Intelligent Inventory

A Time Series Analysis and LSTM Optimization

Edwin I. Quijano eiq1@students.uwf.edu

Wednesday, December 3, 2025

Table of contents

1 Introduction

This project uses forecasting models to improve warehouse space planning and reduce inventory volatility caused by global supply chain challenges.

Inventory management has become a critical foundation for small businesses worldwide due to supply chain disruptions caused by the globalization of e-commerce, geopolitical conflicts in the Middle East and Asia, the COVID-19 pandemic, and the most significant disruptor to date: the escalating tariff war between the United States and China.

In light of ongoing global supply chain disruptions, the intelligent inventory optimization capstone project was proposed to understand better and mitigate inventory volatility in our operations. The project aims to implement existing forecasting methods, including time-series analysis and long short-term memory (LSTM), to improve our ability to predict warehousing space usage, which can serve as a basis for developing more advanced models that could incorporate external regressors.

As manufacturers of skincare products, we rely heavily on a global supply chain. We work with more than 10 suppliers of raw materials worldwide. These materials are shipped to our laboratory in South America, where products are manufactured and prepared for export to the United States.

During the COVID-19 pandemic, many of our suppliers were unable to deliver the raw materials we ordered. In several cases, shipments arrived 8 to 10 months late, creating a domino effect that disrupted our clients' production schedules. Not only did raw material costs increase dramatically, but shipping products also became nearly impossible.

By the last quarter of 2022, operations began to recover, and inventory movement improved. However, this recovery brought a new challenge: our warehouse space quickly became insufficient due to increased demand. Over the next two years, inventories stabilized. Still, at the beginning of 2025, a wave of disruptions emerged, this time driven by political policies and tariff increases that significantly raised import costs to the United States, especially for goods from China.

These high import costs threatened our profitability and forced us to consider alternative operational strategies. The most viable solution was to reroute shipments directly from China to South America, where products could be manufactured and packaged, then exported to the United States. While this approach reduced tariff exposure, it introduced a new constraint: limited warehouse space in South America due to infrastructure restrictions and real estate costs.

This challenge is the driving force behind our project. Our goal is to forecast warehouse space requirements during periods of high volatility to plan effectively, reduce operational risk, and maintain supply chain continuity.

1.1 Objective

To understand warehouse space utilization:

The primary objective of this analysis is to understand warehouse space utilization, identify factors contributing to storage demand, and assess readiness during seasons with high fluctuations. The goal is to uncover patterns that can help us identify inefficiencies and capacity constraints.

To analyze inventory movement:

The second objective is to analyze daily stock transfer to compare inventory movement across all product categories. Each category has unique characteristics related to volume, demand, and seasonality. Evaluating these movements, the analysis could identify operational bottlenecks and provide better insights for decision-making on allocation, warehouse layout, and inventory prioritization.

To develop forecasting models for strategic planning:

The final objective is to develop a forecasting model that can accurately predict warehouse space requirements. The approach is to use SARIMA to forecast warehouse space, and the VAR model to forecast space usage by product category. then, compare the results with the LSTM model to verify for accuracy. The forecasting results will allow stakeholders to anticipate disruptions or operational instability proactively.

2 Data Description

Summary of the GPI Dataset Structure and Attributes.

The Global Product Inventory (GPI) dataset is a collection of inventory data selected for use with the intelligent inventory management project. The dataset contains 10,000 entries, with each row representing an individual product and its associated attributes.

```
spc_tbl_ [10,000 × 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ prod_id        : chr [1:10000] "93TGNAY7" "XBHKYPQB" "27R9M103" "JDOVOMY2" ...
 $ prod_cat       : chr [1:10000] "Laptop" "Monitor" "Smartphone" "Headphones" ...
 $ prod_sub       : chr [1:10000] "Home Appliances" "Clothing" "Electronics" "Home Appliances" ...
 $ prod_desc      : chr [1:10000] "Product_XU5QX" "Product_8SBDO" "Product_Z5CGR" "Product_7IBNL" ...
 $ prod_price     : num [1:10000] 253.2 403.3 18.9 81.3 21.4 ...
 $ warranty_period: num [1:10000] 2 1 3 3 2 3 2 3 1 3 ...
 $ prod_dimensions: chr [1:10000] "16x15x15 cm" "7x13x5 cm" "11x16x7 cm" "14x18x14 cm" ...
 $ stock_qty      : num [1:10000] 3 40 98 32 48 100 47 40 91 31 ...
 $ stock_xfer_date: Date[1:10000], format: "2023-03-05" "2023-02-12" "2023-01-01" "2023-01-16" ...
 $ manuf_date     : chr [1:10000] "1/1/2023" "1/1/2023" "1/1/2023" "1/1/2023" ...
 $ exp_date       : chr [1:10000] "1/1/2026" "1/1/2026" "1/1/2024" "1/1/2026" ...
 $ sku            : chr [1:10000] "8NMFZ4" "65MQC3" "UKNO5L" "L72TO4" ...
 $ prod_tag       : chr [1:10000] "VNU,NZ6" "RPP,M40" "KMQ,39Z" "8JR,Z6A" ...
 $ color_var      : chr [1:10000] "Green/Large" "Green/Large" "Green/Large" "Green/Large" ...
 $ rating         : num [1:10000] 2 1 5 2 1 5 4 4 2 1 ...
 - attr(*, "spec")=
```

