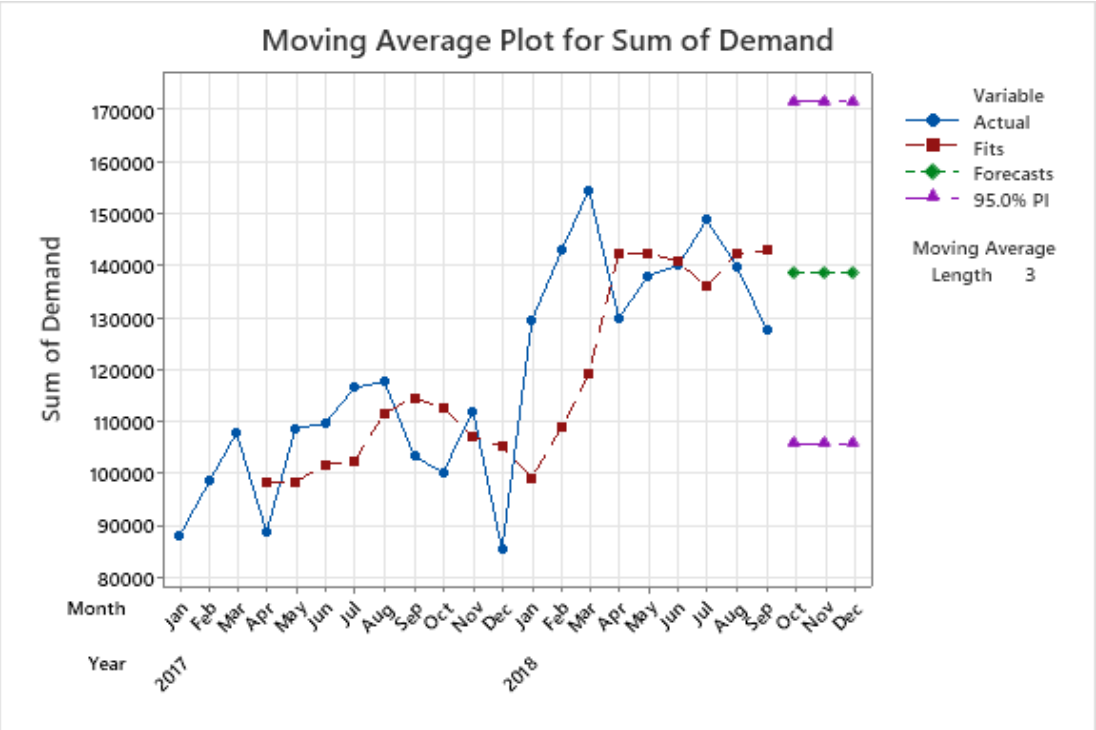


Figure 25. Moving Average plot for Sum of Demand.

4.4.2. Single Exponential Smoothing

As shown in Figure 26 Single Exponential Smoothing (SES) was applied with an optimized smoothing constant of α =0.577424. In Minitab, the smoothing constant α is automatically optimized by minimizing the RMSE, ensuring the best fit to the training data. Compared to the moving average, SES provides more flexibility by assigning greater weight to recent observations, allowing it to adjust more responsively to changes in demand. Using the adopted split of 21 months for training (January 2017 – September 2018) and 3 months for testing (October – December 2018), the model produced an MAE of 29,283.33, RMSE of 33,239.34, MAPE of 18.83%, and R^2 of –0.29.

These results indicate that while SES offers a modest improvement in adaptability compared to the moving average, its predictive performance remains weak in volatile demand conditions. The negative R^2 reflects poor generalization on the test set, suggesting that SES is unable to effectively capture abrupt fluctuations or more complex structures in the demand series. scenarios

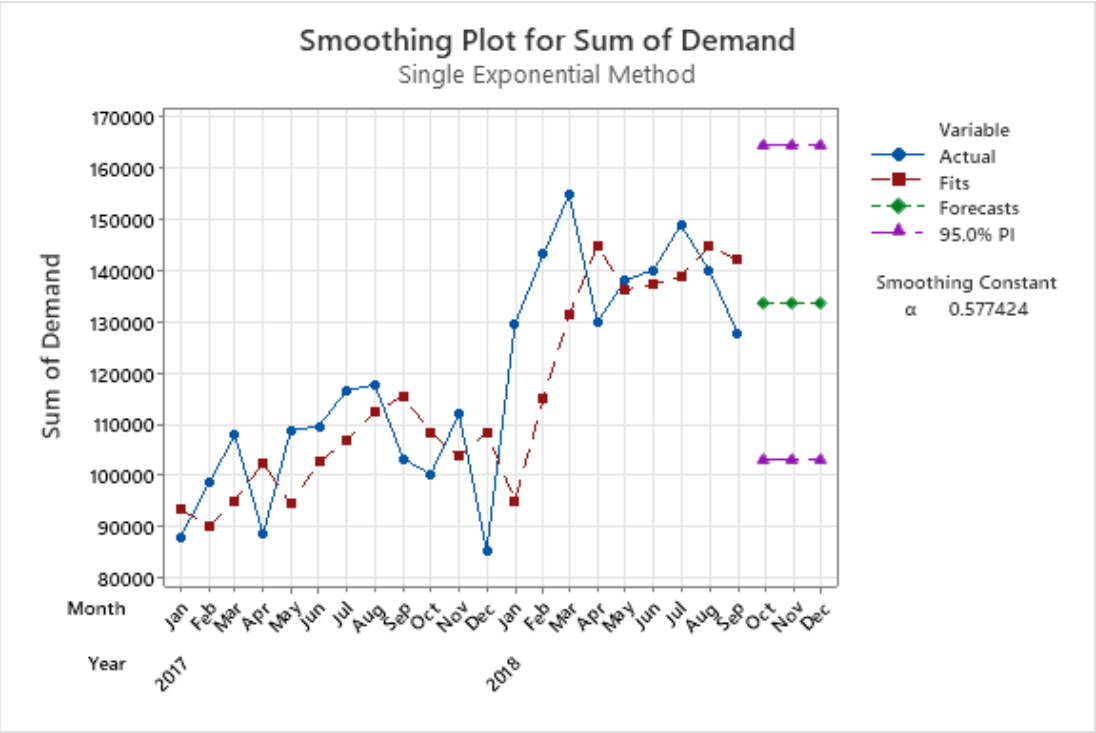


Figure 26. Smoothing Plot for Sum of Demand.

4.4.3. ARIMA

The ARIMA modeling process began with the Augmented Dickey–Fuller (ADF) test applied to the original demand series Figure 27, The null hypothesis of non-stationarity could not be rejected (p >0.05), indicating that the data was not stationary. The need for first-order differencing (d =1) was confirmed not only by the ADF test but also by the inspection of the ACF and PACF plots Figure 28. The slow decay in the ACF and the significant spike at lag 1 in the PACF both indicated non-stationarity in the original series, supporting the application of a first difference. After applying first-order differencing, the series passed the stationarity test (p<0.05), confirming its suitability for ARIMA modeling Figure 29 And Figure 30.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-2.18195	0.213	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 27. Augmented Dickey–Fuller (ADF) test before differencing: the series is non-stationary.

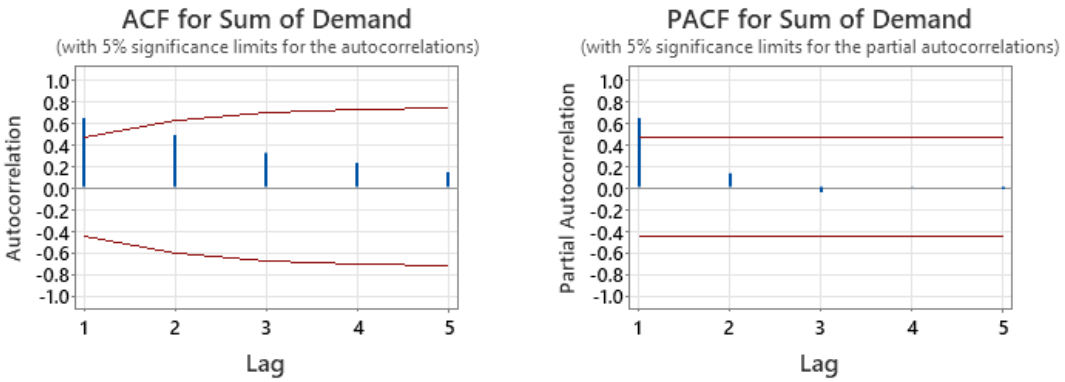


Figure 28. ACF and PACF plots of the original series.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-5.75430	0.000	Test statistic <= critical value of -3.03123. Significance level = 0.05 Reject null hypothesis. Data appears to be stationary, not supporting differencing.

Figure 29. Augmented Dickey–Fuller (ADF) test after first differencing.

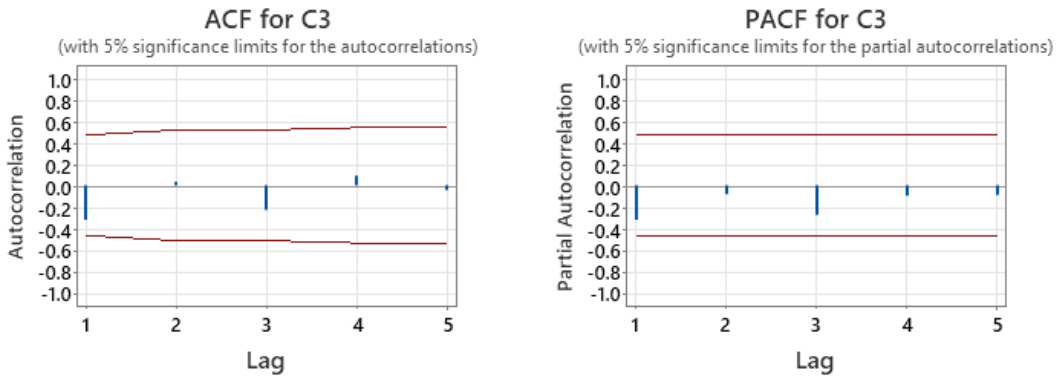


Figure 30. ACF and PACF plots of the series after first differencing

Once stationarity was achieved, the next step was to identify the most appropriate ARIMA specification. Candidate models were compared using the corrected Akaike Information Criterion (AICc), and the best-fitting model was selected as ARIMA (1,1,0) Figure 31.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 1, q = 0*	-221.490	447.687	446.981	448.972
p = 0, q = 1	-220.387	448.275	446.775	449.762
p = 0, q = 0	-222.295	449.296	448.590	450.582
p = 1, q = 1	-219.607	449.881	447.214	451.197
p = 0, q = 2	-219.650	449.967	447.300	451.283
p = 2, q = 0	-221.485	450.469	448.969	451.957
p = 1, q = 2	-219.769	453.824	449.539	454.517
p = 2, q = 1	-221.619	453.905	451.238	455.221

** Best model with minimum AICc. Output for the best model follows.*

Figure 31. ARIMA model selection based on AICc.

Finally, the chosen model was fitted on the training set and used to generate forecasts for the testing horizon Figure 32 shows the forecast results along with 95% confidence intervals. Using the adopted split of 21 months for training (January 2017 – September 2018) and 3 months for testing (October – December 2018), ARIMA (1,1,0) produced an MAE of 30,118.00, RMSE of 34,367.77, MAPE of 19.08%, and R^2 of –0.38.

While ARIMA captured the autoregressive structure more effectively than smoothing-based approaches, the negative R^2 reflects weak generalization on the test horizon.

This suggests that although ARIMA can model temporal dependencies, its predictive capacity remains limited under the volatility of the given dataset.

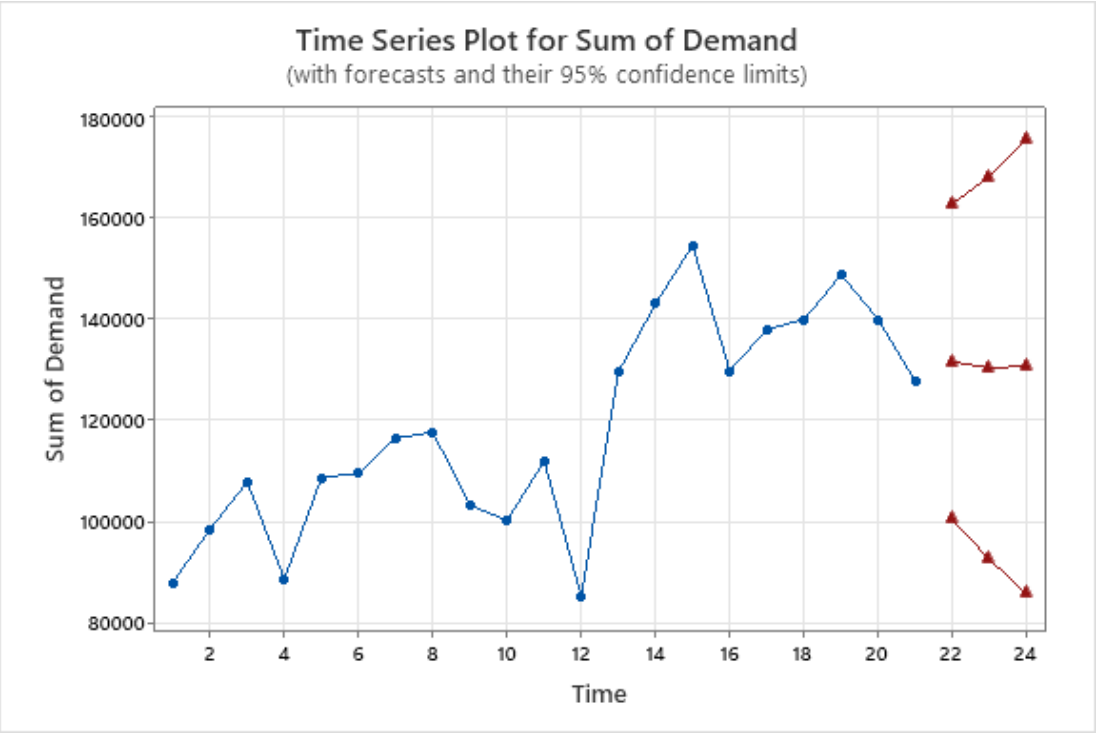


Figure 32. Forecast results from ARIMA (1,1,0)

4.4.4. SARIMA

The SARIMA model was also considered to explicitly capture seasonality, as the dataset spans two years of monthly demand data. A seasonal period of $s = 12$ would normally be appropriate; however, this configuration could not be executed in Minitab due to software constraints. This represents a technical limitation of the software rather than a methodological weakness. The SARIMA approach itself remains valid and applicable, and future work could revisit this model using more flexible platforms. Consequently, the analysis proceeded with non-seasonal ARIMA instead.

4.4.5. Evaluation of Statistical Models

As illustrated in Table 3, the comparative evaluation of statistical models revealed clear differences in their ability to capture the dynamics of pharmaceutical demand.

The 3-month Moving Average provided only a baseline level of accuracy. Its forecasts were smoother than the actual demand series, which limited its responsiveness to sudden peaks and troughs. The testing errors were moderate, with a MAPE of 18.27%

and R^2 of –0.13, confirming that the model captured only general trends but lacked predictive precision.

Single Exponential Smoothing (SES), optimized with a smoothing constant of

α =0.577424, did not offer substantial improvement over the moving average. The model produced a MAPE of 18.83% and R^2 of –0.29, indicating weak predictive capacity and highlighting that its constant smoothing factor could not effectively account for irregular fluctuations in demand.

The ARIMA (1,1,0) model showed slightly higher errors, with a MAPE of 19.08% and R^2 of –0.38, reflecting limited generalization and poor alignment with the observed demand series. Although ARIMA is theoretically capable of capturing autoregressive dynamics, the short time horizon and volatile patterns in the dataset constrained its performance.

In summary, none of the tested statistical models delivered strong predictive accuracy, with all exhibiting MAPE values near 18–19% and negative R^2 scores. These findings suggest that while statistical methods can provide a basic understanding of demand dynamics, their predictive power is limited when applied to short and volatile time series such as the one used in this study.

To evaluate these models, the Mean Absolute Percentage Error (MAPE) and the coefficient of determination (R^2) were selected as the primary error metrics. MAPE was chosen because it expresses forecast errors as a percentage of actual demand, making it easily interpretable in supply chain contexts. For example, a MAPE of 18% means that if true demand in a given month is 10,000 units, the forecast is on average off by about ±1,800 units. This relative measure not only provides intuitive managerial insight but also allows fair comparison across products and time horizons with different demand scales.

R^2 was included as a complementary indicator of explanatory power. While MAPE quantifies the magnitude of errors, R^2 reveals whether the model improves upon a naive mean predictor. Negative R^2 values, as observed in this study, indicate that the models failed to generalize well to unseen data.

By combining MAPE and R^2, the evaluation ensures both practical interpretability and statistical rigor. MAPE directly reflects forecast accuracy in operational terms, while R^2 assesses the models' ability to capture variance in the demand series. Together, they

provide a balanced framework for comparing statistical forecasting models with machine learning approaches.

In addition to the overall demand analysis, the same statistical Models procedures were replicated for each product class within the dataset. This allowed for a more granular understanding of demand dynamics across different product classes. The full set of results for these class-level analyses is documented in Appendix D.

Table 3. Statistical model results on the testing set.

Model	Train–Test Split	MAE	RMSE	MAPE	R^2
Moving Average (3)	**2017-01 → 2018-09 / 2018-10 → 2018-12**	27581.67	31155.28	18.27	-0.13
Single Exp. Smoothing (α=0.577424)	**2017-01 → 2018-09 / 2018-10 → 2018-12**	29283.33	33239.34	18.83	-0.29
ARIMA (1, 1, 0)	**2017-01 → 2018-09 / 2018-10 → 2018-12**	30118.00	34367.77	19.08	-0.38

4.5. Machine Learning in the Case Study

To assess the predictive performance of machine learning models in forecasting pharmaceutical demand, three regression-based algorithms were applied: Decision Tree Regressor, Random Forest Regressor, and Extreme Gradient Boosting (XGBoost). These algorithms were selected due to their ability to capture complex non-linear patterns and interactions across categorical and numerical predictors, which are characteristic of the pharmaceutical supply chain.

All models were trained on the cleaned dataset obtained after applying box plots and removing outliers, ensuring that the learning process was not biased by extreme values. Their performance was then evaluated by comparing the predicted values against the actual observations in the test sets using MAE, RMSE, MAPE, and R^2.

As explained in the statistical modeling section, the methodological design initially planned for two temporal splits to ensure robustness. However, due to the row limitations in Minitab, the first split (18 months training / 6 months testing) could not be executed. Therefore, for consistency across approaches, the second split (21 months

training / 3 months testing) was adopted as the standard evaluation framework for both statistical and machine learning models.

By applying the same cleaned dataset, the same split, and the same evaluation metrics, the comparative analysis between statistical and machine learning models remains fair, unbiased, and directly interpretable.

4.5.1. Data Preparation

The dataset contained 129,391 transaction-level records with 18 features. Prior to model development, the data set was refined by removing irrelevant or redundant variables such as 'Distributor', 'Customer Name', 'Latitude', 'Longitude', 'Name of Sales Rep', 'Manager', 'Sales Team', 'Sales' to avoid redundancy and potential data leakage. The cleaned dataset used for modeling was the same file obtained after outlier removal using box plots in the exploratory analysis stage.

Categorical variables including 'City', 'Country', 'Channel', 'Sub-channel', 'Product Name', 'Product Class', were transformed through label encoding to make them suitable for regression-based algorithms. The target variable was defined as Quantity, while predictors included time-related attributes (Month, Year), Price, and the encoded categorical features.

To improve predictive capacity, three lag features (Quantity_lag1, Quantity_lag2, Quantity_lag3) were engineered, enabling the models to exploit temporal dependencies in the demand sequence. After introducing the lagged variables, missing values resulting from shifting were dropped to ensure data consistency.

A correlation matrix was then generated Figure 33 to examine associations between predictors and the target variable. The results showed that the lagged demand variables had the strongest correlations with Quantity: Quantity_lag1 (0.35), Quantity_lag2 (0.28), and Quantity_lag3 (0.25). By contrast, Price exhibited almost no correlation with demand ($r \approx 0.00$), while categorical features such as City, Country, and Channel also demonstrated negligible or weak correlations.

This analysis confirms that historical demand patterns are the most influential predictors in the dataset, justifying the creation of lag features during feature engineering. It also highlights why tree-based machine learning algorithms (Decision

Tree, Random Forest, XGBoost) were applied, as they can effectively capture temporal dependencies while also modeling potential non-linear interactions among weaker predictors.

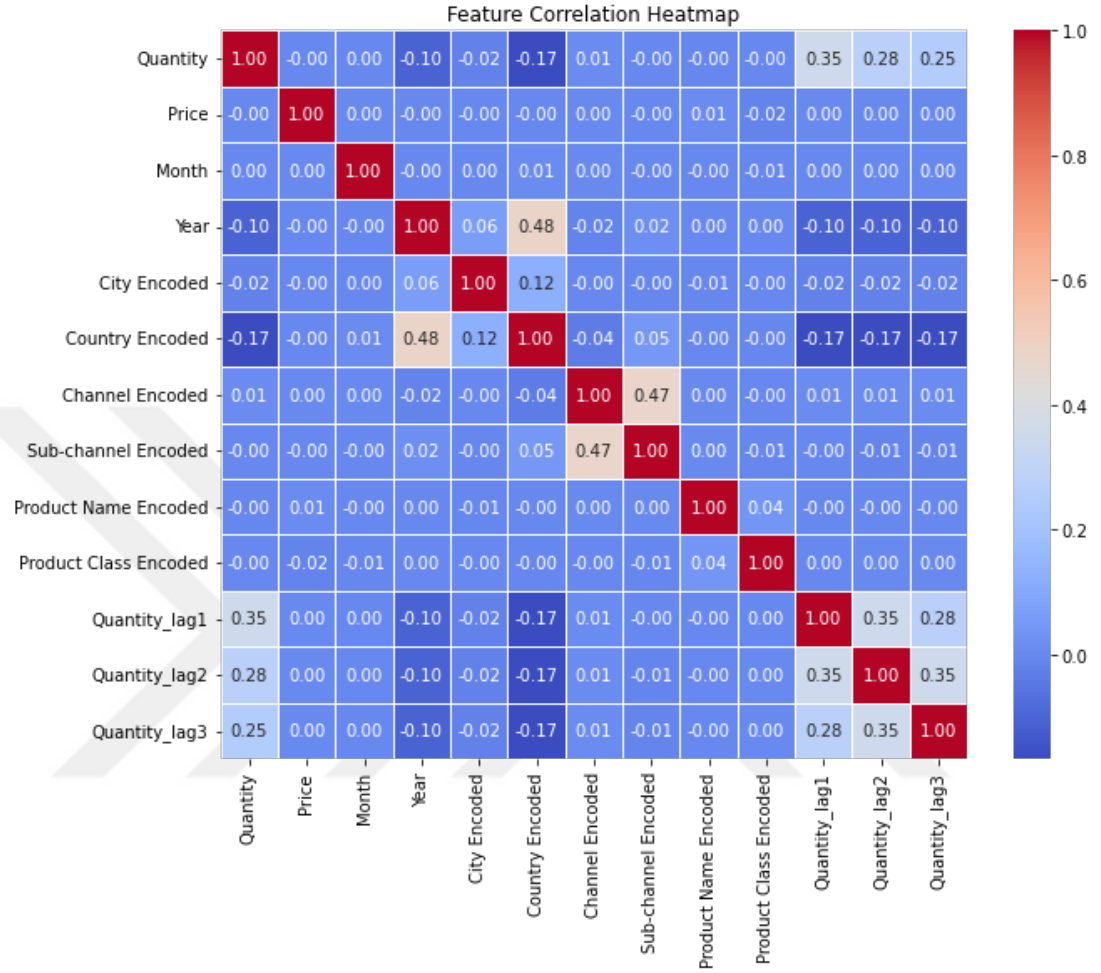


Figure 33. Correlation heatmap.

4.5.2. Training and Testing Design

A time-aware splitting strategy was implemented to ensure chronological consistency between training and testing data. The initial design planned for two temporal splits to enable robustness checks. However, due to the row limitations in Minitab that prevented the first split (18 months training / 6 months testing) from being executed, the analysis proceeded with the second split, which was also applied consistently to the machine learning models.

In this setup, the training set covered January 2017 to September 2018 (21 months), while the testing set spanned October to December 2018 (3 months). This design

ensured that forecasts were always generated on unseen future data, avoiding any overlap between training and testing sets.

The models were evaluated under two feature configurations: Without lag features, where predictors included encoded categorical features (City, Country, Channel, Sub-channel, Product Name, Product Class), time-related attributes (Month, Year), and Price. With lag features, where the above predictors were extended by three additional variables: Quantity_lag1, Quantity_lag2, and Quantity_lag3, representing historical demand in the previous three months.

After applying this design, the training set contained 107,494 rows and the testing set contained 21,894 rows for both configurations. While the baseline models (without lag) relied on structural and categorical predictors, the inclusion of lag features enabled the algorithms to exploit temporal dependencies in the demand sequence, potentially improving predictive accuracy.

4.5.3. Model Training

All three algorithms Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor were trained using Grid Search combined with TimeSeriesSplit (n_splits = 3) cross-validation to optimize hyperparameters while respecting the chronological order of the data. This setup avoids look-ahead bias by ensuring that, in every fold, the model is trained on earlier periods and validated on later, unseen periods.

Procedure. For each candidate hyperparameter configuration, TimeSeriesSplit generated three sequential folds. In fold k=3, the training window expanded cumulatively up to time t_k, and validation was performed on the immediately following horizon. The model's validation score was computed on each fold, and the average cross-validation score across the three folds was then used to rank configurations. In the implemented code (default scikit-learn settings), the score is R^2, so GridSearchCV selected the hyperparameters that achieved the highest average R^2 across the three time-ordered validations. This averaging provides a robust estimate of generalization, preventing tuning to a single validation period.

Search spaces. Hyperparameter grids were defined to balance model expressiveness and the dataset's size/structure. The search covered:

- Decision Tree Regressor
    - Without lags: max_depth ∈ {3}, min_samples_split ∈ {9} (selected)
    - With lags: max_depth ∈ {5}, min_samples_split ∈ {9} (selected)
- Random Forest Regressor
    - Without lags: n_estimators ∈ {60} and max_depth ∈ {5} (selected)
    - With lags: n_estimators ∈ {80} and max_depth ∈ {7} (selected)
- XGBoost Regressor
    - Without lags: n_estimators ∈ {40}, max_depth ∈ {3}, learning_rate ∈ {0.05} (selected)
    - With lags: n_estimators ∈ {60}, max_depth ∈ {3}, learning_rate ∈ {0.1} (selected)

Rationale. The grid search consistently favored shallow tree depths and moderate ensemble sizes, indicating that increasing complexity did not yield tangible gains given the data's characteristics. Using TimeSeriesSplit was essential to (i) mirror the real forecasting workflow (train on the past, validate on the future), and (ii) produce reliable, unbiased estimates of performance during hyperparameter tuning.

Data and configurations. All models were trained on the cleaned dataset obtained after box-plot outlier removal and evaluated under two feature settings: (1) without lag features, and (2) with lag to capture temporal dependencies. For comparability with the statistical models and due to the software limitation encountered in Minitab, the 21-month training / 3-month testing split was adopted as the unified evaluation design for machine learning as well.

Note. If an error-based criterion is preferred for selection, GridSearchCV can be configured with scoring (e.g., negative MAE/MAPE/RMSE). In this study, the default R^2 scoring was used, and the best_params_ correspond to the configuration with the highest mean R^2 across the three time-ordered folds.

4.5.4. Evaluation of Machine Learning Models

Predictions were post-processed to ensure interpretability by rounding to integer values and replacing negative outputs with zero demand. Model performance was assessed using MAE, RMSE, MAPE, and R^2, with the results summarized in Table 4

Table 4. Machine learning model results on the testing set.

Model	Lag Setting	Train–Test Split	MAE	RMSE	MAPE	R^2
Decision Tree	**No Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	18.41	25.36	328.81	0.02
Decision Tree	**With Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	16.00	23.17	246.34	0.19
Random Forest	**No Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	18.29	25.34	320.62	0.03
Random Forest	**With Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	15.76	22.97	237.0	0.20
XGBoost	**No Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	18.56	25.32	333.79	0.03
XGBoost	**With Lag**	**2017-01 → 2018-09 / 2018-10 → 2018-12**	15.74	22.95	234.13	0.20

The comparative evaluation showed that the incorporation of lag features consistently improved predictive accuracy across all three algorithms. For example, the Decision Tree model improved from a MAPE of 328.81% ($R^2 = 0.02$) without lag features to 246.34% ($R^2 = 0.19$) with lag features. Similarly, the Random Forest model decreased from 320.62% ($R^2 = 0.03$) without lag to 237.0% ($R^2 = 0.20$) with lag, while XGBoost achieved the best overall results, reducing the error from 333.79% ($R^2 = 0.03$) without lag to 234.13% ($R^2 = 0.20$) with lag. These findings confirm that incorporating temporal dependencies through lag variables substantially enhanced the ability of tree-based algorithms to capture demand dynamics.

Despite these improvements, the models' overall accuracy remained weak, as indicated by the still very high MAPE values (all above 230%). This limitation is largely attributable to the restricted set of predictors in the dataset. The available features were primarily categorical variables and price, which provided limited explanatory power. Moreover, the two-year time horizon was too short for the models to learn robust seasonal or long-term patterns. The transformation of categorical attributes through label encoding also reduced contextual richness, while external drivers of pharmaceutical demand such as market shocks, promotional activities, or supply-side disruptions were absent from the dataset.

In conclusion, while the inclusion of lag features improved performance and increased R^2 values to around 0.20, the persistently high MAPE scores indicate that the models captured only a portion of the demand dynamics, leaving a substantial share of variability unexplained.

All models were implemented in Python. The full code, including data Preparation, Training and Testing design, model training, and evaluation procedures, is provided in Appendix E.

4.6. Comparative Analysis in the Case Study

This sub-section presents a comparative evaluation of the statistical and machine learning models applied to the pharmaceutical demand case study. The analysis is based on error metrics obtained from the testing sets under a consistent train–test split (21 months training, 3 months testing), ensuring that the reported results reflect true out-of-sample forecasting performance. Although R^2 values were calculated and reported in the individual model results, the comparative analysis focused on MAE, RMSE, and MAPE, as these metrics provide a more direct and interpretable assessment of forecasting error.

Statistical Models. The statistical forecasting methods demonstrated stable and relatively accurate results. The 3-month Moving Average achieved MAPE of 18.27%, with MAE = 27,581.67 and RMSE = 31,155.28, providing a simple but informative baseline. Single Exponential Smoothing ($\alpha = 0.577424$) produced similar results, with a MAPE of 18.83% (MAE = 29,283.33; RMSE = 33,239.34). The ARIMA (1,1,0) model yielded a slightly higher MAPE of 19.08% (MAE = 30,118.00; RMSE = 34,367.77), showing no clear advantage over simpler smoothing methods. Despite differences in specification, all three statistical models consistently produced testing errors within the 18–19% MAPE range, highlighting their relative robustness.

Machine Learning Models. Machine learning approaches showed a stark contrast in performance depending on the use of lag features. Without lags, all models performed extremely poorly, with MAPE values exceeding 320% (Decision Tree = 328.81%, Random Forest = 320.62%, XGBoost = 333.79), confirming that categorical encodings and price alone could not explain demand variability. Incorporating lag features substantially improved performance: Decision Tree with lag reduced MAPE to

246.34% (MAE = 16.00; RMSE = 23.17). Random Forest with lag achieved a MAPE of 237.0% (MAE = 15.76; RMSE = 22.97). XGBoost with lag produced the best machine learning results, with a MAPE of 234.13% (MAE = 15.74; RMSE = 22.95).

Although lag-enhanced models outperformed their no-lag counterparts, their error rates remained an order of magnitude higher than the statistical models, indicating that machine learning could not achieve comparable reliability for this dataset.

4.6.1. Comparative Findings

Taken together, the testing set results clearly demonstrate that statistical models substantially outperformed machine learning models in this case study. The statistical approaches achieved MAPE values of about 18–19%, while even the best-performing machine learning models with lag features had MAPE values above 230%. The performance gap is largely attributable to the dataset's limitations: a short two-year time horizon, reliance on categorical variables and price as predictors, loss of contextual richness through label encoding, absence of external demand drivers such as market shocks, promotions, or supply disruptions.

Conclusion. Comparative analysis demonstrates that classical statistical approaches, particularly Moving Average and SES, ARIMA proved more reliable than advanced machine learning methods for this dataset. While lag-based machine learning models showed significant improvements relative to their no-lag versions, they were unable to match the accuracy of statistical forecasting. These findings emphasize the importance of dataset length, feature richness, and external variables in determining the effectiveness of machine learning relative to traditional statistical methods in pharmaceutical demand forecasting.

5. CONCLUSION

This thesis examined the problem of demand forecasting in pharmaceutical supply chains, an area of engineering management where accuracy is essential to prevent shortages, reduce excess inventory, and support effective decision-making. Both statistical and machine learning models were applied independently to a pharmaceutical case study to evaluate their forecasting performance under consistent conditions.

The findings revealed that statistical models, particularly Moving Average, SES, and ARIMA, provided the most reliable accuracy. All three delivered testing errors around 18–19% MAPE, despite negative R^2 values, highlighting their robustness even with a limited dataset. In contrast, machine learning models improved substantially when lag features were introduced, especially Random Forest and XGBoost, which achieved R^2 ≈0.20. However, their testing MAPE values remained extremely high (above 230%), making them less suitable for practical forecasting in this context.

From a managerial perspective, the study demonstrates that while lag features enhance temporal representation, classical series models remain more effective than machine learning models when data are short, categorical, and lack external drivers. The overall predictive performance was constrained by the two-year time horizon, reliance on encoded categorical and price variables, and the absence of explanatory factors such as promotions, regulatory changes, or supply-side disruptions.

It is therefore recommended that future research extend the dataset to cover longer historical periods, integrate richer explanatory variables (e.g., promotional activities, market shocks, or distribution constraints), and investigate hybrid approaches that combine the interpretability of statistical methods with the flexibility of machine learning techniques. Such advances will not only improve forecasting accuracy but also strengthen resilience and decision-making in pharmaceutical supply chains.

For pharmaceutical companies, these findings provide practical guidance: implementing statistical models such as ARIMA for short-term planning can reduce inventory costs, minimize shortages, and improve supply chain reliability, particularly when data availability is limited. More broadly, the demonstrated reliability of statistical methods under such conditions suggests that organizations can adopt them

as cost-effective and robust forecasting tools. Integrating these approaches into existing planning systems can further support inventory optimization and strengthen decision-making efficiency in pharmaceutical distribution.

REFERENCES

Aamer, Ammar & Yani, Luh Putu Eka & Priyatna, I Made. (2021). Data Analytics in the Supply Chain Management: Review of Machine Learning Applications in Demand Forecasting. Operations and Supply Chain Management An International Journal. 14. 1-13. 10.31387/oscm0440281.

Abbasimehr, Hossein & Shabani, Mostafa & Yousefi, Mohsen. (2020). An optimized model using LSTM network for demand forecasting. Computers & Industrial Engineering. 143. 106435. 10.1016/j.cie.2020.106435.

Abdulla, Ahmad & Baryannis, George & Badi, Ibrahim. (2019). Weighting the Key Features Affecting Supplier Selection using Machine Learning Techniques. 10.20944/preprints201912.0154.v1.

Abouloifa, Houria & Bahaj, Mohamed. (2023). Using Machine Learning Algorithms to Increase the Supplier Selection Process Efficiency in Supply Chain 4.0. 10.1007/978-3-031-26384-2_19.

Ali, Md & Nipu, Ashiquzzaman & Khan, Sharfuddin. (2023). A decision support system for classifying supplier selection criteria using machine learning and random forest approach. Decision Analytics Journal. 7. 100238. 10.1016/j.dajour.2023.100238.

Aliyeva, Xeniya & Memon, Shazim & Nazir, Kashif & Kim, Jong. (2024). Energy consumption forecasting in PCM-integration buildings considering building and environmental parameters for future climate scenarios. Energy. 310. 133248. 10.1016/j.energy.2024.133248.

Arif, Md. Ariful & Sany, Saiful & Nahin, Faiza & Rabby, Akm Shahariar Azad. (2019). Comparison Study: Product Demand Forecasting with Machine Learning for Shop. 171-176. 10.1109/SMART46866.2019.9117395.

Badr, H., & Ahmed, W. (2023). A comprehensive analysis of demand prediction models in supply chain management. American Journal of Industrial and

Business Management, 13(12), 1353–1376. https://doi.org/10.4236/ajibm.2023.1312075

Chandra, Charu & Grabis, Janis. (2005). Application of Multi-Steps Forecasting for Restraining the Bullwhip Effect and Improving Inventory Performance Under Autoregressive Demand. European Journal of Operational Research. 166. 337-350. 10.1016/j.ejor.2004.02.012.

Daeyoung Yoon, Seonyeong Park, Yechan Song et al. Methodology for Improving the Performance of Demand Forecasting Through Machine Learning, 02 March 2023, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-2637740/v1]

Dejonckheere, Jeroen & Disney, Stephen & Lambrecht, Marc & Towill, DR. (2001). Measuring and avoiding the bullwhip effect: a control theoretic approach. Katholieke Universiteit Leuven, Open Access publications from Katholieke Universiteit Leuven.

Devi, P. & Meesala, Srujitha & Reddy, Ramya & Senapathi, Kushal & Kolala, Udaya. (2023). Anticipating Consumer Demand using ML. International Journal for Research in Applied Science and Engineering Technology. 11. 1053-1058. 10.22214/ijraset.2023.50283.

Dou, Zixin & Sun, Yanming & Zhang, Yuan & Wang, Tao & Wu, Chuliang & Fan, Shiqi. (2021). Regional Manufacturing Industry Demand Forecasting: A Deep Learning Approach. Applied Sciences. 11. 6199. 10.3390/app11136199.

El Filali, A., Lahmer, E.H., El Filali, S., Kasbouya, M., Ajouary, M.A., & Akantous, S. (2022). Machine Learning Applications in Supply Chain Management: A Deep Learning Model Using an Optimized LSTM Network for Demand Forecasting. International Journal of Intelligent Engineering and Systems.

Elmir, Walid & Allaoua, Hemmak & Senouci, Benaoumeur. (2023). Smart Platform for Data Blood Bank Management: Forecasting Demand in Blood Supply Chain Using Machine Learning. Information. 14. 31. 10.3390/info14010031.

Filali, Aicha & Benlahmar, EL Habib & el Filali, Sanaa & Aissam, Jadli. (2022). Application of Deep Learning in the Supply Chain Management: A comparison

of forecasting demand for electrical products using different ANN methods. 1-7. 10.1109/ICECET55527.2022.9872903.

Forrester, J. W. (1997). Industrial Dynamics. Journal of the Operational Research Society, 48(10), 1037–1041. https://doi.org/10.1057/palgrave.jors.2600946

Harikrishnakumar, Ramkumar & Dand, Alok & Nannapaneni, Saideep & Krishnan, Krishna. (2020). Supervised Machine Learning Approach for Effective Supplier Classification. 10.1109/ICMLA.2019.00045.

Impact of Big Data on Supply Chain Performance through Demand Forecasting. (2023). International Journal of Computations, Information and Manufacturing (IJCIM), 3(1), 19-26. https://doi.org/10.54489/ijcim.v3i1.232

Kamal, Esraa & Abdel-Gawad, Amal & Ibraheem, Basem & Zaki, Shereen. (2023). Machine Learning Fusion and Data Analytics Models for Demand Forecasting in the Automotive Industry: A Comparative Study. Fusion: Practice and Applications. 12. 24-37. 10.54216/FPA.120102.

Kandananond, Karin. (2012). A Comparison of Various Forecasting Methods for Autocorrelated Time Series. International Journal of Engineering Business Management. 4. 1. 10.5772/51088.

Kara, Ahmet. (2021). A data-driven approach based on deep neural networks for lithium-ion battery prognostics. Neural Computing and Applications. 33. 1-14. 10.1007/s00521-021-05976-x.

Kara, Ahmet. (2022). Multi-scale deep neural network approach with attention mechanism for remaining useful life estimation. Computers & Industrial Engineering. 169. 108211. 10.1016/j.cie.2022.108211.

Khaldi, Rohaifa & El Afia, Abdellatif & Chiheb, Raddouane & Faizi, Rdouan. (2017). Artificial Neural Network Based Approach for Blood Demand Forecasting: Fez Transfusion Blood Center Case Study. 1-6. 10.1145/3090354.3090415.

Lee, Hau & Padmanabhan, V. & Whang, Seungjin. (2004). Information Distortion in a Supply Chain: The Bullwhip Effect. Management Science. 43. 546-558. 10.1287/mnsc.1040.0266.

Lei, Yang & Qiaoming, Hou & Tong, Zhao. (2023). Research on Supply Chain Financial Risk Prevention Based on Machine Learning. Computational Intelligence and Neuroscience. 2023. 1-15. 10.1155/2023/6531154.

Lin, Haifeng & Lin, Ji & Wang, Fang. (2022). An innovative machine learning model for supply chain management. Journal of Innovation & Knowledge. 7. 100276. 10.1016/j.jik.2022.100276.

Lingelbach, Katharina & Lingelbach, Yannick & Otte, Sebastian & Bui, Michael & Künzell, Tobias & Peissner, Matthias. (2021). Demand Forecasting Using Ensemble Learning for Effective Scheduling of Logistic Orders. 10.1007/978-3-030-80624-8_39.