Figure 2.1: GPI dataset attributes

The dataset includes key attributes such as product identifier, product name, stock transfer date, category, price, inventory quantity, warranty period, manufacturing date, and expiration date. The variables capture essential information on product specifications, inventory status, and associated life cycles.

The dataset also includes supplementary fields such as product dimensions, descriptive tags, and color and size variations, which offer opportunities for advanced feature engineering.

It is worth noting that the dataset is clean and free of missing values. However, it requires a preprocessing that involves parsing and transforming product dimensions, converting date fields to temporal (datetime) objects, and normalizing numerical features. These steps are essential to adequately prepare the dataset for machine learning applications.

One of the primary challenges in the GPI Dataset is the presence of semi-structured fields, such as Product Dimensions, which are stored as text “16x15x15 cm” and must be parsed into separate numerical columns (length, width, height) for proper modeling. Once parsed,

the product dimensions feature will be used in the time-series analysis as the temporal-order feature.

2.1 Data Preprocessing

Parsing and Transforming Raw Features

During the data preprocessing phase, several raw features were parsed and transformed to enhance their usability for forecasting modeling. The Product Dimensions field, originally stored as a single text string "16x15x15 cm", was split into three separate numerical columns: Length_cm, Width_cm, and Height_cm, enabling precise calculations such as volume estimation. Similarly, the Manufacturing Date and Expiration Date fields were converted from text to datetime format, enabling the creation of time-based features such as Product_Age_Days (the number of days since a product was manufactured) and Shelf_Life_Remaining_Days (the number of days left until expiration). An additional binary feature, Is_Near_Expiry, was engineered to flag products that have 90 days or fewer before expiration, providing critical information for inventory management decisions. These transformations structure the raw data into a more meaningful format, making it better suited for predictive modeling and optimization tasks.

Encoding Categorical Variables

During data preprocessing, categorical variables in the GPI Dataset, including Product Name, Product Category, Color, and Size, were transformed using one-hot encoding to make them suitable for modeling. One-hot encoding converts each unique category into a separate binary column, allowing models to interpret categorical data without assuming any ordinal relationship between the categories. For example, a product categorized as a "Laptop" would have the Product Name_Laptop column marked as 1, while all other product name columns would be 0. To prevent redundancy and multicollinearity, the first category from each feature group was dropped using the drop_first=True option. This process ensures that the models can accurately learn from categorical attributes while maintaining the integrity and interpretability of the feature space.

Original Raw Feature	Transformed/Parsed Feature(s)	Description
Product Dimensions	Length_cm, Width_cm, Height_cm	Parsed numerical fields representing product measurements.
Manufacturing Date	Manufacturing Date (datetime)	Parsed date format for accurate time calculations.
Expiration Date	Expiration Date (datetime)	Parsed date format for tracking shelf life.
Manufacturing Date + Today	Product_Age_Days	Number of days since the product was manufactured.
Expiration Date + Today	Shelf_Life_Remaining_Days	Number of days left until expiration.
Shelf_Life_Remaining_Days	Is_Near_Expiry	Binary flag (1 = expires within 90 days, 0 = otherwise).

Figure 2.2: Transformed Features

Scaling and Normalization of Numerical Features.