Ma, Xiaoya & Li, Mengxiu & Tong, Jin & Feng, Xiaying. (2023). Deep Learning Combinatorial Models for Intelligent Supply Chain Demand Forecasting. Biomimetics. 8. 312. 10.3390/biomimetics8030312.

Martins, Emerson & Galegale, Napoleão. (2023). Sales forecasting using machine learning algorithms. Revista de Gestão e Secretariado (Management and Administrative Professional Review). 14. 11294-11308. 10.7769/gesec.v14i7.1670.

Moalem, Sepehr & Ahari, Roya & Shahgholian, Ghazanfar & Moazzami, Majid & Kazemi, Seyed. (2022). Long-Term Electricity Demand Forecasting in the Steel Complex Micro-Grid Electricity Supply Chain—A Coupled Approach. Energies. 15. 7972. 10.3390/en15217972.

Nassibi, Nouran & Fasihuddin, Heba & Hsairi, Lobna. (2023). Demand Forecasting Models for Food Industry by Utilizing Machine Learning Approaches. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.01403101.

Nguyen, Tuan & Haider, Mahfuz & Jisan, Afjal & Raju, Md Azad Hossain & Imam, Touhid & Khan, Md & Al-Jarf, Abdullah. (2024). Product Demand Forecasting with Neural Networks and Macroeconomic Indicators: A Comparative Study among Product Categories. Journal of Business and Management Studies. 6. 170-175. 10.32996/jbms.2024.6.2.17.

P, Anish & S, Hema & S.J, Jothika & M, Manochitra. (2021). Predicting the Demand for Fmcg using Machine Learning. International Journal of Engineering and Advanced Technology. 10. 169-171. 10.35940/ijeat.C2253.0210321.

Praveenadevi, D. & Sreekala, S.P. & Girimurugan, B. & Teja, K & Kamal, G. & Chandra, Asturi. (2023). An Enhanced Method on Using Deep Learning Techniques in Supply Chain Management. 210-213. 10.1109/ICDT57929.2023.10151338.

Raghunathan, S. Interorganizational Collaborative Forecasting and Replenishment Systems and Supply Chain Implications. Decis. Sci. 1999, 30, 1053–1071, https://doi.org/10.1111/j.1540-5915.1999.tb00918.x.

Raizada, Stuti & Saini, Jatinderkumar. (2021). Comparative Analysis of Supervised Machine Learning Techniques for Sales Forecasting. International Journal of Advanced Computer Science and Applications. 12. 10.14569/IJACSA.2021.0121112.

S, Praveena & S, Prasanna. (2022). A Hybrid Demand Forecasting for Intermittent Demand Patterns using Machine Learning Techniques. 557-561. 10.1109/ICCST55948.2022.10040407.

Saadallah, A., Jakobs, M., & Morik, K. (2021). Explainable Online Deep Neural Network Selection Using Adaptive Saliency Maps for Time Series Forecasting. ECML/PKDD.

Saadallah, Amal & Jakobs, Matthias & Morik, Katharina. (2022). Explainable online ensemble of deep neural network pruning for time series forecasting. Machine Learning. 111. 1-29. 10.1007/s10994-022-06218-4.

Saglam, Mustafa & Spataru, Catalina & Karaman, Ömer. (2023). Forecasting Electricity Demand in Turkey Using Optimization and Machine Learning Algorithms. Energies. 16. 4499. 10.3390/en16114499.

Shibu, N.B., & Agarwal, R. (2023). Analysing and Visualising trends for Supply Chain Demand Forecasting. 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), 901-906.

Singh, Neelam & Mittal, Varsha & Rawat, Vandana & Malik, Preeti & Rana, Aayushi. (2023). Research Trends in Supply Chain Management: A Machine Learning Perspective. 988-992. 10.1109/CISES58720.2023.10183623.

Singha, D & Panse, Chetan. (2022). Application of different Machine Learning models for Supply Chain Demand Forecasting: Comparative Analysis. 312-318. 10.1109/ICIPTM54933.2022.9753864.

Su, Xin & Huang, Shanshan. (2023). An improved machine learning model Shapley value-based to forecast demand for aquatic product supply chain. Frontiers in Ecology and Evolution. 11. 10.3389/fevo.2023.1160684.

Terrada, Loubna & El Khaili, M. & Hassan, Ouajji. (2022). Demand Forecasting Model using Deep Learning Methods for Supply Chain Management 4.0. International Journal of Advanced Computer Science and Applications. 13. 10.14569/IJACSA.2022.0130581.

Vithitsoontorn, Chayuth & Chongstitvatana, Prabhas. (2022). Demand Forecasting in Production Planning for Dairy Products Using Machine Learning and Statistical Method. 1-4. 10.1109/iEECON53204.2022.9741683.

Wang, Jiaxing & Liu, G. & Liu, Lu. (2019). A Selection of Advanced Technologies for Demand Forecasting in the Retail Industry. 317-320. 10.1109/ICBDA.2019.8713196.

Younis, Hassan & Sundarakani, Balan & Alsharairi, Malek. (2021). Applications of Artificial Intelligence and Machine Learning within Supply Chains: Systematic review and future research directions. Journal of Modelling in Management. ahead-of-print. 10.1108/JM2-12-2020-0322.

Younis, Hassan & Sundarakani, Balan & Alsharairi, Malek. (2021). Applications of Artificial Intelligence and Machine Learning within Supply Chains: Systematic review and future research directions. Journal of Modelling in Management. ahead-of-print. 10.1108/JM2-12-2020-0322.

APPENDIX

APPENDIX A: The dataset used in this study was obtained from Kaggle and contains detailed records of pharmaceutical demand. It includes 150,920 rows and 18 columns, each representing key variables related to medicine distribution and demand forecasting.

Dataset Title: Pharma_Sales_Dataset

Source: Kaggle

Link: https://www.kaggle.com/datasets/qinsights/pharma-sales-dataset

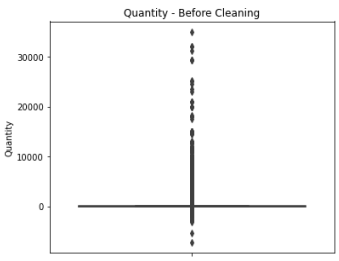
APPENDIX B: Outlier Detection and Removal using Boxplot. Before training statistical forecasting models and machine learning models, a boxplot method was applied to identify and remove outliers from the dataset. This preprocessing step was necessary to minimize the impact of extreme values and to ensure more reliable forecasting results.

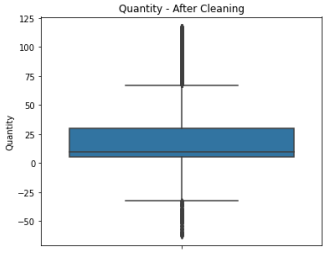
In [2]:
```
#Load dataset.
df = pd.read_csv(r"C:\Users\pc\OneDrive - Istanbul Kultur Universitesi\Safa Alreaai\Thesis Report Template\Dataset and Data Analytics\pharma-data.csv")
df.head()
```

Out[2]:

	Distributor	Customer Name	City	Country	Latitude	Longitude	Channel	Sub-channel	Product Name	Product Class	Quantity	Price	Sales	Month	Year	Name of Sales Rep	Manager	Sales Team
0	Gerlach LLC	Fadel-West Pharmacy	Ditzingen	Germany	48.8264	9.0667	Hospital	Government	Exotropin Empizine	Mood Stabilizers	20	785	15700.0	January	2017	Sheila Stones	Britanny Bold	Delta
1	Gerlach LLC	Nader-Gaylord Pharmacy	Backnang	Germany	48.9464	9.4306	Hospital	Private	Robapril	Antipiretics	10	453	4530.0	January	2017	Stella Given	Alisha Cordwell	Charlie
2	Gerlach LLC	McKenzie-Zemlak Pharm	Rheinbach	Germany	50.6256	6.9491	Pharmacy	Institution	Secrelazine Insonamic	Antipiretics	25	694	17350.0	January	2017	Daniel Gates	Alisha Cordwell	Charlie
3	Gerlach LLC	Fritsch-Hauck Pharmaceutical Ltd	Fürth	Germany	49.4783	10.9903	Pharmacy	Retail	Megenorphine	Antimalarial	5	402	2010.0	January	2017	Mary Gerrard	Britanny Bold	Delta
4	Gerlach LLC	Bernier, Murphy and Rau Pharma Plc	Geldern	Germany	51.5197	6.3325	Hospital	Government	Agalsiline	Mood Stabilizers	20	64	1280.0	January	2017	Mary Gerrard	Britanny Bold	Delta

In [3]:
```
#Step 1: Plot Quantity before cleaning.
plt.figure(figsize=(6,5))
sns.boxplot(y=df["Quantity"])
plt.title("Quantity - Before Cleaning")
plt.show()

#Step 2: Remove outliers.
Q1 = df["Quantity"].quantile(0.25)
Q3 = df["Quantity"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df_clean = df[(df["Quantity"] >= lower_bound) & (df["Quantity"] <= upper_bound)]

#Step 3: Plot Quantity after cleaning.
plt.figure(figsize=(6,5))
sns.boxplot(y=df_clean["Quantity"])
plt.title("Quantity - After Cleaning")
plt.show()

#Step 4: Save cleaned file.
df_clean.to_csv(r"C:\Users\pc\OneDrive - Istanbul Kultur Universitesi\Safa Alreaai\Thesis Report Template\Dataset and Data Analytics\pharma-data-clean.csv", index=False)
print("After cleaning:", df_clean.shape)
```






```
After cleaning: (129391, 18)
```

In [4]:
```
df=df_clean
```

APPENDIX C: Python Code for Statistical Models (Error Metrics Calculation) The following Python script was used to calculate the error metrics (MAE, RMSE, MAPE, and R^2) for the statistical forecasting models. This ensured that the evaluation of model performance was consistent and standardized across all methods.

In [35]:

```
rows = [
    ("17-Jan", 87986,   None,   None,   None),
    ("17-Feb", 98517,   None,   None,   None),
    ("17-Mar", 107849,  None,   None,   None),
    ("17-Apr", 88653,   None,   None,   None),
    ("17-May", 108680,  None,   None,   None),
    ("17-Jun", 109594,  None,   None,   None),
    ("17-Jul", 116502,  None,   None,   None),
    ("17-Aug", 117662,  None,   None,   None),
    ("17-Sep", 103240,  None,   None,   None),
    ("17-Oct", 100219,  None,   None,   None),
    ("17-Nov", 111951,  None,   None,   None),
    ("17-Dec", 85294,   None,   None,   None),
    ("18-Jan", 129595,  None,   None,   None),
    ("18-Feb", 143010,  None,   None,   None),
    ("18-Mar", 154612,  None,   None,   None),
    ("18-Apr", 129792,  None,   None,   None),
    ("18-May", 138029,  None,   None,   None),
    ("18-Jun", 139946,  None,   None,   None),
    ("18-Jul", 148780,  None,   None,   None),
    ("18-Aug", 139941,  None,   None,   None),
    ("18-Sep", 127718,  None,   None,   None),
    ("18-Oct", 185094, 138813, 133708, 131593),
    ("18-Nov", 149792, 138813, 133708, 130365),
    ("18-Dec", 113328, 138813, 133708, 130754),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand","MA(3)","SES (α=0.577424)","ARIMA(1,1,0)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
def safe_mape(y_true, y_pred):
    """MAPE that safely ignores zero targets to avoid division by zero."""
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R2 for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )


split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand"].to_numpy()
```

```
        R2   = r2_score(y_true, y_pred),
    )


split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand"].to_numpy()


#Compute metrics per model column
rows_out = []
for col in ["MA(3)", "SES (α=0.577424)", "ARIMA(1,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    metrics = compute_metrics(y_true, y_pred)

    rows_out.append({
        "Title": f"{col} • Univariate • Train {split_train} | Test {split_test}",
        "MAE": None if pd.isna(metrics["MAE"]) else round(metrics["MAE"], 2),
        "RMSE": None if pd.isna(metrics["RMSE"]) else round(metrics["RMSE"], 2),
        "MAPE (%)": None if pd.isna(metrics["MAPE"]) else round(metrics["MAPE"], 2),
        "R²": None if pd.isna(metrics["R2"]) else round(metrics["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                Title      MAE     RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 27581.67 31155.28     18.27 -0.13
SES (α=0.577424) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 29283.33 33239.34     18.83 -0.29
    ARIMA(1,1,0) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 30118.00 34367.77     19.08 -0.38
```

APPENDIX D: Statistical Analysis by Product Class For each product class, statistical forecasting models were applied independently to evaluate demand patterns. The models tested included Moving Average (MA), Single Exponential Smoothing (SES), and ARIMA. Two train–test scenarios were initially designed: Scenario 1: 18 months for training and 6 months for testing. Scenario 2: 21 months for training and 3 months for testing. Due to row limitations in Minitab, Scenario 1 could not be executed successfully, and the analysis therefore proceeded with Scenario 2, which ensured compatibility and allowed forecasts to be generated strictly on unseen data. While ARIMA models were successfully estimated for each product class, SARIMA could not be implemented under the intended seasonal specification (s =12) because of software restrictions in Minitab. As a result, the statistical evaluation focused on MA, SES, and ARIMA models. All model fitting and forecasting were performed in Minitab, while Python scripts were used exclusively to calculate the error metrics (MAE, RMSE, MAPE, and R^2) to ensure accuracy and consistency across product classes. The results for each product class are presented in the following appendix, including fitted models, testing errors, and graphical forecasts

D.1. Analgesics Class:

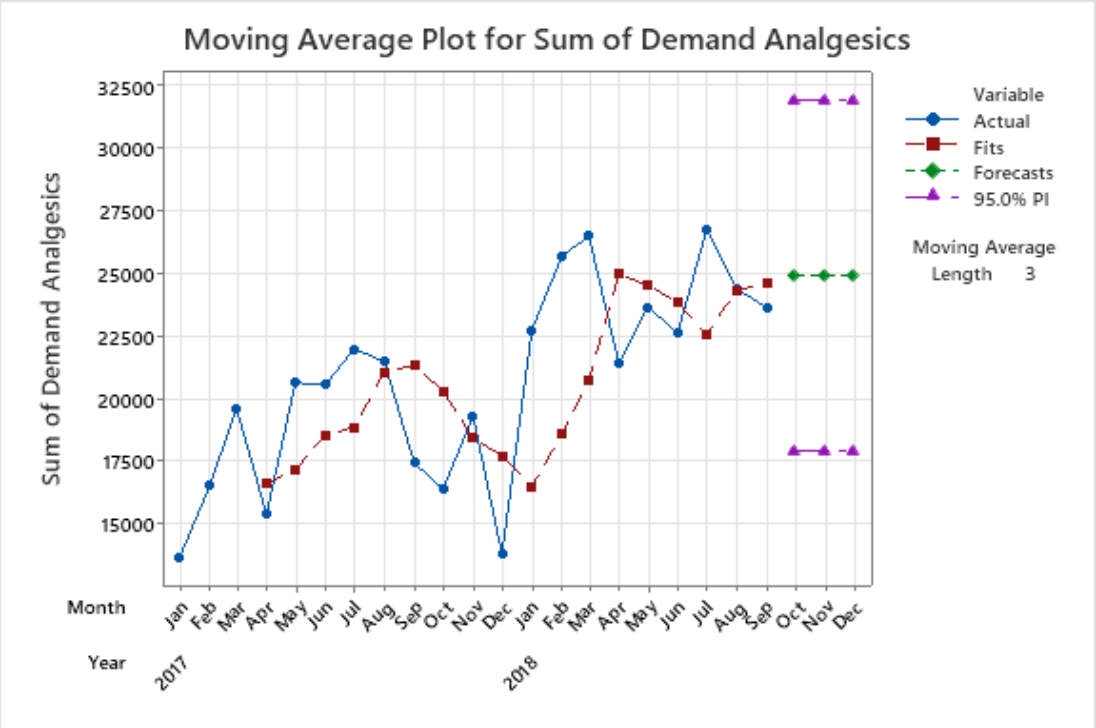


Figure 34. Moving Average Plot for Sum of Demand Analgesics.

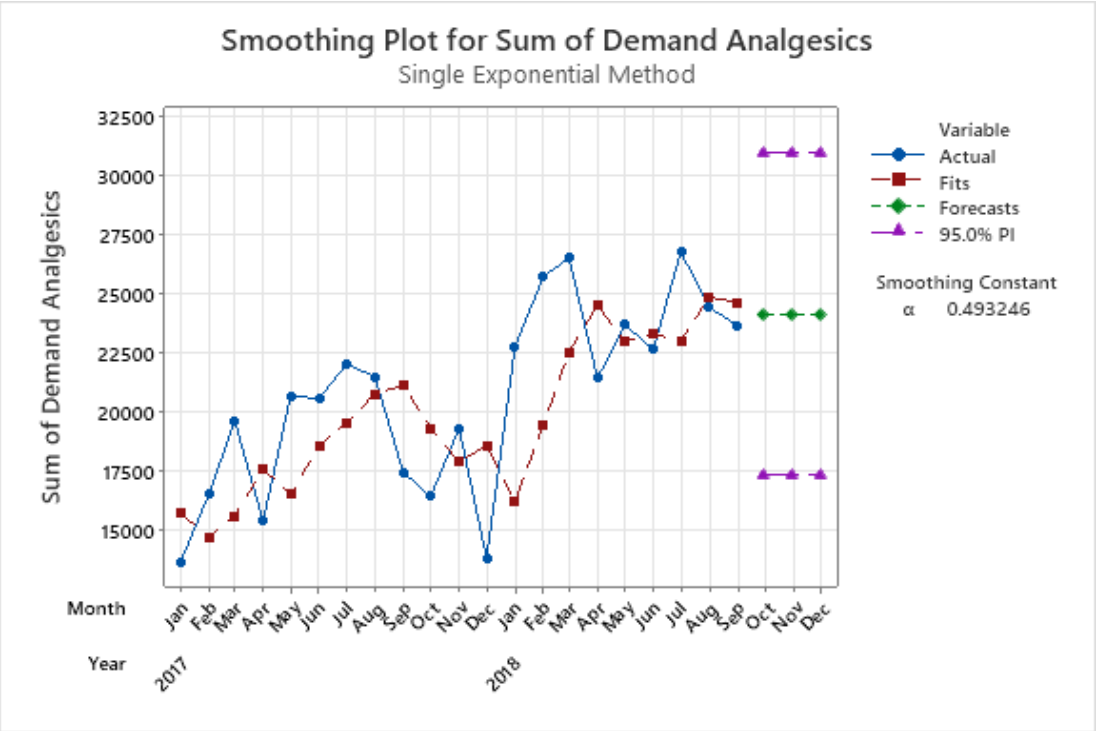


Figure 35. Smoothing Plot for Sum of Demand Analgesics.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-2.72176	0.070	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 36. ADF Test of Demand Analgesics before differencing.

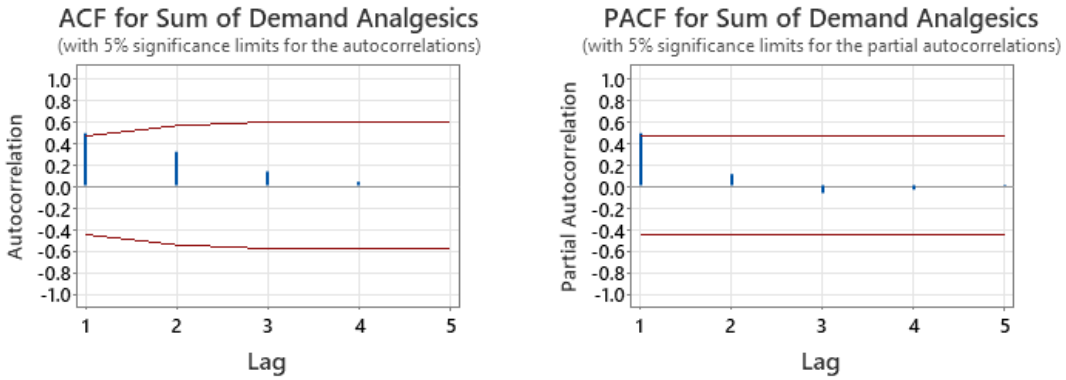


Figure 37. ACF & PACF of Demand Analgesics before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-6.10583	0.000	Test statistic <= critical value of -3.03123.
		Significance level = 0.05
		Reject null hypothesis.
		Data appears to be stationary, not supporting differencing.

Figure 38. ADF Test of Demand Analgesics after differencing.

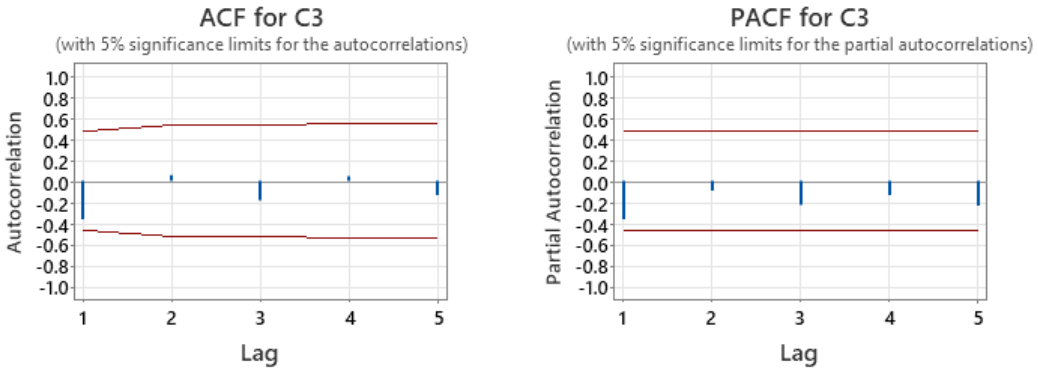


Figure 39. ACF & PACF of Demand Analgesics after differencing.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 0, q = 1*	-189.368	386.236	384.736	387.723
p = 0, q = 2	-188.822	388.312	385.645	389.628
p = 1, q = 1	-189.073	388.812	386.146	390.129
p = 0, q = 0	-192.266	389.238	388.532	390.524
p = 1, q = 0	-190.917	389.333	387.833	390.820
p = 1, q = 2	-188.955	392.195	387.910	392.888
p = 2, q = 1	-190.799	392.265	389.599	393.582
p = 2, q = 0	-190.854	392.375	389.708	393.691

** Best model with minimum AICc. Output for the best model follows.*

Figure 40. ARIMA model selection based on AICc for Demand Analgesics.

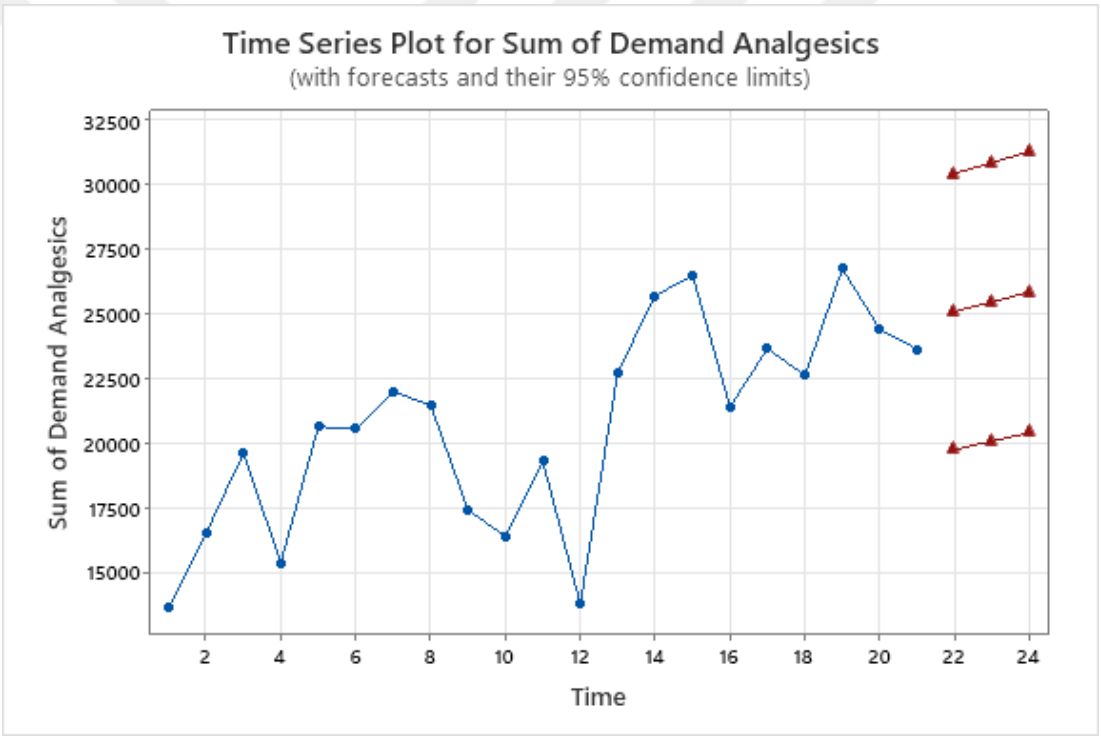


Figure 41. Forecast results from ARIMA (0,1,1) for Demand Analgesics.

In [36]:
```
#Analgesics
rows = [
    ("17-Jan", 13642, None, None, None),
    ("17-Feb", 16508, None, None, None),
    ("17-Mar", 19602, None, None, None),
    ("17-Apr", 15377, None, None, None),
    ("17-May", 20637, None, None, None),
    ("17-Jun", 20550, None, None, None),
    ("17-Jul", 21985, None, None, None),
    ("17-Aug", 21472, None, None, None),
    ("17-Sep", 17422, None, None, None),
    ("17-Oct", 16394, None, None, None),
    ("17-Nov", 19294, None, None, None),
    ("17-Dec", 13775, None, None, None),
    ("18-Jan", 22743, None, None, None),
    ("18-Feb", 25701, None, None, None),
    ("18-Mar", 26484, None, None, None),
    ("18-Apr", 21402, None, None, None),
    ("18-May", 23657, None, None, None),
    ("18-Jun", 22621, None, None, None),
    ("18-Jul", 26743, None, None, None),
    ("18-Aug", 24400, None, None, None),
    ("18-Sep", 23631, None, None, None),
    ("18-Oct", 33113, 24924.7, 24131, 25089.9),
    ("18-Nov", 26999, 24924.7, 24131, 25464),
    ("18-Dec", 20793, 24924.7, 24131, 25838.2),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Analgesics","MA(3)","SES (α=0.493246)","ARIMA(0,1,1)"])
```

```
    ("18-Dec", 20793, 24924.7, 24131, 25838.2),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Analgesics","MA(3)","SES (α=0.493246)","ARIMA(0,1,1)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)

def safe_mape(y_true, y_pred):
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Analgesics"].to_numpy()
```

```
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Analgesics"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.493246)", "ARIMA(0,1,1)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    metrics = compute_metrics(y_true, y_pred)

    rows_out.append({
        "Title": f"{col} • Univariate • Train {split_train} | Test {split_test}",
        "MAE": None if pd.isna(metrics["MAE"]) else round(metrics["MAE"], 2),
        "RMSE": None if pd.isna(metrics["RMSE"]) else round(metrics["RMSE"], 2),
        "MAPE (%)": None if pd.isna(metrics["MAPE"]) else round(metrics["MAPE"], 2),
        "R²": None if pd.isna(metrics["R2"]) else round(metrics["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                Title     MAE    RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4798.10 5428.99     17.43 -0.17
SES (α=0.493246) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 5062.67 5774.77     17.93 -0.32
    ARIMA(0,1,1) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4867.77 5543.18     18.06 -0.21
```

Table 5. Statistical model results on the testing set for Analgesics class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Analgesics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	4798.10	5428.99	17.43	-0.17

Product	Split	Model	MAE	RMSE	MAPE	R^2
Analgesics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α=0.493246)	5062.67	5774.77	17.93	-0.32
Analgesics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 1)	4867.77	5543.18	18.06	-0.2

D.2. Antibiotics Class:

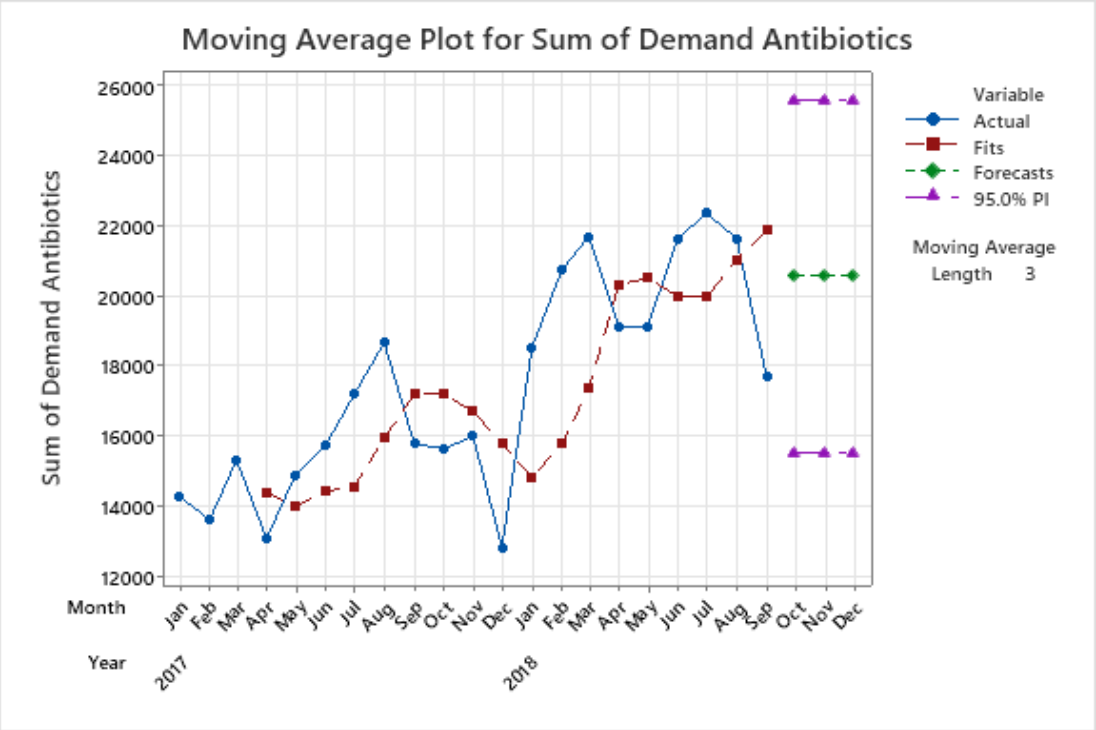


Figure 42. Moving Average Plot for Sum of Demand Antibiotics.

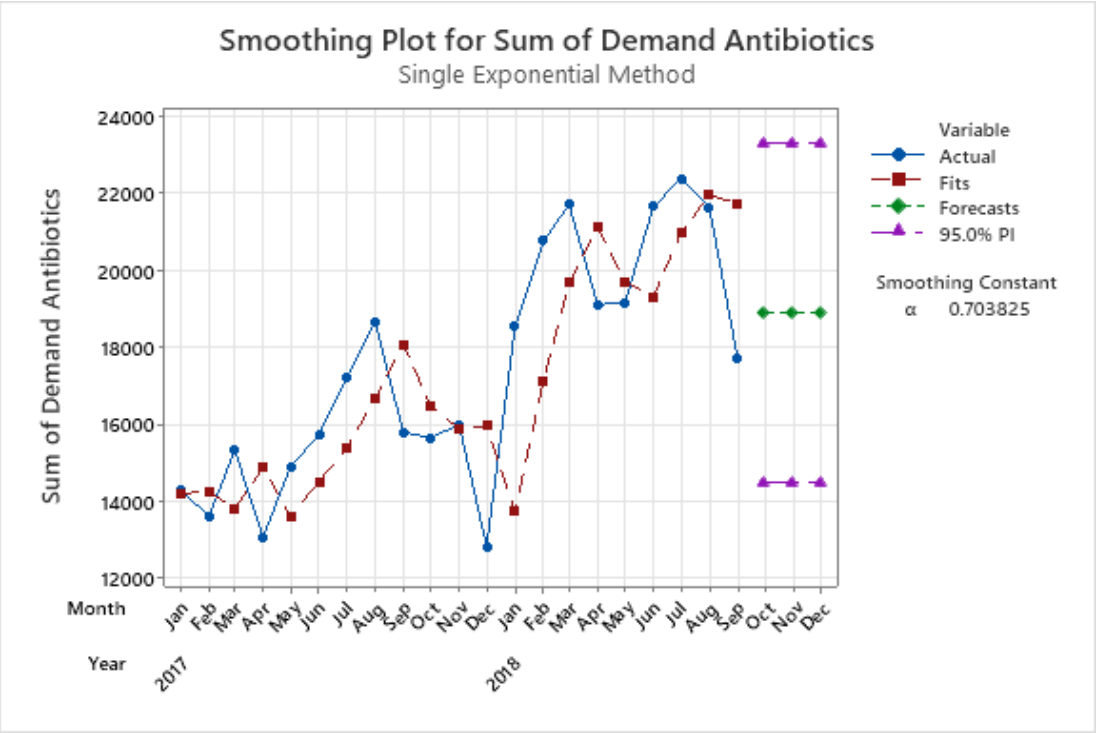


Figure 43. Smoothing Plot for Sum of Demand Antibiotics.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-1.90741	0.329	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 44. ADF Test of Demand Antibiotics before differencing.

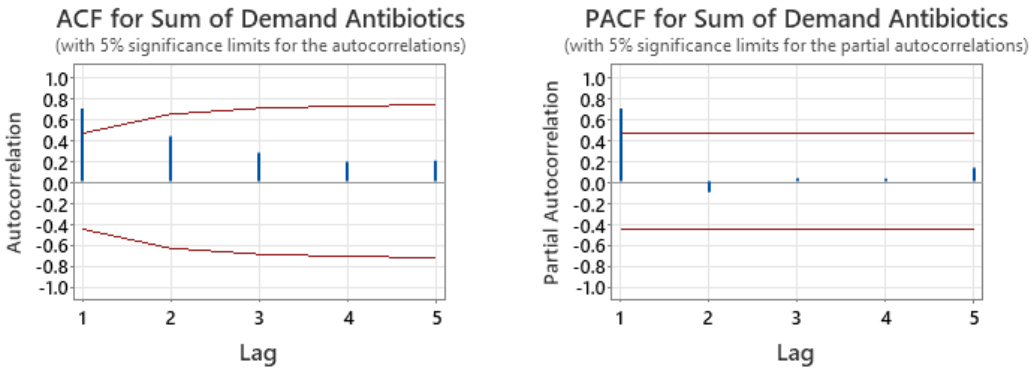


Figure 45. ACF & PACF of Demand Antibiotics before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-4.35391	0.000	Test statistic <= critical value of -3.03123. Significance level = 0.05 Reject null hypothesis. Data appears to be stationary, not supporting differencing.

Figure 46. ADF Test of Demand Antibiotics after differencing.

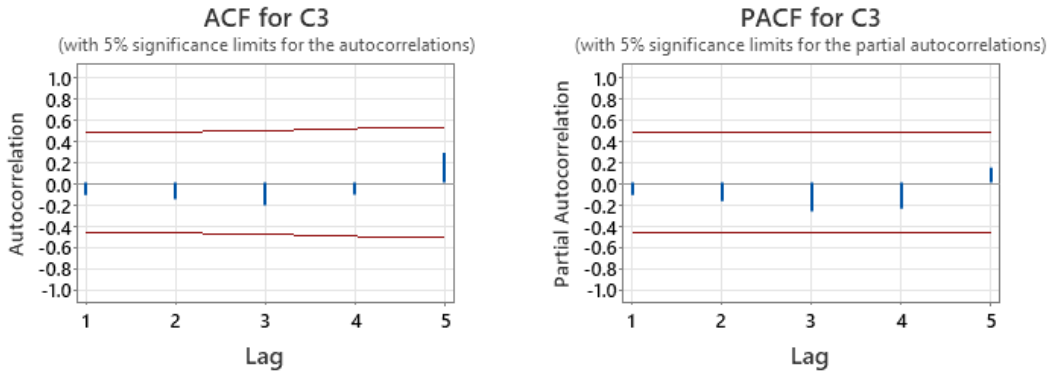


Figure 47. ACF & PACF of Demand Antibiotics after differencing.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 0, q = 0*	-182.877	370.459	369.753	371.745
p = 1, q = 0	-182.755	373.010	371.510	374.498
p = 0, q = 2	-181.653	373.972	371.305	375.288
p = 1, q = 1	-182.283	375.234	372.567	376.550
p = 2, q = 0	-182.462	375.591	372.924	376.907
p = 0, q = 1	-184.899	377.299	375.799	378.786
p = 1, q = 2	-182.094	378.474	374.188	379.167
p = 2, q = 1	-182.243	378.772	374.487	379.465
p = 2, q = 2	-181.517	381.496	375.035	381.009

** Best model with minimum AICc. Output for the best model follows.*

Figure 48. ARIMA model selection based on AICc for Demand Antibiotics.

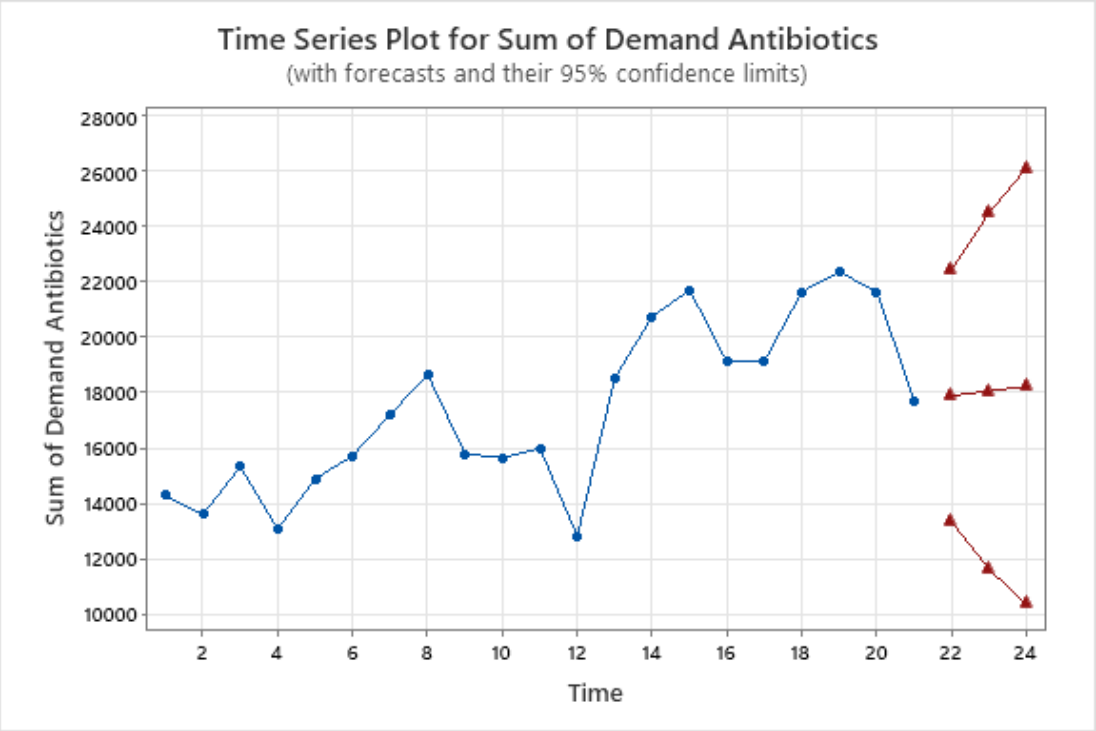


Figure 49. Forecast results from ARIMA (0,1,0) for Demand Antibiotics.

```
In [37]:
#Antibiotics
rows = [
    ("17-Jan", 14284, None, None, None),
    ("17-Feb", 13605, None, None, None),
    ("17-Mar", 15320, None, None, None),
    ("17-Apr", 13072, None, None, None),
    ("17-May", 14892, None, None, None),
    ("17-Jun", 15722, None, None, None),
    ("17-Jul", 17200, None, None, None),
    ("17-Aug", 18663, None, None, None),
    ("17-Sep", 15788, None, None, None),
    ("17-Oct", 15636, None, None, None),
    ("17-Nov", 15992, None, None, None),
    ("17-Dec", 12822, None, None, None),
    ("18-Jan", 18521, None, None, None),
    ("18-Feb", 20748, None, None, None),
    ("18-Mar", 21696, None, None, None),
    ("18-Apr", 19102, None, None, None),
    ("18-May", 19143, None, None, None),
    ("18-Jun", 21640, None, None, None),
    ("18-Jul", 22356, None, None, None),
    ("18-Aug", 21635, None, None, None),
    ("18-Sep", 17712, None, None, None),
    ("18-Oct", 27134, 20567.7, 18900.6, 17883.4),
    ("18-Nov", 22005, 20567.7, 18900.6, 18054.8),
    ("18-Dec", 16795, 20567.7, 18900.6, 18226.2),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antibiotics","MA(3)","SES (α=0.703825)","ARIMA(0,1,0)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antibiotics","MA(3)","SES (α=0.703825)","ARIMA(0,1,0)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)

def safe_mape(y_true, y_pred):
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antibiotics"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.703825)", "ARIMA(0,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    metrics = compute_metrics(y_true, y_pred)
```

```
split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antibiotics"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.703825)", "ARIMA(0,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    metrics = compute_metrics(y_true, y_pred)

    rows_out.append({
        "Title": f"{col} • Univariate • Train {split_train} | Test {split_test}",
        "MAE": None if pd.isna(metrics["MAE"]) else round(metrics["MAE"], 2),
        "RMSE": None if pd.isna(metrics["RMSE"]) else round(metrics["RMSE"], 2),
        "MAPE (%)": None if pd.isna(metrics["MAPE"]) else round(metrics["MAPE"], 2),
        "R²": None if pd.isna(metrics["R2"]) else round(metrics["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                  Title     MAE    RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 3925.43 4450.30     17.73 -0.11
SES (α=0.703825) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4481.13 5223.66     19.00 -0.53
    ARIMA(0,1,0) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4877.33 5865.89     20.19 -0.93
```