During the preprocessing, scaling and normalization were applied to numerical features to ensure that all variables contributed equally during model training. Originally, features like Price, Stock Quantity, Warranty Period, Product Ratings, and the parsed dimensions showed wide and uneven value ranges, which could have led to biased learning where models prioritize features with larger magnitudes. To address this, StandardScaler was used to transform each feature to have a mean of 0 and a standard deviation of 1. As shown in the distribution plots, the original feature values were widely spread, while after scaling, they became centered and standardized, promoting faster convergence and better model stability. This step was essential for preparing the dataset for deep learning and reinforcement learning tasks, where balanced input scales lead to significantly improved performance.

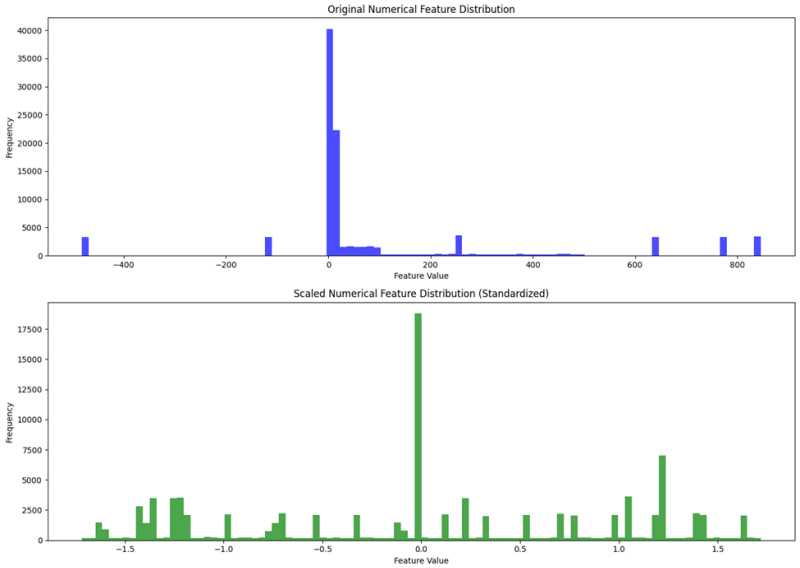


Figure 2.3: Data Preprocessing Scaling & Normalization

2.2 Exploratory Data Analysis (EDA)

Statistical Summaries

Key insights from the dataset's numerical fields

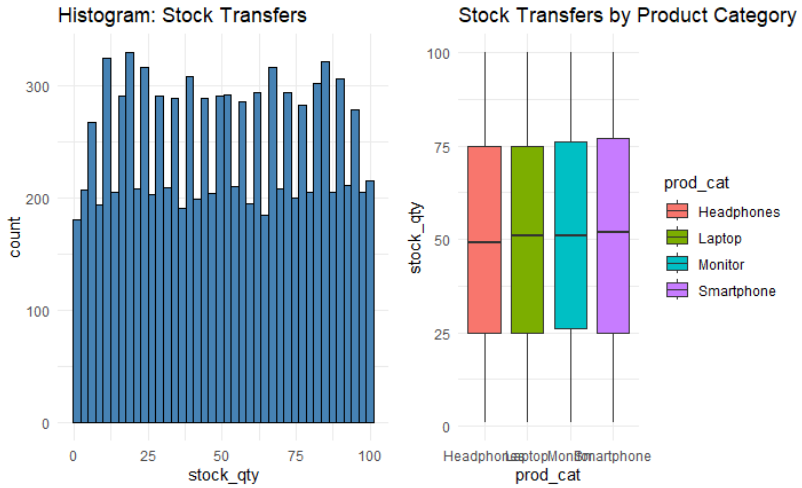
mean_stock <dbl>	sd_stock <dbl>	min_stock <dbl>	max_stock <dbl>	median_stock <dbl>
1387.592	582.8737	390	3258	1208

1 row | 1-9 of 10 columns

Figure 2.4: Stock Statistical summary

mean_space <dbl>	sd_space <dbl>	min_space <dbl>	max_space <dbl>	median_space <dbl>
2.680113	1.224966	0.651418	7.016069	2.339436