Table 6. Statistical model results on the testing set for Antibiotics class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Antibiotics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	3925.43	4450.30	17.73	-0.11
Antibiotics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α=0.703825)	4481.13	5223.66	19.00	-0.53
Antibiotics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 0)	4877.33	5865.89	20.19	-0.93

D.3. Antimalarial Class:

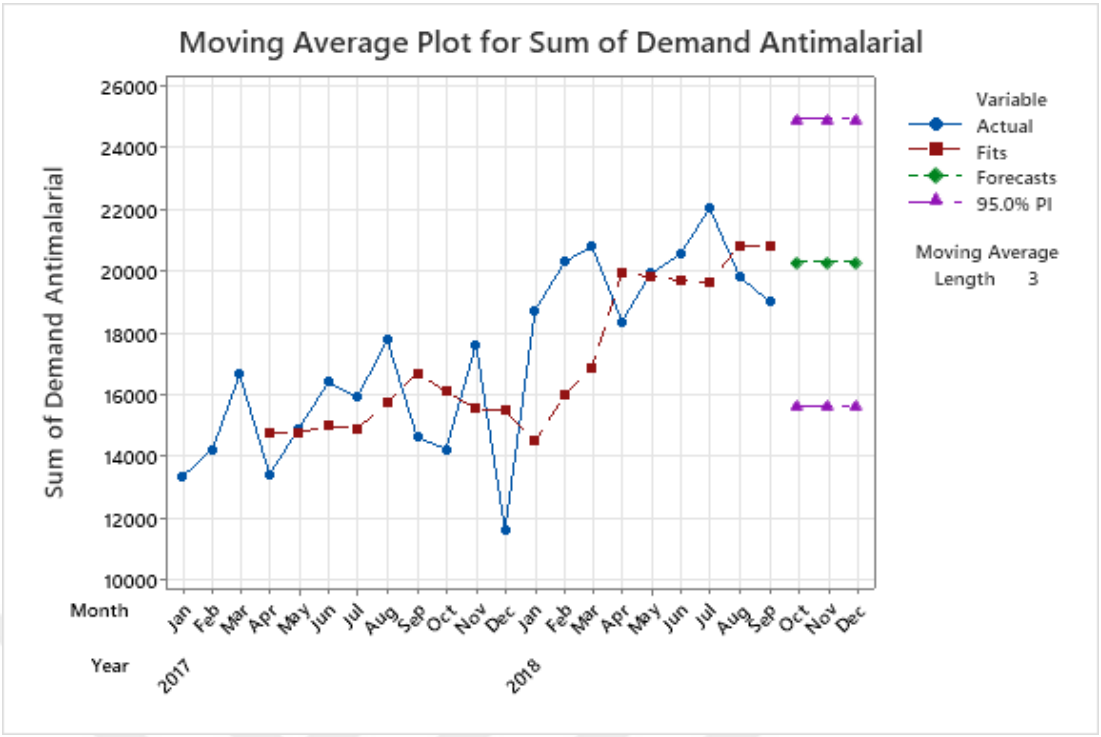


Figure 50. Moving Average Plot for Sum of Demand Antimalarial.

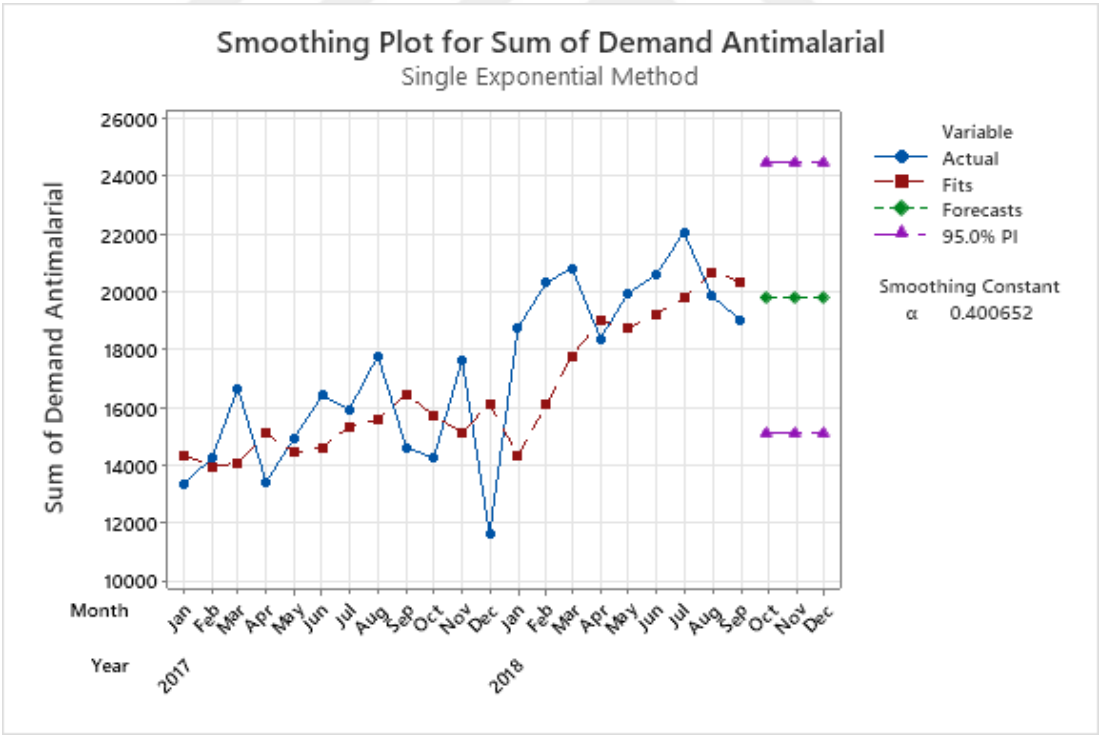


Figure 51. Smoothing Plot for Sum of Demand Antimalarial.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-2.51866	0.111	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 52. ADF Test of Demand Antimalarial before differencing.

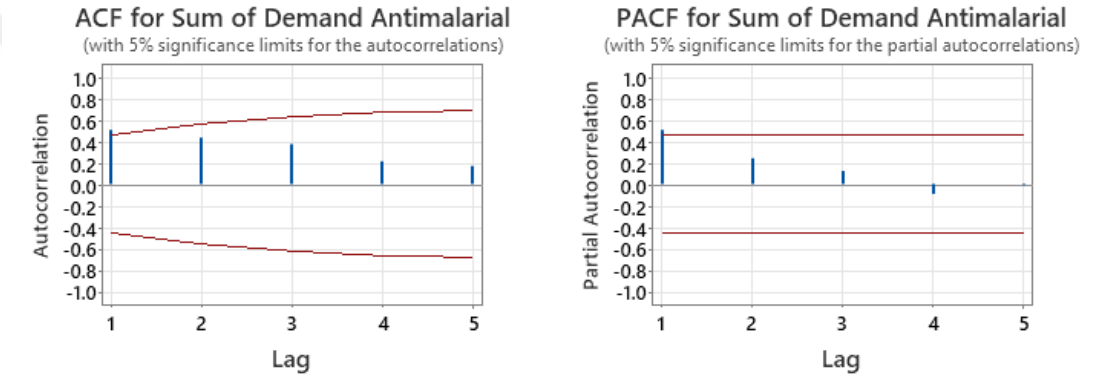


Figure 53. ACF & PACF of Demand Antimalarial before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-6.99156	0.000	Test statistic <= critical value of -3.03123. Significance level = 0.05 Reject null hypothesis. Data appears to be stationary, not supporting differencing.

Figure 54. ADF Test of Demand Antimalarial after differencing.

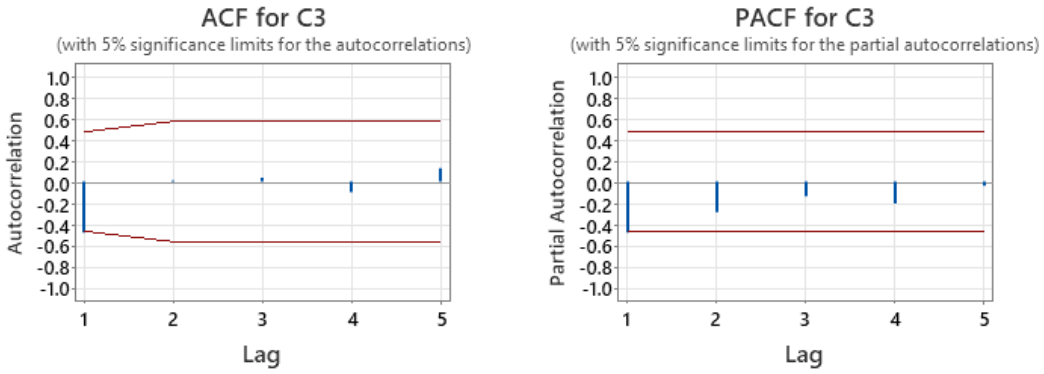


Figure 55. ACF & PACF of Demand Antimalarial after differencing.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 0, q = 1*	-181.806	371.112	369.612	372.599
p = 0, q = 2	-181.850	374.367	371.700	375.683
p = 1, q = 1	-181.851	374.368	371.701	375.684
p = 1, q = 0	-184.238	375.975	374.475	377.462
p = 2, q = 0	-183.249	377.164	374.498	378.480
p = 1, q = 2	-181.936	378.159	373.873	378.852
p = 0, q = 0	-186.750	378.206	377.500	379.492
p = 2, q = 1	-182.115	378.516	374.231	379.209
p = 2, q = 2	-183.803	381.892	377.606	382.585

** Best model with minimum AICc. Output for the best model follows.*

Figure 56. ARIMA model selection based on AICc for Demand Antimalarial.

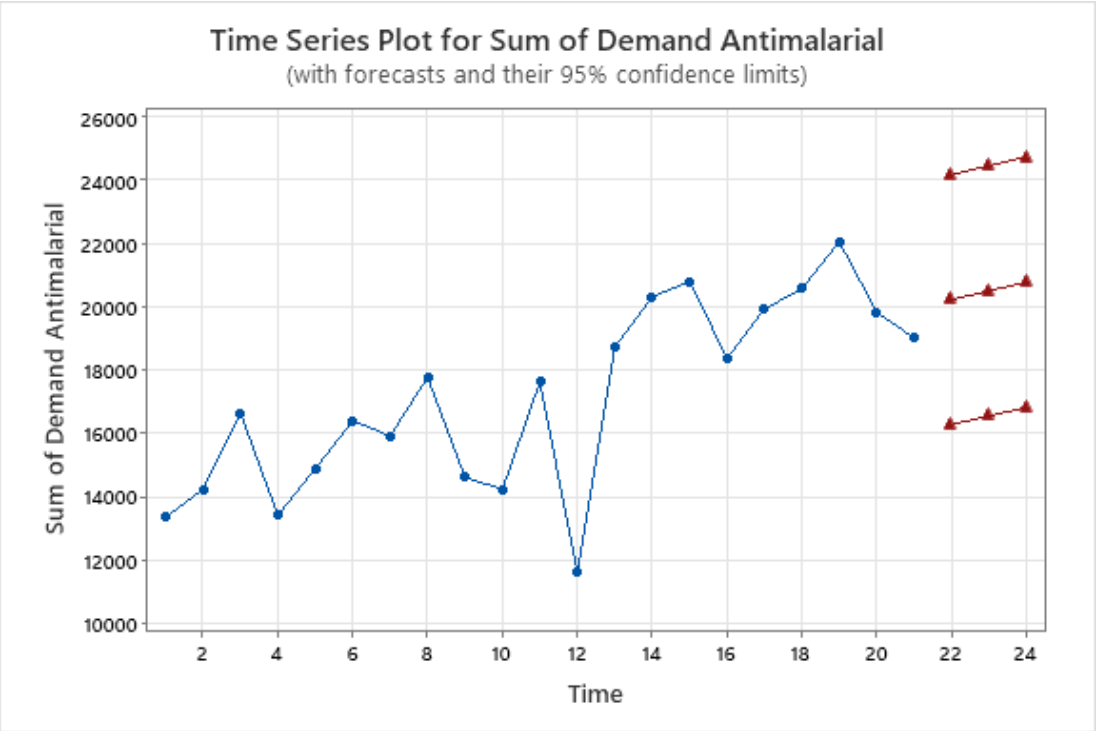


Figure 57. Forecast results from ARIMA (0,1,1) for Demand Antimalarial.

In [38]:

```
#Antimalarial
rows = [
    ("17-Jan", 13354, None, None, None),
    ("17-Feb", 14232, None, None, None),
    ("17-Mar", 16644, None, None, None),
    ("17-Apr", 13418, None, None, None),
    ("17-May", 14880, None, None, None),
    ("17-Jun", 16399, None, None, None),
    ("17-Jul", 15916, None, None, None),
    ("17-Aug", 17765, None, None, None),
    ("17-Sep", 14608, None, None, None),
    ("17-Oct", 14235, None, None, None),
    ("17-Nov", 17618, None, None, None),
    ("17-Dec", 11613, None, None, None),
    ("18-Jan", 18719, None, None, None),
    ("18-Feb", 20298, None, None, None),
    ("18-Mar", 20787, None, None, None),
    ("18-Apr", 18360, None, None, None),
    ("18-May", 19920, None, None, None),
    ("18-Jun", 20560, None, None, None),
    ("18-Jul", 22032, None, None, None),
    ("18-Aug", 19814, None, None, None),
    ("18-Sep", 19001, None, None, None),
    ("18-Oct", 25741, 20282.3, 19793.5, 20214.1),
    ("18-Nov", 22382, 20282.3, 19793.5, 20490.0),
    ("18-Dec", 17748, 20282.3, 19793.5, 20765.8),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antimalarial","MA(3)","SES (α=0.400652)","ARIMA(0,1,1)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antimalarial","MA(3)","SES (α=0.400652)","ARIMA(0,1,1)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)

def safe_mape(y_true, y_pred):
    """MAPE that safely ignores zero targets to avoid division by zero."""
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R² for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"
```

```python
def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R² for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antimalarial"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.400652)", "ARIMA(0,1,1)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    metrics = compute_metrics(y_true, y_pred)
    rows_out.append({
        "Title": f"{col} • Univariate • Train {split_train} | Test {split_test}",
        "MAE": None if pd.isna(metrics["MAE"]) else round(metrics["MAE"], 2),
        "RMSE": None if pd.isna(metrics["RMSE"]) else round(metrics["RMSE"], 2),
        "MAPE (%)": None if pd.isna(metrics["MAPE"]) else round(metrics["MAPE"], 2),
        "R²": None if pd.isna(metrics["R2"]) else round(metrics["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                                Title     MAE    RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 3364.23 3680.07     14.96 -0.26
SES (α=0.400652) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 3527.17 3926.71     15.40 -0.44
    ARIMA(0,1,1) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 3478.90 3796.20     15.64 -0.34
```

Table 7. Statistical model results on the testing set for Antimalarial class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Antimalarial	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	3364.23	3680.07	14.96	-0.26
Antimalarial	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α= 0.400652)	3527.17	3926.71	15.40	-0.44
Antimalarial	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 1)	3478.90	3796.20	15.64	-0.34

D.4. Antipiretics Class:

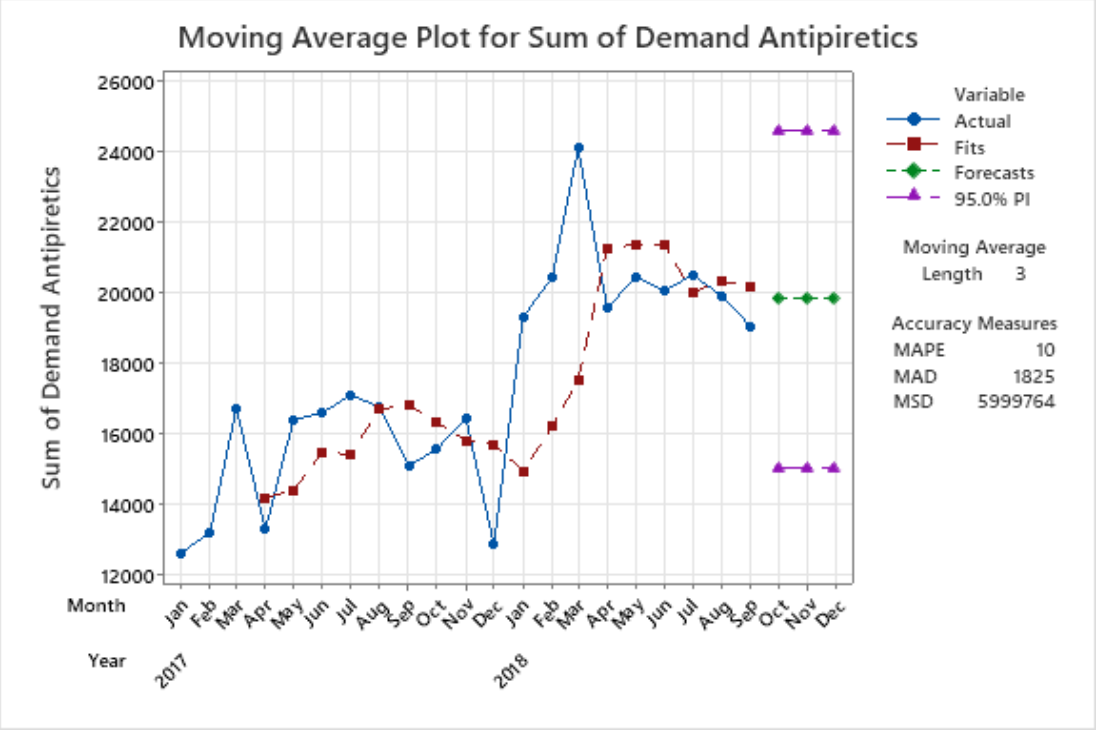


Figure 58. Moving Average Plot for Sum of Demand Antipiretics.

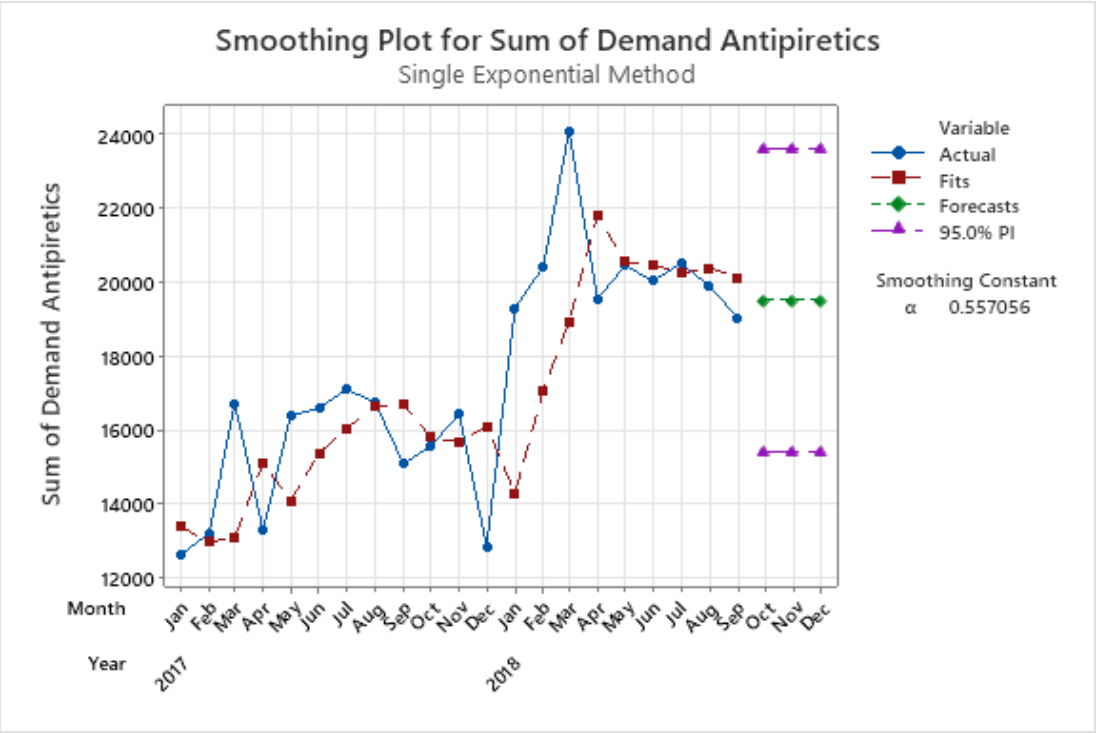


Figure 59. Smoothing Plot for Sum of Demand Antipiretics.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-2.36497	0.152	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 60. ADF Test of Demand Antipiretics before differencing.

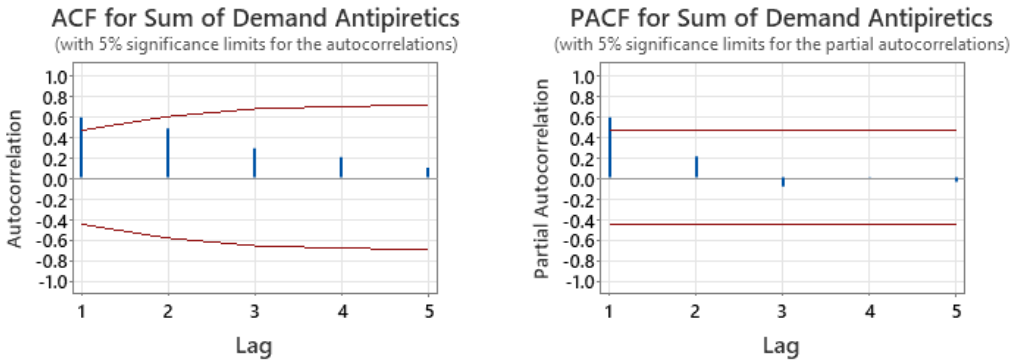


Figure 61. ACF & PACF of Demand Antipiretics before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-6.72773	0.000	Test statistic <= critical value of -3.03123. Significance level = 0.05 Reject null hypothesis. Data appears to be stationary, not supporting differencing.

Figure 62. ADF Test of Demand Antipiretics after differencing.

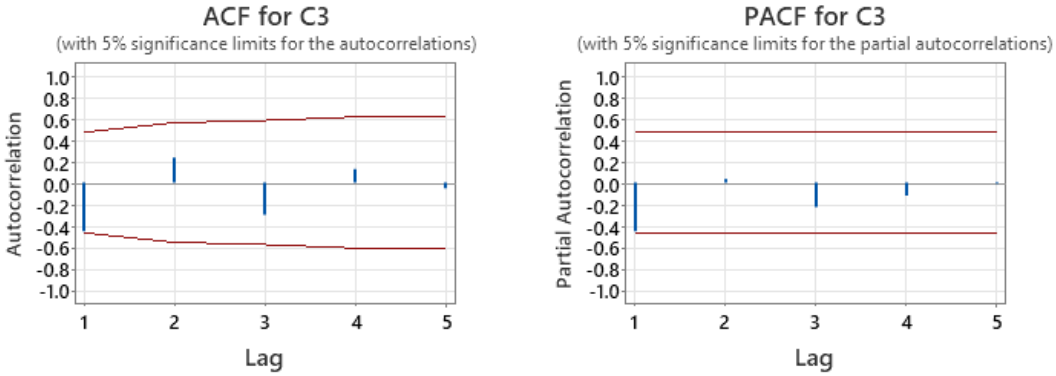


Figure 63. ACF & PACF of Demand Antipiretics after differencing.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 0, q = 1*	-182.357	372.213	370.713	373.700
p = 1, q = 0	-182.974	373.447	371.947	374.934
p = 0, q = 0	-185.157	375.019	374.313	376.305
p = 1, q = 1	-182.188	375.042	372.376	376.358
p = 0, q = 2	-182.417	375.502	372.835	376.818
p = 2, q = 0	-183.040	376.747	374.080	378.063
p = 1, q = 2	-182.488	379.261	374.976	379.954

** Best model with minimum AICc. Output for the best model follows.*

Figure 64. ARIMA model selection based on AICc for Demand Antipiretics.

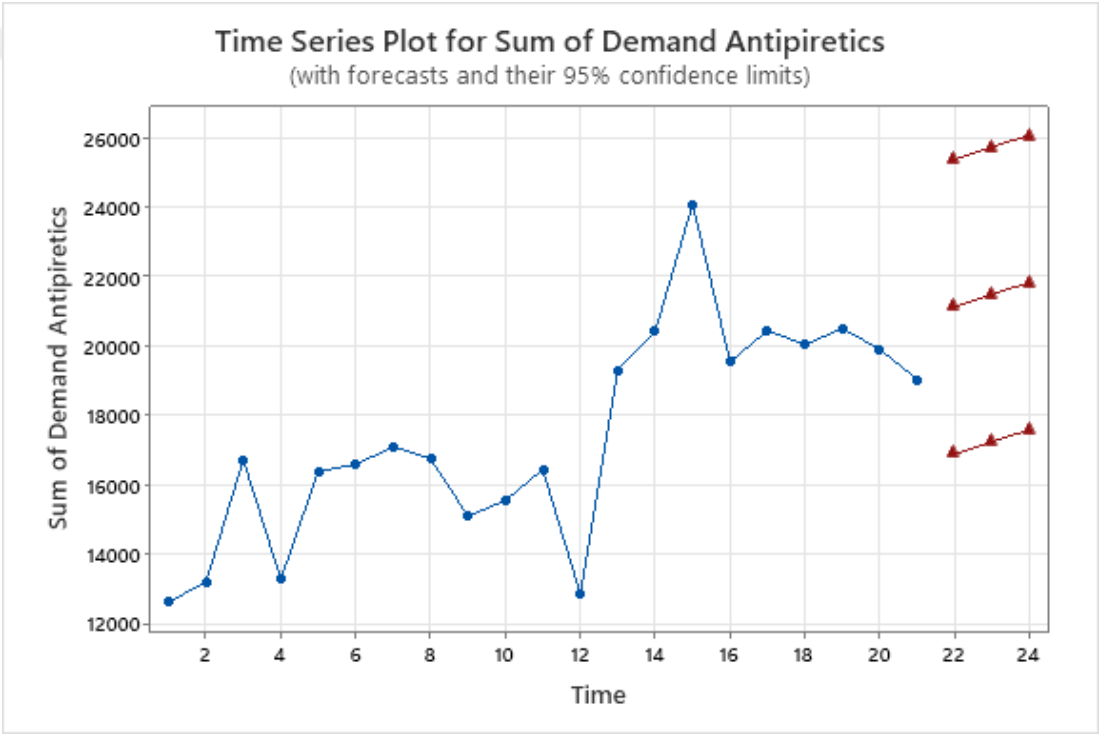


Figure 65. Forecast results from ARIMA (0,1,1) for Demand Antipiretics.

In [39]:

```
#Antipiretics
rows = [
    ("17-Jan", 12620, None, None, None),
    ("17-Feb", 13196, None, None, None),
    ("17-Mar", 16720, None, None, None),
    ("17-Apr", 13277, None, None, None),
    ("17-May", 16388, None, None, None),
    ("17-Jun", 16587, None, None, None),
    ("17-Jul", 17091, None, None, None),
    ("17-Aug", 16758, None, None, None),
    ("17-Sep", 15094, None, None, None),
    ("17-Oct", 15551, None, None, None),
    ("17-Nov", 16439, None, None, None),
    ("17-Dec", 12847, None, None, None),
    ("18-Jan", 19299, None, None, None),
    ("18-Feb", 20420, None, None, None),
    ("18-Mar", 24086, None, None, None),
    ("18-Apr", 19541, None, None, None),
    ("18-May", 20445, None, None, None),
    ("18-Jun", 20040, None, None, None),
    ("18-Jul", 20506, None, None, None),
    ("18-Aug", 19909, None, None, None),
    ("18-Sep", 19034, None, None, None),
    ("18-Oct", 25546, 19816.3, 19515.5, 21132.6),
    ("18-Nov", 22788, 19816.3, 19515.5, 21483.9),
    ("18-Dec", 14629, 19816.3, 19515.5, 21835.1),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antipiretics","MA(3)","SES (α=0.557056)","ARIMA(0,1,1)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
    ("18-Dec", 14629, 19816.3, 19515.5, 21835.1),
]

df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antipiretics","MA(3)","SES (α=0.557056)","ARIMA(0,1,1)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)

def safe_mape(y_true, y_pred):
    """MAPE that safely ignores zero targets to avoid division by zero."""
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R² for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antipiretics"].to_numpy()
```

```
split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antipiretics"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.557056)", "ARIMA(0,1,1)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    m = compute_metrics(y_true, y_pred)
    rows_out.append({
        "Title": f"{col} • Univariate • Train {split_train} | Test {split_test}",
        "MAE": None if pd.isna(m["MAE"]) else round(m["MAE"], 2),
        "RMSE": None if pd.isna(m["RMSE"]) else round(m["RMSE"], 2),
        "MAPE (%)": None if pd.isna(m["MAPE"]) else round(m["MAPE"], 2),
        "R²": None if pd.isna(m["R2"]) else round(m["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                Title     MAE    RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4629.57 4780.81     23.64 -0.06
SES (α=0.557056) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4729.83 4863.27     23.79 -0.10
    ARIMA(0,1,1) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 4307.87 4936.49     24.09 -0.13
```

Table 8. Statistical model results on the testing set for Antipiretics class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Antipiretics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	4629.57	4780.81	23.64	-0.06
Antipiretics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α= 0.557056)	4729.83	4863.27	23.79	-0.10
Antipiretics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 1)	4307.87	4936.49	24.09	-0.13

D.5. Antiseptics Class:

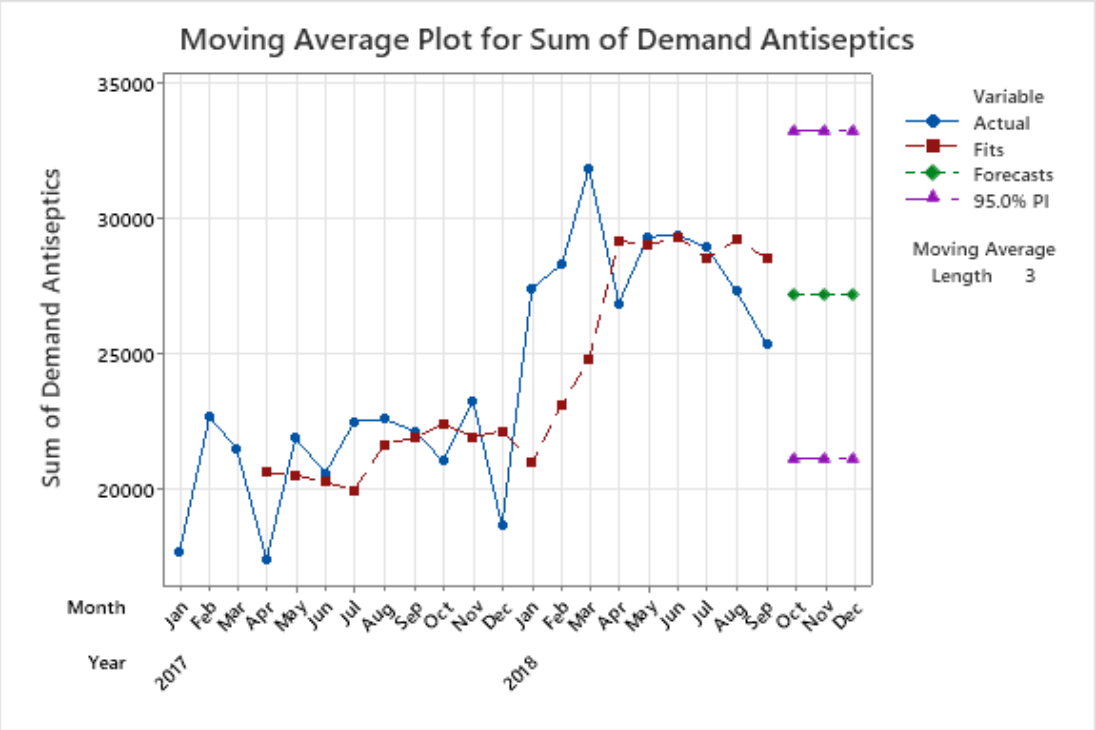


Figure 66. Moving Average Plot for Sum of Demand Antiseptics.

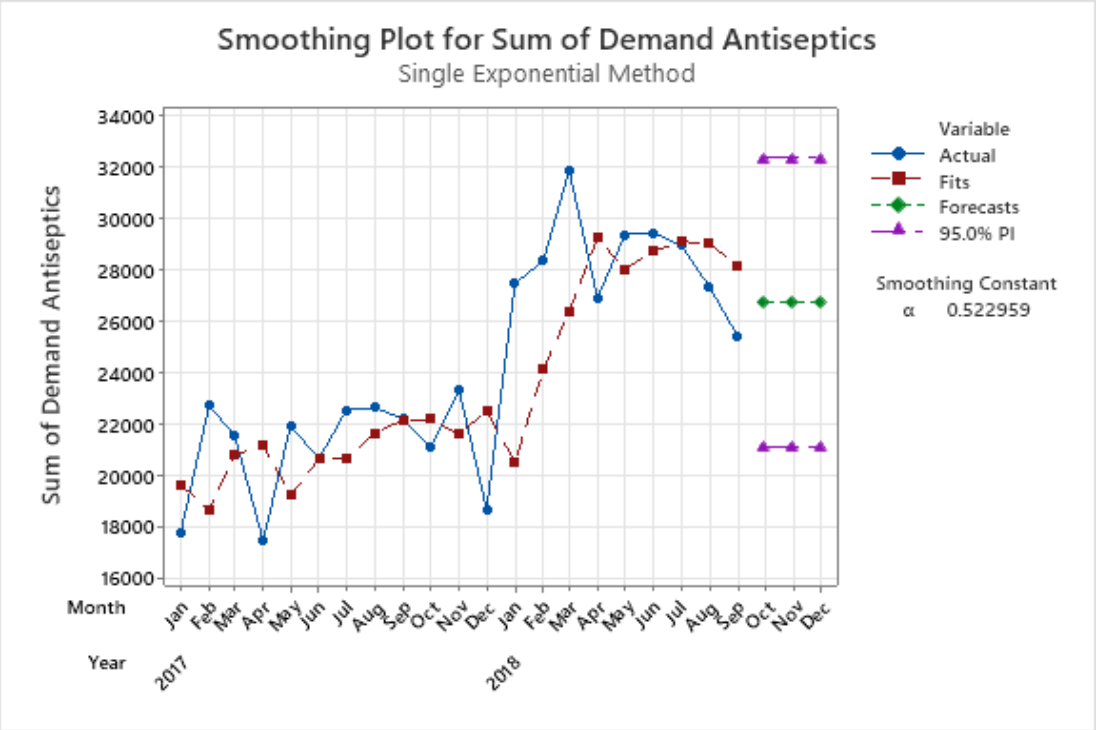


Figure 67. Smoothing Plot for Sum of Demand Antiseptics.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-1.22676	0.662	Test statistic > critical value of -3.03123. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 68. ADF Test of Demand Antiseptics before differencing.

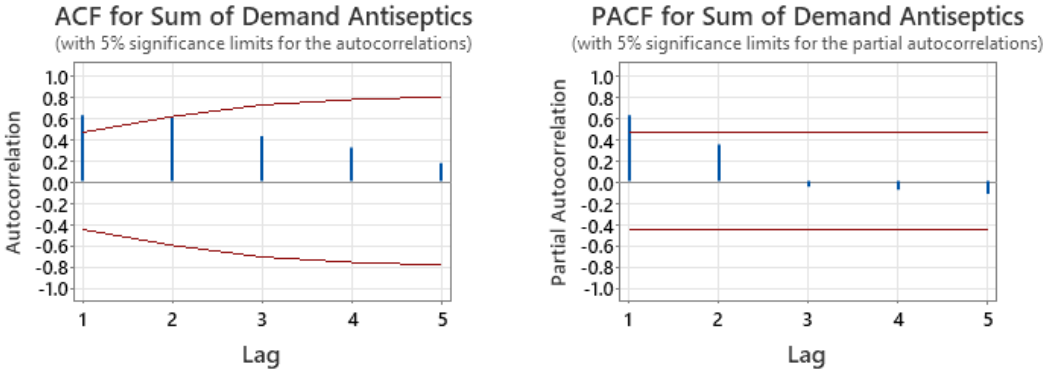


Figure 69. ACF & PACF of Demand Antiseptics before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-7.03224	0.000	Test statistic <= critical value of -3.03123. Significance level = 0.05 Reject null hypothesis. Data appears to be stationary, not supporting differencing.

Figure 70. ADF Test of Demand Antiseptics after differencing.

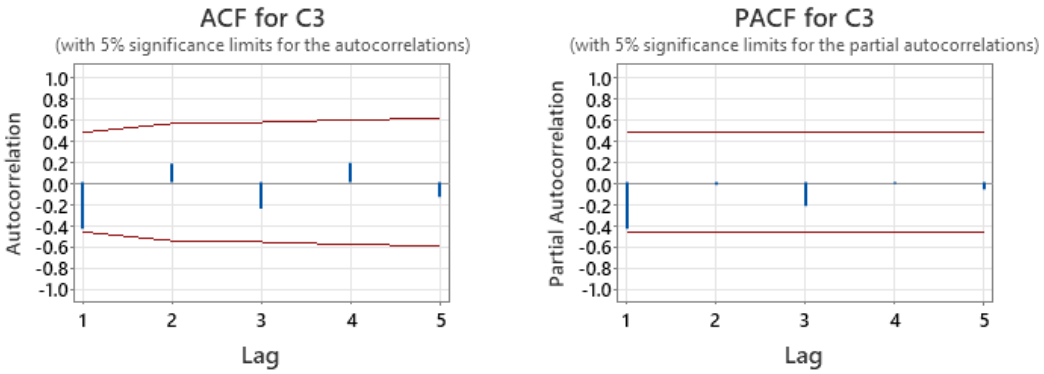


Figure 71. ACF & PACF of Demand Antiseptics after differencing.

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 1, q = 0*	-188.190	383.880	382.380	385.367
p = 0, q = 1	-188.297	384.093	382.593	385.580
p = 0, q = 0	-190.512	385.729	385.023	387.015
p = 1, q = 1	-187.665	385.997	383.330	387.313
p = 0, q = 2	-187.746	386.159	383.492	387.475
p = 2, q = 0	-188.218	387.103	384.436	388.419
p = 1, q = 2	-187.948	390.181	385.895	390.874
p = 2, q = 1	-188.324	390.935	386.649	391.628
p = 2, q = 2	-187.345	393.152	386.691	392.665

** Best model with minimum AICc. Output for the best model follows.*

Figure 72. ARIMA model selection based on AICc for Demand Antiseptics.

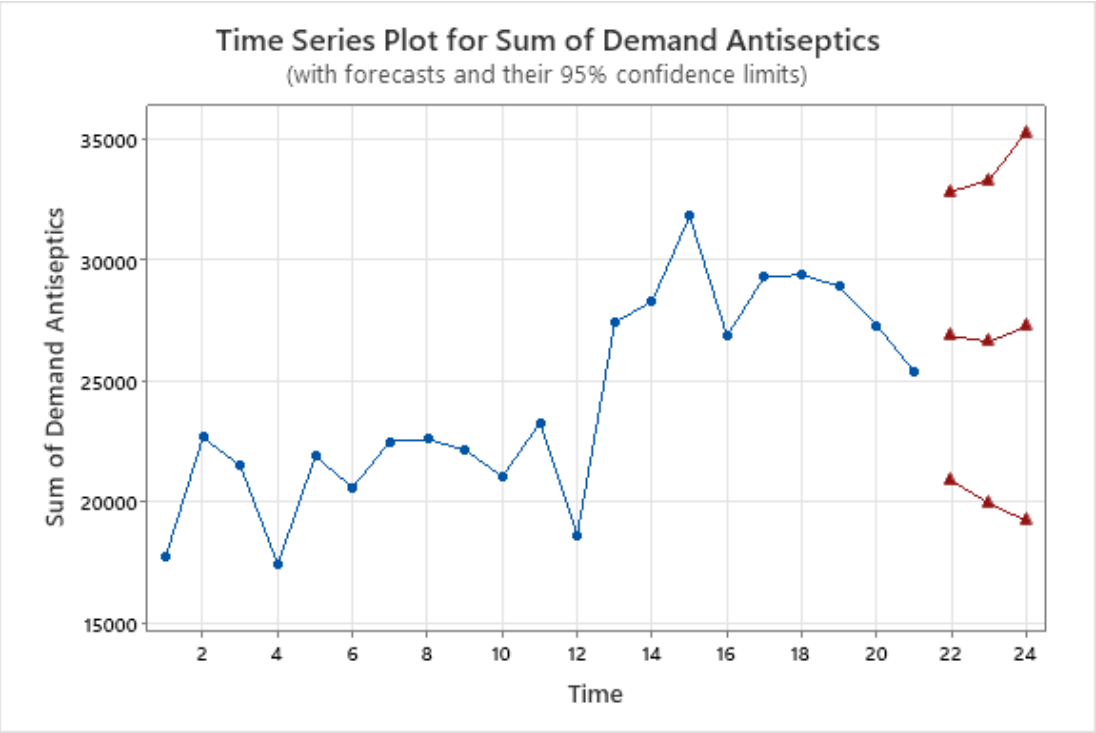


Figure 73. Forecast results from ARIMA (1,1,0) for Demand Antiseptics.