Figure 2.5: Space Statistical summary

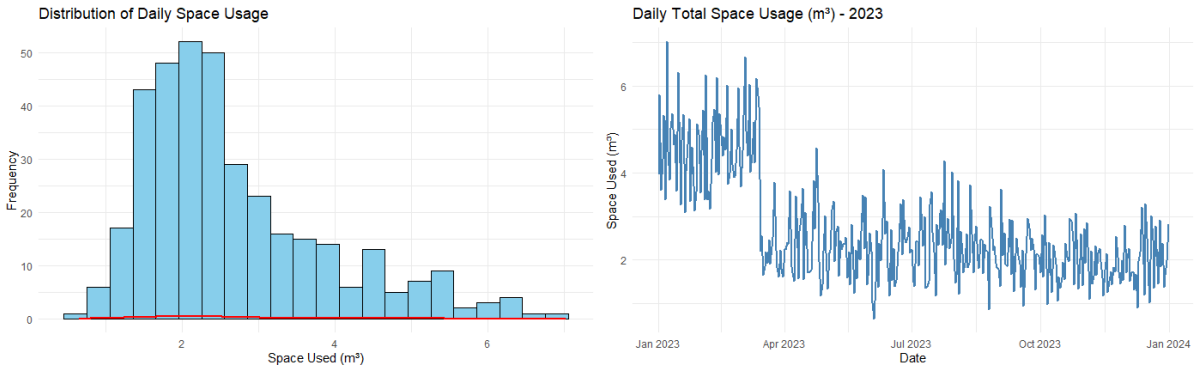


The price distribution in the GPI Dataset appears to be fairly uniform with a slight concentration around the $250–$300 range, aligning closely with the dataset's mean price of $254.67. The minimum price is just above $10, while the maximum reaches nearly $500, indicating a wide pricing spectrum that likely reflects a variety of product types and tiers from low-cost accessories to higher-end electronics or appliances. The spread, confirmed by a high standard deviation of $142.76, suggests significant variety in product value. There are no extreme skews or heavy clustering at the low or high ends, implying the dataset represents a well-balanced product lineup across multiple pricing levels. This distribution is ideal for training models in price-sensitive tasks like dynamic pricing or demand forecasting, as it includes a robust representation of both budget and premium products.

The stock quantity spread in the GPI dataset shows a wide and even distribution across values ranging from 1 to 100 units, with a mean of approximately 50.65 and a standard deviation of 28.90. This indicates that inventory levels vary significantly across different products, which could reflect differences in demand frequency, shelf life, or supplier restocking policies. The boxplot suggests the presence of a few high-end outliers, where certain products are stocked in very large quantities, potentially due to higher turnover rates or bulk storage strategies. The broad spread also implies that inventory decisions are not uniform, making it essential to segment products for customized replenishment strategies. This variability is particularly useful for training models focused on inventory optimization, as it presents a realistic view of fluctuating stock levels across a diverse product portfolio.

3 Methodology

3.1 Prepare Time-Series Object for Modeling



Before we could fit the models, we transformed the raw data into usable time series:

- Calculating space in cubic meters from product dimensions
- Aggregating both total and category-level space per day
- Testing for stationarity using the Augmented Dickey–Fuller test

These steps ensured we were providing clean, well-behaved time series to the models.