In [43]:
```
#Antiseptics
rows = [
    ("17-Jan", 17729, None, None, None),
    ("17-Feb", 22679, None, None, None),
    ("17-Mar", 21511, None, None, None),
    ("17-Apr", 17429, None, None, None),
    ("17-May", 21900, None, None, None),
    ("17-Jun", 20612, None, None, None),
    ("17-Jul", 22516, None, None, None),
    ("17-Aug", 22607, None, None, None),
    ("17-Sep", 22167, None, None, None),
    ("17-Oct", 21054, None, None, None),
    ("17-Nov", 23281, None, None, None),
    ("17-Dec", 18658, None, None, None),
    ("18-Jan", 27431, None, None, None),
    ("18-Feb", 28312, None, None, None),
    ("18-Mar", 31824, None, None, None),
    ("18-Apr", 26879, None, None, None),
    ("18-May", 29321, None, None, None),
    ("18-Jun", 29395, None, None, None),
    ("18-Jul", 28923, None, None, None),
    ("18-Aug", 27312, None, None, None),
    ("18-Sep", 25382, None, None, None),
    ("18-Oct", 37994, 27205.7, 26683.7, 26863.7),
    ("18-Nov", 30216, 27205.7, 26683.7, 26638.0),
    ("18-Dec", 23327, 27205.7, 26683.7, 27266.8),
]
df = pd.DataFrame(rows, columns=["Date","Sum of Demand Antiseptics","MA(3)","SES (α=0.522959)","ARIMA(1,1,0)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
def safe_mape(y_true, y_pred):
    """MAPE that safely ignores zero targets to avoid division by zero."""
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R² for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antiseptics"].to_numpy()
```

```
split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Antiseptics"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.522959)", "ARIMA(1,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    m = compute_metrics(y_true, y_pred)
    rows_out.append({
        "Title": f"{col} • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12",
        "MAE": None if pd.isna(m["MAE"]) else round(m["MAE"], 2),
        "RMSE": None if pd.isna(m["RMSE"]) else round(m["RMSE"], 2),
        "MAPE (%)": None if pd.isna(m["MAPE"]) else round(m["MAPE"], 2),
        "R²": None if pd.isna(m["R2"]) else round(m["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                                  Title     MAE    RMSE  MAPE (%)    R²
                                MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 5892.43 6843.33     18.33 -0.30
SES (α=0.522959) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 6066.43 7110.26     18.62 -0.41
            ARIMA(1,1,0) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 6216.03 7122.91     19.34 -0.41
```

Table 9. Statistical model results on the testing set for Antiseptics class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Antiseptics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	5892.43	6843.33	18.33	-0.30
Antiseptics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α= 0.522959)	6066.43	7110.26	18.62	-0.41
Antiseptics	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 1)	6216.03	7122.91	19.34	-0.41

D.6. Mood Stabilizers Class:

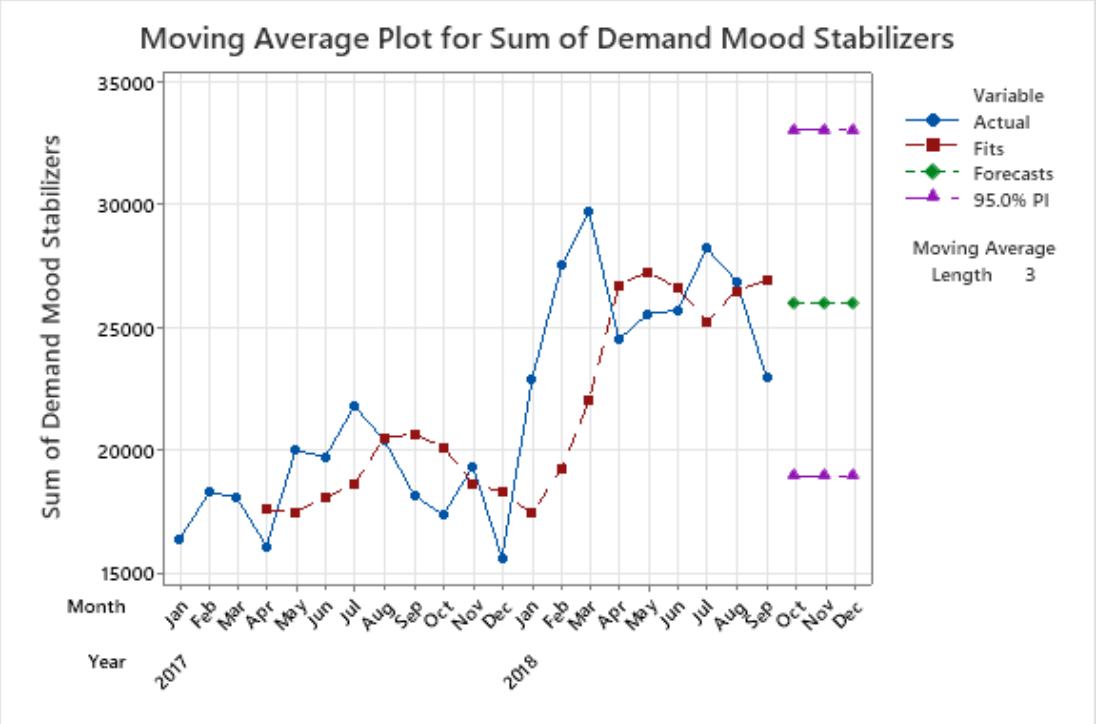


Figure 74. Moving Average Plot for Sum of Demand Mood Stabilizers.

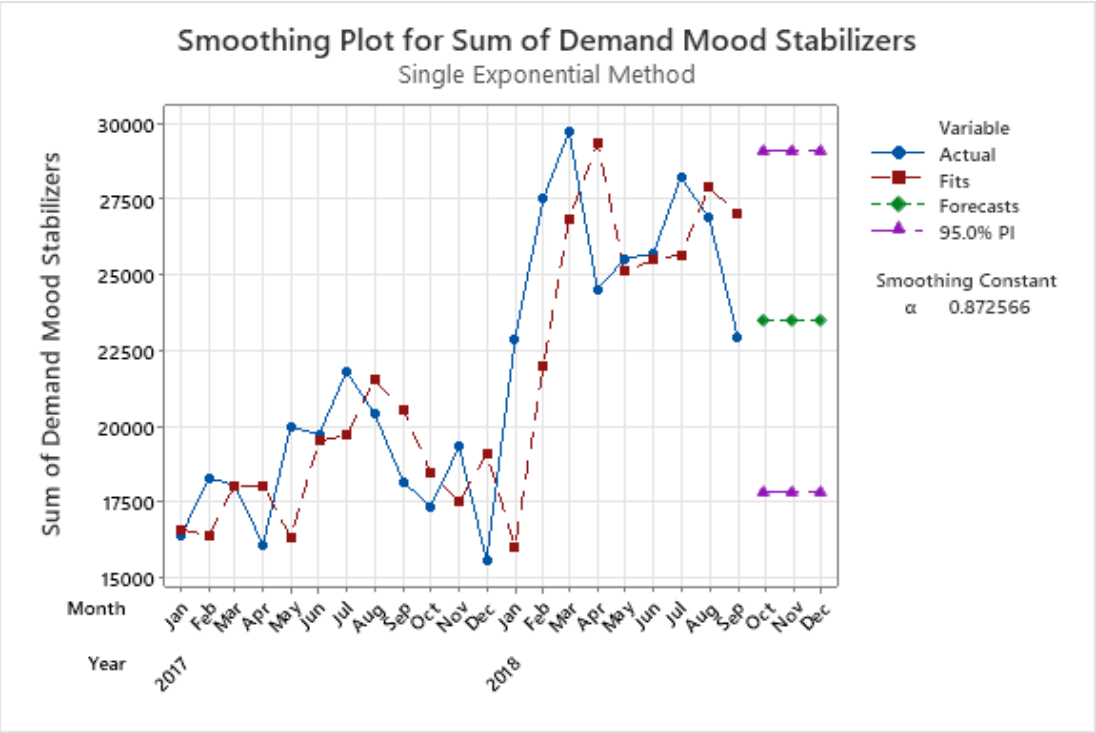


Figure 75. Smoothing Plot for Sum of Demand Mood Stabilizers.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-1.86073	0.351	Test statistic > critical value of -3.02165. Significance level = 0.05 Fail to reject null hypothesis. Consider differencing to make data stationary.

Figure 76. ADF Test of Demand Mood Stabilizers before differencing.

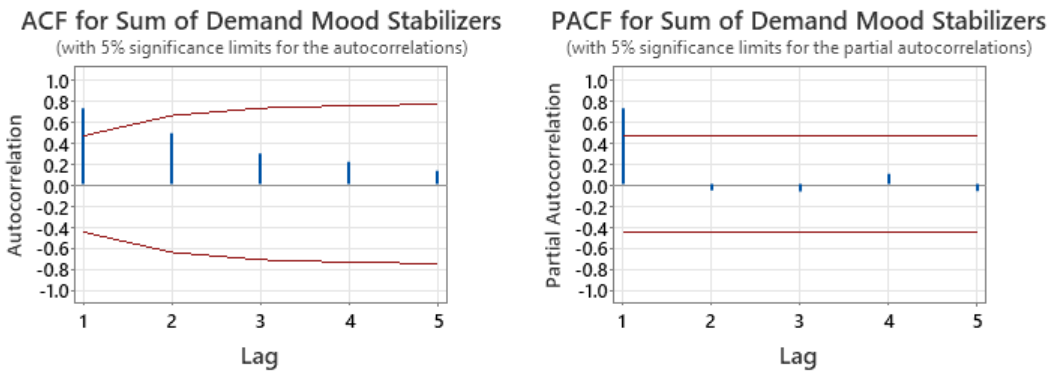


Figure 77. ACF & PACF of Demand Mood Stabilizers before differencing.

Augmented Dickey-Fuller Test

Null hypothesis: Data are non-stationary
Alternative hypothesis: Data are stationary

Test Statistic	P-Value	Recommendation
-4.34974	0.000	Test statistic <= critical value of -3.03123.
		Significance level = 0.05
		Reject null hypothesis.
		Data appears to be stationary, not supporting differencing.

Figure 78. ADF Test of Demand Mood Stabilizers after differencing.

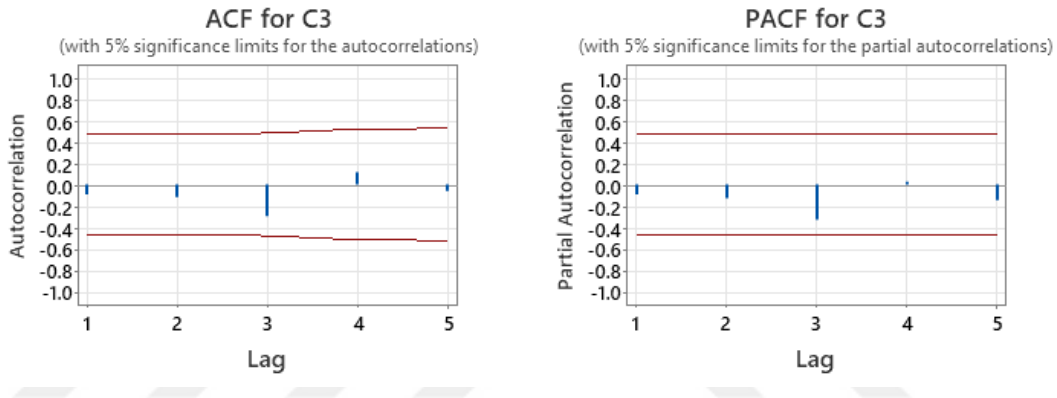


Figure 79. ACF & PACF of Demand Mood Stabilizers after differencing

Model Selection

Model (d = 1)	LogLikelihood	AICc	AIC	BIC
p = 0, q = 0*	-188.654	382.013	381.308	383.299
p = 0, q = 1	-188.551	384.601	383.101	386.088
p = 1, q = 0	-188.603	384.706	383.206	386.193
p = 1, q = 1	-187.413	385.493	382.827	386.810
p = 0, q = 2	-187.723	386.113	383.447	387.430
p = 2, q = 0	-188.494	387.655	384.989	388.972
p = 1, q = 2	-187.309	388.903	384.617	389.596
p = 2, q = 2	-187.581	393.624	387.162	393.137

** Best model with minimum AICc. Output for the best model follows.*

Figure 80. ARIMA model selection based on AICc for Demand Mood Stabilizers.

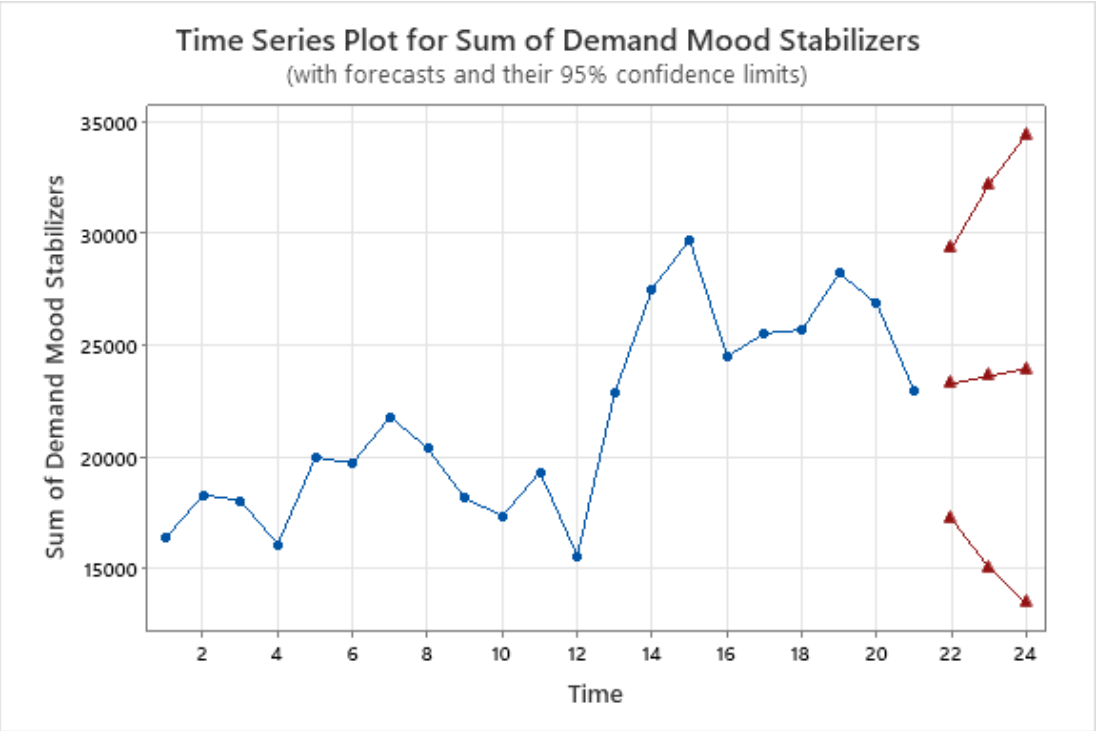


Figure 81. Forecast results from ARIMA (0,1,0) for Demand Mood Stabilizers.

In [44]:
```
#Mood Stabilizers
rows = [
    ("17-Jan", 16357, None, None, None),
    ("17-Feb", 18297, None, None, None),
    ("17-Mar", 18052, None, None, None),
    ("17-Apr", 16080, None, None, None),
    ("17-May", 19983, None, None, None),
    ("17-Jun", 19724, None, None, None),
    ("17-Jul", 21794, None, None, None),
    ("17-Aug", 20397, None, None, None),
    ("17-Sep", 18161, None, None, None),
    ("17-Oct", 17349, None, None, None),
    ("17-Nov", 19327, None, None, None),
    ("17-Dec", 15579, None, None, None),
    ("18-Jan", 22882, None, None, None),
    ("18-Feb", 27531, None, None, None),
    ("18-Mar", 29735, None, None, None),
    ("18-Apr", 24508, None, None, None),
    ("18-May", 25543, None, None, None),
    ("18-Jun", 25690, None, None, None),
    ("18-Jul", 28220, None, None, None),
    ("18-Aug", 26871, None, None, None),
    ("18-Sep", 22958, None, None, None),
    ("18-Oct", 35566, 26016.3, 23473.3, 23288.0),
    ("18-Nov", 25402, 26016.3, 23473.3, 23618.1),
    ("18-Dec", 20036, 26016.3, 23473.3, 23948.2),
]
df = pd.DataFrame(rows, columns=["Date","Sum of Demand Mood Stabilizers","MA(3)","SES (α=0.872566)","ARIMA(0,1,0)"])
df["Date"] = pd.to_datetime(df["Date"], format="%y-%b")
df = df.sort_values("Date").reset_index(drop=True)
```

```
def safe_mape(y_true, y_pred):
    """MAPE that safely ignores zero targets to avoid division by zero."""
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def compute_metrics(y_true, y_pred):
    """Compute MAE, RMSE, MAPE, and R² for equal-length arrays."""
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE  = mean_absolute_error(y_true, y_pred),
        RMSE = np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE = safe_mape(y_true, y_pred),
        R2   = r2_score(y_true, y_pred),
    )

split_train = "2017-01 → 2018-09"
split_test  = "2018-10 → 2018-12"

test_mask = (df["Date"] >= "2018-10-01") & (df["Date"] <= "2018-12-31")
y_true = df.loc[test_mask, "Sum of Demand Mood Stabilizers"].to_numpy()

rows_out = []
for col in ["MA(3)", "SES (α=0.872566)", "ARIMA(0,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    m = compute_metrics(y_true, y_pred)
    rows_out.append({
        "Title": f"{col} • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12",
        "MAE": None if pd.isna(m["MAE"]) else round(m["MAE"], 2),
        "RMSE": None if pd.isna(m["RMSE"]) else round(m["RMSE"], 2),
        "MAPE (%)": None if pd.isna(m["MAPE"]) else round(m["MAPE"], 2),
        "R²": None if pd.isna(m["R2"]) else round(m["R2"], 2),
    })
```

```
rows_out = []
for col in ["MA(3)", "SES (α=0.872566)", "ARIMA(0,1,0)"]:
    y_pred = df.loc[test_mask, col].to_numpy()
    m = compute_metrics(y_true, y_pred)
    rows_out.append({
        "Title": f"{col} • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12",
        "MAE": None if pd.isna(m["MAE"]) else round(m["MAE"], 2),
        "RMSE": None if pd.isna(m["RMSE"]) else round(m["RMSE"], 2),
        "MAPE (%)": None if pd.isna(m["MAPE"]) else round(m["MAPE"], 2),
        "R²": None if pd.isna(m["R2"]) else round(m["R2"], 2),
    })

df_out = pd.DataFrame(rows_out)
print(df_out.to_string(index=False))
```

```
                                                                  Title     MAE    RMSE  MAPE (%)    R²
          MA(3) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 5381.43 6515.06     19.71 -0.02
SES (α=0.872566) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 5819.57 7343.21     19.58 -0.30
    ARIMA(0,1,0) • Univariate • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 5991.37 7510.81     20.36 -0.36
```

Table 10. Statistical model results on the testing set for Mood Stabilizers class.

Product	Split	Model	MAE	RMSE	MAPE	R^2
Mood Stabilizers	**2017-01 → 2018-09 / 2018-10 → 2018-12**	MA(3)	5381.43	6515.06	19.71	-0.02
Mood Stabilizers	**2017-01 → 2018-09 / 2018-10 → 2018-12**	SES (α= 0.872566)	5819.57	7343.21	19.58	-0.30
Mood Stabilizers	**2017-01 → 2018-09 / 2018-10 → 2018-12**	ARIMA(0, 1, 0)	5991.37	7510.81	20.36	-0.36

APPENDIX E: All machine learning models were implemented in Python. The full code, including, data Preparation, Training and Testing design, model training, and evaluation procedures, is provided here.

1) Data Preparation

In [5]:
```
#Load dataset.
df = pd.read_csv(r"C:\Users\pc\OneDrive - Istanbul Kultur Universitesi\Safa Alreaai\Thesis Report Template\Dataset and Data Analytics\pharma-data-clean.csv")
df.head()
```

Out[5]:

	Distributor	Customer Name	City	Country	Latitude	Longitude	Channel	Sub-channel	Product Name	Product Class	Quantity	Price	Sales	Month	Year	Name of Sales Rep	Manager	Sales Team
0	Gerlach LLC	Fadel-West Pharmacy	Ditzingen	Germany	48.8264	9.0667	Hospital	Government	Exotropin Empizine	Mood Stabilizers	20	785	15700.0	January	2017	Sheila Stones	Britanny Bold	Delta
1	Gerlach LLC	Nader-Gaylord Pharmacy	Backnang	Germany	48.9464	9.4306	Hospital	Private	Robapril	Antipiretics	10	453	4530.0	January	2017	Stella Given	Alisha Cordwell	Charlie
2	Gerlach LLC	McKenzie-Zemlak Pharm	Rheinbach	Germany	50.6256	6.9491	Pharmacy	Institution	Secrelazine Insonamic	Antipiretics	25	694	17350.0	January	2017	Daniel Gates	Alisha Cordwell	Charlie
3	Gerlach LLC	Fritsch-Hauck Pharmaceutical Ltd	Fürth	Germany	49.4783	10.9903	Pharmacy	Retail	Megenorphine	Antimalarial	5	402	2010.0	January	2017	Mary Gerrard	Britanny Bold	Delta
4	Gerlach LLC	Bernier, Murphy and Rau Pharma Plc	Geldern	Germany	51.5197	6.3325	Hospital	Government	Agalsiline	Mood Stabilizers	20	64	1280.0	January	2017	Mary Gerrard	Britanny Bold	Delta

In [6]:
```
#checking the shape of the dataset.
df.shape
```

Out[6]:
```
(129391, 18)
```

In [7]:
```
#print the features of the dataset.
df.columns.tolist()
```

Out[7]:
```
['Distributor',
 'Customer Name',
 'City',
 'Country',
 'Latitude',
 'Longitude',
 'Channel',
 'Sub-channel',
 'Product Name',
 'Product Class',
 'Quantity',
 'Price',
 'Sales',
 'Month',
 'Year',
 'Name of Sales Rep',
 'Manager',
 'Sales Team']
```

In [8]:
```
#checking if there is null values and the type of data.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129391 entries, 0 to 129390
Data columns (total 18 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Distributor        129391 non-null  object
 1   Customer Name      129391 non-null  object
 2   City               129391 non-null  object
 3   Country            129391 non-null  object
 4   Latitude           129391 non-null  float64
 5   Longitude          129391 non-null  float64
 6   Channel            129391 non-null  object
 7   Sub-channel        129391 non-null  object
 8   Product Name       129391 non-null  object
 9   Product Class      129391 non-null  object
 10  Quantity           129391 non-null  int64
 11  Price              129391 non-null  int64
 12  Sales              129391 non-null  float64
 13  Month              129391 non-null  object
 14  Year               129391 non-null  int64
 15  Name of Sales Rep  129391 non-null  object
 16  Manager            129391 non-null  object
 17  Sales Team         129391 non-null  object
dtypes: float64(3), int64(3), object(12)
memory usage: 17.8+ MB
```

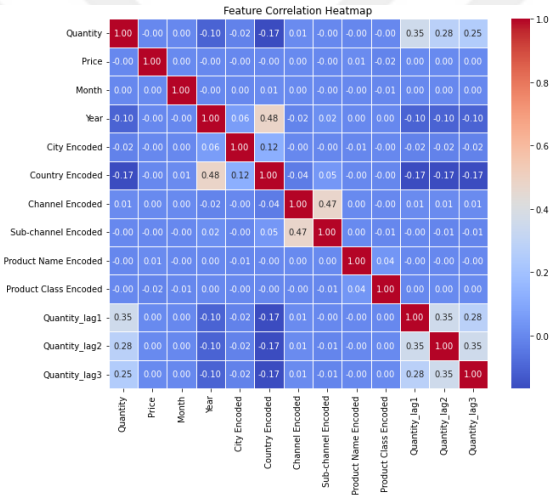
In [9]:
```
#some features will not be useful sand will not give any benefit, so it is better to drop those features.
columns = ['Distributor', 'Customer Name', 'Latitude', 'Longitude','Name of Sales Rep', 'Manager', 'Sales Team','Sales']
df.drop(columns, inplace=True, axis=1)
```

In [10]:
```
#convert Month Names to Numbers.
month_map = {
    'January': 1, 'February': 2, 'March': 3, 'April': 4,
    'May': 5, 'June': 6, 'July': 7, 'August': 8,
    'September': 9, 'October': 10, 'November': 11, 'December': 12
}
df['Month'] = df['Month'].map(month_map)
```

In [11]:
```
#encode categorical Features.
#store encoders in a dictionary.
encoders = {}
#encode and store the encoders
categorical_features = ['City', 'Country', 'Channel', 'Sub-channel', 'Product Name', 'Product Class']
for col in categorical_features:
    le = LabelEncoder()
    df[col + ' Encoded'] = le.fit_transform(df[col])
    encoders[col] = le
df = df.drop(columns=categorical_features)
```

In [12]:
```
#lag features to increase the accuracy.
df["Quantity_lag1"] = df["Quantity"].shift(1)
df["Quantity_lag2"] = df["Quantity"].shift(2)
df["Quantity_lag3"] = df["Quantity"].shift(3)
#after shifting delete nan
df = df.dropna()
```

In [13]:
```
#Correlation matrix.
corr = df.corr()
#Heatmap.
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [14]:
```
#The target variable, Demand, is a continuous variable, meaning it can take any value within a
#range and is not limited to discrete categories.
#This characteristic makes regression models a natural choice for predicting future demand.
```

2) Training and Testing Design

In [15]:
```
#define features and target without lag.
feature_columns = [
    'Month', 'Year','Price', 'City Encoded', 'Country Encoded',
    'Channel Encoded', 'Sub-channel Encoded', 'Product Name Encoded', 'Product Class Encoded',
]

x_withoutlag = df[feature_columns]
y_withoutlag = df['Quantity']
```

In [16]:
```
#define features and target with lag.
feature_columns = [
    'Month', 'Year','Price', 'City Encoded', 'Country Encoded',
    'Channel Encoded', 'Sub-channel Encoded', 'Product Name Encoded', 'Product Class Encoded',
    'Quantity_lag1','Quantity_lag2','Quantity_lag3'
]

x = df[feature_columns]
y = df['Quantity']
```

In [17]:
```
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
#for validation (time-series aware) Time-Based split.
tscv = TimeSeriesSplit(n_splits=3)
#without lag.
#Training: Jan 2017 to Sep 2018.
x_train_withoutlag2 = x_withoutlag[(df['Year'] == 2017) | ((df['Year'] == 2018) & (df['Month'] <= 9))]
y_train_withoutlag2 = y_withoutlag[(df['Year'] == 2017) | ((df['Year'] == 2018) & (df['Month'] <= 9))]

#Testing: oct 2018 to Dec 2018.
x_test_withoutlag2 = x_withoutlag[(df['Year'] == 2018) & (df['Month'] >= 10)]
y_test_withoutlag2 = y_withoutlag[(df['Year'] == 2018) & (df['Month'] >= 10)]
```

In [18]:
```
#with lag.
#Training: Jan 2017 to Sep 2018.
x_train_2 = x[(df['Year'] == 2017) | ((df['Year'] == 2018) & (df['Month'] <= 9))]
y_train_2 = y[(df['Year'] == 2017) | ((df['Year'] == 2018) & (df['Month'] <= 9))]

#Testing: oct 2018 to Dec 2018.
x_test_2 = x[(df['Year'] == 2018) & (df['Month'] >= 10)]
y_test_2 = y[(df['Year'] == 2018) & (df['Month'] >= 10)]
```

In [19]:
```
#checking the shape of the training and testing set without lag.
#Training: Jan 2017 to Sep 2018.Testing: oct 2018 to Dec 2018.
x_train_withoutlag2.shape, x_test_withoutlag2.shape
```

Out[19]:
```
((107494, 9), (21894, 9))
```

In [20]:
```
#checking the shape of the training and testing set with lag.
#Training: Jan 2017 to Sep 2018.Testing: oct 2018 to Dec 2018.
x_train_2.shape, x_test_2.shape
```

Out[20]:
```
((107494, 12), (21894, 12))
```

3) Model Training

In [21]:
```
#Decision Tree Regressor model without lag.
from sklearn.tree import DecisionTreeRegressor
#define the parameter grid for Decision Tree.
param_grid_dt = {
  'max_depth': [3,5],
    'min_samples_split': [8,9],
}
#initialize Grid Search.
grid_dt_withoutlag2 = GridSearchCV(
    DecisionTreeRegressor(),
    param_grid=param_grid_dt,
    cv=tscv,
)
#train the model using grid search.
grid_dt_withoutlag2.fit(x_train_withoutlag2, y_train_withoutlag2)
#print the best parameters.
print("Best parameters (Decision Tree):", grid_dt_withoutlag2.best_params_)
```

```
Best parameters (Decision Tree): {'max_depth': 3, 'min_samples_split': 8}
```

In [22]:
```
#Decision Tree Regressor model with lag.
from sklearn.tree import DecisionTreeRegressor
#define the parameter grid for Decision Tree.
param_grid_dt = {
  'max_depth': [3,5],
    'min_samples_split': [8,9],
}
#initialize Grid Search.
grid_dt2 = GridSearchCV(
    DecisionTreeRegressor(),
    param_grid=param_grid_dt,
    cv=tscv,
)
#train the model using grid search.
grid_dt2.fit(x_train_2, y_train_2)
#print the best parameters.
print("Best parameters (Decision Tree):", grid_dt2.best_params_)
```

```
Best parameters (Decision Tree): {'max_depth': 5, 'min_samples_split': 8}
```

In [23]:
```
#Random Forest Regressor model without lag.
from sklearn.ensemble import RandomForestRegressor
#define the parameter grid for Random Forest.
param_grid_rf = {
    'n_estimators': [60, 80],
    'max_depth': [5,7]
}
#initialize Grid Search.
grid_rf_withoutlag2 = GridSearchCV(
    RandomForestRegressor(),
    param_grid=param_grid_rf,
    cv=tscv,
)
#train the model using grid search.
grid_rf_withoutlag2.fit(x_train_withoutlag2, y_train_withoutlag2)
#print the best parameters.
print("Best parameters (Random Forest):", grid_rf_withoutlag2.best_params_)
```

```
Best parameters (Random Forest): {'max_depth': 5, 'n_estimators': 60}
```

In [24]:
```
#Random Forest Regressor model with lag.
from sklearn.ensemble import RandomForestRegressor
#define the parameter grid for Random Forest.
param_grid_rf = {
    'n_estimators': [60, 80],
    'max_depth': [5,7]
}
#initialize Grid Search.
grid_rf2 = GridSearchCV(
    RandomForestRegressor(),
    param_grid=param_grid_rf,
    cv=tscv,
)
#train the model using grid search.
grid_rf2.fit(x_train_2, y_train_2)
#print the best parameters.
print("Best parameters (Random Forest):", grid_rf2.best_params_)
```

```
Best parameters (Random Forest): {'max_depth': 7, 'n_estimators': 80}
```

In [25]:
```
#XGBoost model without lag.
import xgboost as xgb
#define the parameter grid for XGBoost.
param_grid_xgb = {
    'n_estimators': [40, 60],
    'max_depth': [3, 4],
    'learning_rate': [0.05,0.1]
}
#initialize Grid Search.
grid_xgb_withoutlag2 = GridSearchCV(
    xgb.XGBRegressor(),
    param_grid=param_grid_xgb,
    cv=tscv,
)
#train the model using grid search.
grid_xgb_withoutlag2.fit(x_train_withoutlag2, y_train_withoutlag2)
#print the best parameters and score.
print("Best parameters (XGBoost):", grid_xgb_withoutlag2.best_params_)
```

```
Best parameters (XGBoost): {'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 40}
```

In [26]:
```
#XGBoost model with lag.
import xgboost as xgb
#define the parameter grid for XGBoost.
param_grid_xgb = {
    'n_estimators': [40, 60],
    'max_depth': [3, 4],
    'learning_rate': [0.05,0.1]
}
#initialize Grid Search.
grid_xgb2 = GridSearchCV(
    xgb.XGBRegressor(),
    param_grid=param_grid_xgb,
    cv=tscv,
)
#train the model using grid search.
grid_xgb2.fit(x_train_2, y_train_2)
#print the best parameters and score.
print("Best parameters (XGBoost):", grid_xgb2.best_params_)
```

```
Best parameters (XGBoost): {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 60}
```

In [27]:
```
#predict
y_pred_modelDecisionTreeRegressor_withoutlag2 = grid_dt_withoutlag2.predict(x_test_withoutlag2)
#remove negtive values.
y_pred_modelDecisionTreeRegressor_withoutlag2 = np.maximum(y_pred_modelDecisionTreeRegressor_withoutlag2, 0)
#convert the values to intergers.
y_pred_modelDecisionTreeRegressor_withoutlag2 = np.round(y_pred_modelDecisionTreeRegressor_withoutlag2).astype(int)
```

In [28]:
```
#predict
y_pred_modelDecisionTreeRegressor2 = grid_dt2.predict(x_test_2)
#remove negtive values.
y_pred_modelDecisionTreeRegressor2 = np.maximum(y_pred_modelDecisionTreeRegressor2, 0)
#convert the values to intergers.
y_pred_modelDecisionTreeRegressor2 = np.round(y_pred_modelDecisionTreeRegressor2).astype(int)
```

In [29]:
```
#predict
y_pred_modelRandomForestRegressor_withoutlag2 = grid_rf_withoutlag2.predict(x_test_withoutlag2)
#remove negtive values.
y_pred_modelRandomForestRegressor_withoutlag2 = np.maximum(y_pred_modelRandomForestRegressor_withoutlag2, 0)
#convert the values to intergers.
y_pred_modelRandomForestRegressor_withoutlag2 = np.round(y_pred_modelRandomForestRegressor_withoutlag2).astype(int)
```

In [30]:
```
#predict
y_pred_modelRandomForestRegressor2 = grid_rf2.predict(x_test_2)
#remove negtive values.
y_pred_modelRandomForestRegressor2 = np.maximum(y_pred_modelRandomForestRegressor2, 0)
#convert the values to intergers.
y_pred_modelRandomForestRegressor2 = np.round(y_pred_modelRandomForestRegressor2).astype(int)
```

In [31]:
```
#predict
y_pred_modelXGBRegressor_withoutlag2 = grid_xgb_withoutlag2.predict(x_test_withoutlag2)
#remove negtive values.
y_pred_modelXGBRegressor_withoutlag2 = np.maximum(y_pred_modelXGBRegressor_withoutlag2, 0)
#convert the values to intergers.
y_pred_modelXGBRegressor_withoutlag2 = np.round(y_pred_modelXGBRegressor_withoutlag2).astype(int)
```

In [32]:
```
#predict
y_pred_modelXGBRegressor2 = grid_xgb2.predict(x_test_2)
#remove negtive values.
y_pred_modelXGBRegressor2 = np.maximum(y_pred_modelXGBRegressor2, 0)
#convert the values to intergers.
y_pred_modelXGBRegressor2 = np.round(y_pred_modelXGBRegressor2).astype(int)
```

4) Model Evaluation

In [33]:
```
#Safe MAPE calculation that avoids division by zero.
def safe_mape(y_true, y_pred):
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    mask = y_true != 0
    if mask.sum() == 0:
        return np.nan
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

#Compute main error metrics.
def compute_metrics(y_true, y_pred):
    if len(y_true) != len(y_pred):
        return dict(MAE=np.nan, RMSE=np.nan, MAPE=np.nan, R2=np.nan)
    return dict(
        MAE=mean_absolute_error(y_true, y_pred),
        RMSE=np.sqrt(mean_squared_error(y_true, y_pred)),
        MAPE=safe_mape(y_true, y_pred),
        R2=r2_score(y_true, y_pred),
    )


split_2_train, split_2_test = "2017-01 → 2018-09", "2018-10 → 2018-12"

models = [
    ("Decision Tree","No Lag",  split_2_train, split_2_test, y_test_withoutlag2, y_pred_modelDecisionTreeRegressor_withoutlag2),
    ("Decision Tree","With Lag",split_2_train, split_2_test, y_test_2,          y_pred_modelDecisionTreeRegressor2),

    ("Random Forest","No Lag",  split_2_train, split_2_test, y_test_withoutlag2, y_pred_modelRandomForestRegressor_withoutlag2),
    ("Random Forest","With Lag",split_2_train, split_2_test, y_test_2,          y_pred_modelRandomForestRegressor2),
```

```
split_2_train, split_2_test = "2017-01 → 2018-09", "2018-10 → 2018-12"

models = [
    ("Decision Tree","No Lag",  split_2_train, split_2_test, y_test_withoutlag2, y_pred_modelDecisionTreeRegressor_withoutlag2),
    ("Decision Tree","With Lag",split_2_train, split_2_test, y_test_2,          y_pred_modelDecisionTreeRegressor2),

    ("Random Forest","No Lag",  split_2_train, split_2_test, y_test_withoutlag2, y_pred_modelRandomForestRegressor_withoutlag2),
    ("Random Forest","With Lag",split_2_train, split_2_test, y_test_2,          y_pred_modelRandomForestRegressor2),

    ("XGBoost","No Lag",        split_2_train, split_2_test, y_test_withoutlag2, y_pred_modelXGBRegressor_withoutlag2),
    ("XGBoost","With Lag",      split_2_train, split_2_test, y_test_2,          y_pred_modelXGBRegressor2),
]

rows = []
for model, lag, tr, te, y_true, y_pred in models:
    m = compute_metrics(y_true, y_pred)

    rows.append({
        "Title": f"{model} • {lag} • Train {tr} | Test {te}",
        "MAE": round(m["MAE"], 2),
        "RMSE": round(m["RMSE"], 2),
        "MAPE (%)": None if pd.isna(m["MAPE"]) else round(m["MAPE"], 2),
        "R²": None if pd.isna(m["R2"]) else round(m["R2"], 2),
    })

df = pd.DataFrame(rows)
print(df.to_string(index=False))
```

```
                                                                Title   MAE  RMSE  MAPE (%)   R²
  Decision Tree • No Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 18.41 25.36    328.81 0.02
Decision Tree • With Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 16.00 23.17    246.34 0.19
  Random Forest • No Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 18.27 25.34    319.25 0.03
Random Forest • With Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 15.75 22.96    236.55 0.20
        XGBoost • No Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 18.56 25.32    333.79 0.03
      XGBoost • With Lag • Train 2017-01 → 2018-09 | Test 2018-10 → 2018-12 15.74 22.95    234.13 0.20
```

In [34]:

```
#prepare training and testing data.
df_train = pd.DataFrame(x_train_2, columns=feature_columns)
df_train['Quantity'] = y_train_2.values if hasattr(y_train_2, 'values') else y_train_2

df_test = pd.DataFrame(x_test_2, columns=feature_columns)
df_test['Quantity'] = y_test_2.values if hasattr(y_test_2, 'values') else y_test
df_test['y_pred'] = y_pred_modelXGBRegressor2

train_columns = list(df_train.columns)
test_columns = list(df_test.columns)

#create a new Excel workbook.
wb = Workbook()
ws = wb.active
ws.title = "All_Data_Side_by_Side"

#first row: Merged labels.
#training part.
ws.merge_cells(start_row=1, start_column=1, end_row=1, end_column=len(train_columns)-1)
ws.cell(row=1, column=1).value = "x_train"
ws.cell(row=1, column=1).alignment = Alignment(horizontal='center')

ws.cell(row=1, column=len(train_columns)).value = "y_train"
ws.cell(row=1, column=len(train_columns)).alignment = Alignment(horizontal='center')

#testing part.
start_test_col = len(train_columns) + 2
ws.merge_cells(start_row=1, start_column=start_test_col, end_row=1, end_column=start_test_col + len(test_columns) - 3)
ws.cell(row=1, column=start_test_col).value = "x_test"
ws.cell(row=1, column=start_test_col).alignment = Alignment(horizontal='center')

ws.cell(row=1, column=start_test_col + len(test_columns) - 2).value = "y_test"
ws.cell(row=1, column=start_test_col + len(test_columns) - 2).alignment = Alignment(horizontal='center')
```

```
ws.cell(row=1, column=start_test_col + len(test_columns) - 2).value = "y_test"
ws.cell(row=1, column=start_test_col + len(test_columns) - 2).alignment = Alignment(horizontal='center')

ws.cell(row=1, column=start_test_col + len(test_columns) - 1).value = "y_pred"
ws.cell(row=1, column=start_test_col + len(test_columns) - 1).alignment = Alignment(horizontal='center')

#second row: Actual column names.
ws.append(train_columns + [''] + test_columns)

#write data rows side by side.
max_len = max(len(df_train), len(df_test))
for i in range(max_len):
    row = []

    #training data row.
    if i < len(df_train):
        row.extend(df_train.iloc[i].tolist())
    else:
        row.extend([''] * len(train_columns))

    row.append('')

    #testing data row.
    if i < len(df_test):
        row.extend(df_test.iloc[i].tolist())
    else:
        row.extend([''] * len(test_columns))

    ws.append(row)

#save the file.
wb.save("train_test_side_by_side4.xlsx")
```