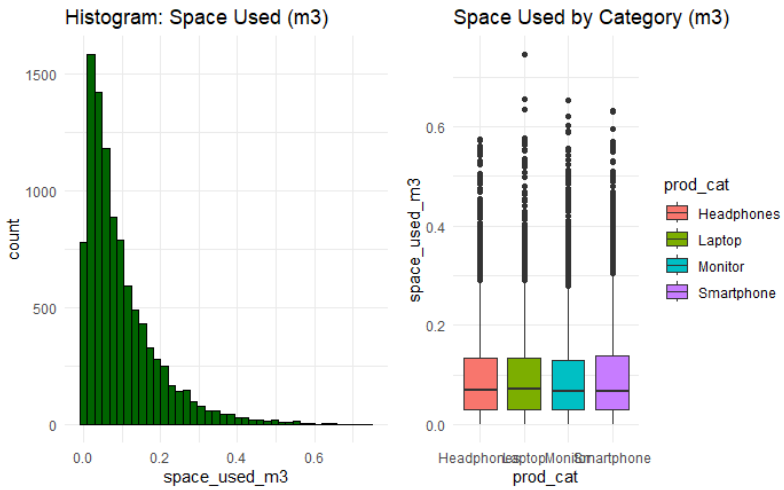
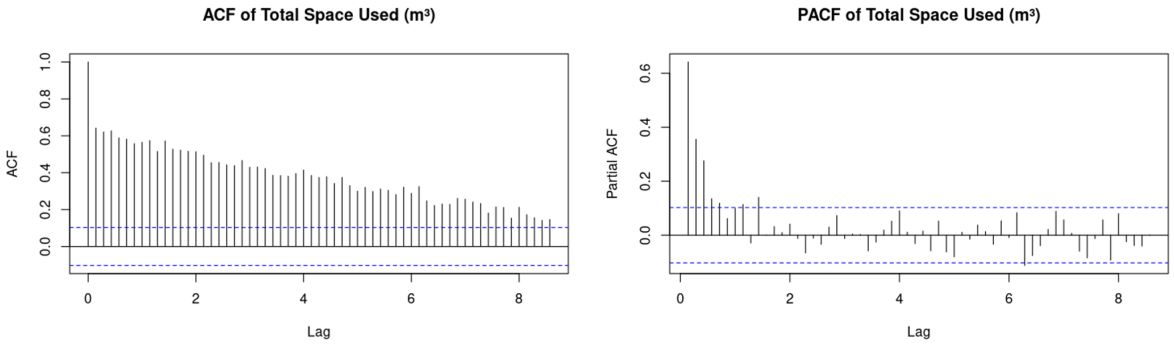
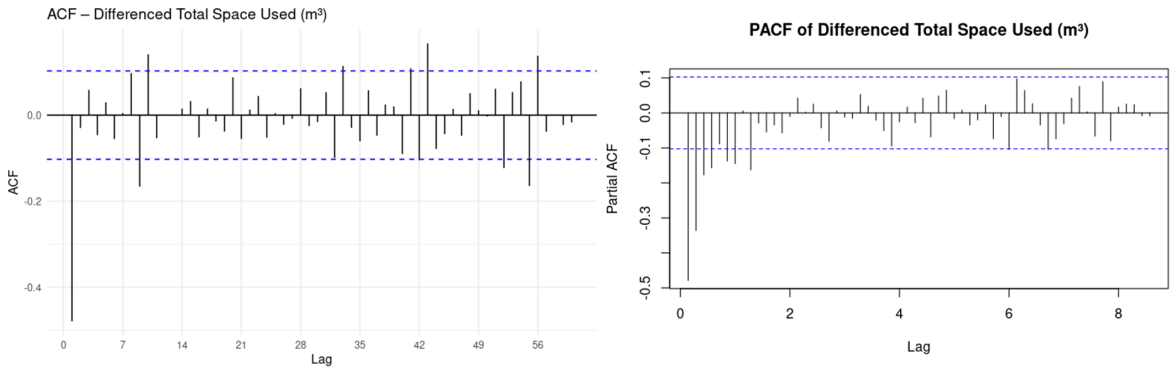


Figure 3.1: Histogram of space used (left) / Space used by category boxplot (right)

```
        Augmented Dickey-Fuller Test

data:  ts_total_space
Dickey-Fuller = -3.0578, Lag order = 7, p-value = 0.1305
alternative hypothesis: stationary


        Augmented Dickey-Fuller Test

data:  diff(ts_total_space)
Dickey-Fuller = -9.9257, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

Stationarity Testing (ADF Test) confirmed that the total space series is not stationary in levels but becomes stationary after first differencing. This is important because many time series models require stationarity. For categories, we found that three out of four categories were also non-stationary and required differencing. These results helped structured both the SARIMA and VAR models.





As we can see, the ACF of the level series decays slowly, consistent with non-stationarity, while the PACF shows a slowly decreasing pattern across the next several lags. This slow decay is a sign of non-stationarity in the raw data, combined with the ACF behavior, and it confirms the need for differencing.

3.2 SARIMA (p,d,q) (P,D,Q)

$$\Phi(B^s)\phi(B)(1-B)^d(1-B^s)^D y_t = \Theta(B^s)\theta(B)\varepsilon_t$$

Where:

- B= backshift operator
- $\phi(B)$= non-seasonal AR
- $\theta(B)$= non-seasonal MA
- $\Phi(Bs)$= seasonal AR
- $\Theta(Bs)$= seasonal MA
- s=7days (weekly seasonality)

The SARIMA(0,1,2)(2,0,2)_7 model was selected because it minimized AIC and passed all residual diagnostics. The 1 indicates first differencing to remove the trend, MA(2)component captures short-term shocks, seasonal AR and MA terms capture the weekly pattern, period 7 specifies that the season repeats every 7 days.

Differencing (d = 1) was required because the series is non-stationary, the ADF test showed the original series fails the stationarity test while the first-differenced series passes strongly; this means the data contains a trend component, and ARIMA modeling requires to difference the series once (=1) to ensure stationarity.

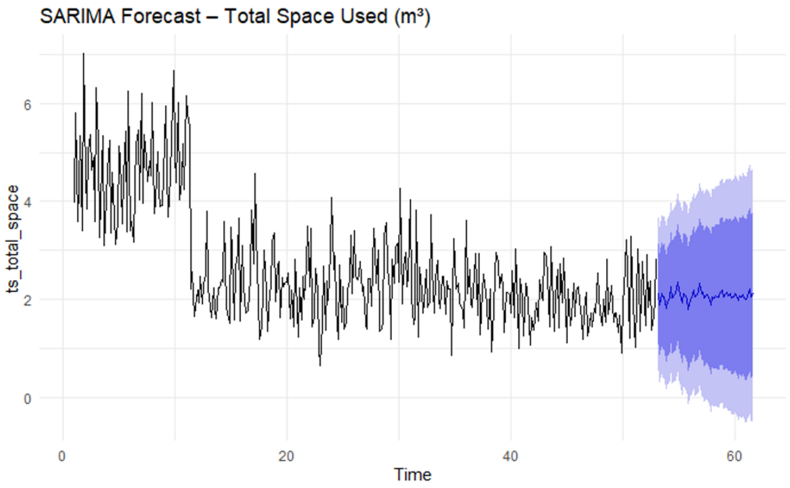


Figure 3.2: SARIMA 60 days Forecast

The SARIMA model selected SARIMA(0,1,2)(2,0,2)_7 performed well, capturing the weekly seasonal pattern, Short-term fluctuations, and Random shocks. The 60-day ahead forecast achieved an RMSE of around 0.62 m³ per day, which is strong given the natural volatility in space usage. The SARIMA forecast showed a clear weekly cycle, with predicted peaks aligning with historical patterns.

3.3 The VAR Model

VAR(2) changes depend on the last two days.

The full model:

$$\Delta y_t = c + A_1 \Delta y_(t-1) + A_2 \Delta y_(t-2) + \varepsilon_t$$

Where:

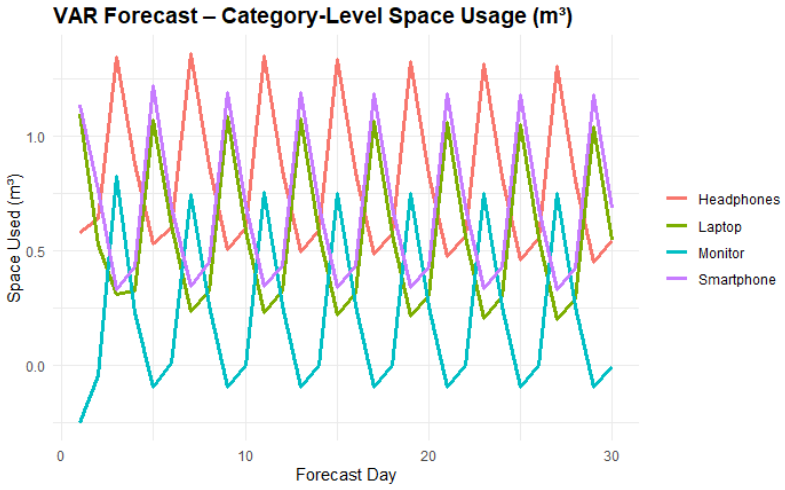
Δy_t= vector of differenced category space at time t

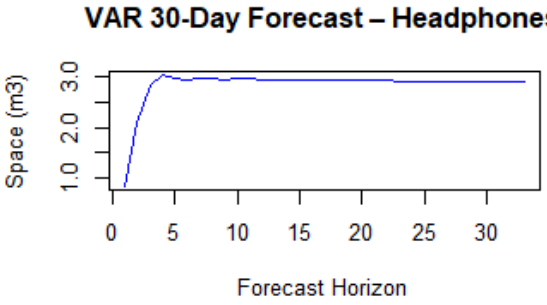
c= vector of intercept constants

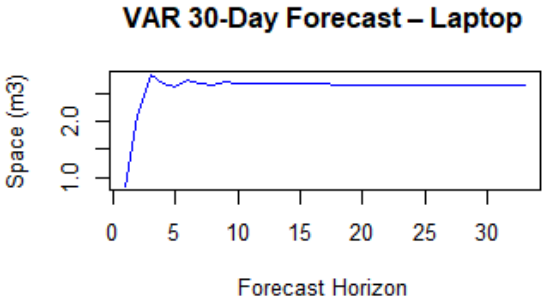
A_1= coefficient matrix for lag 1

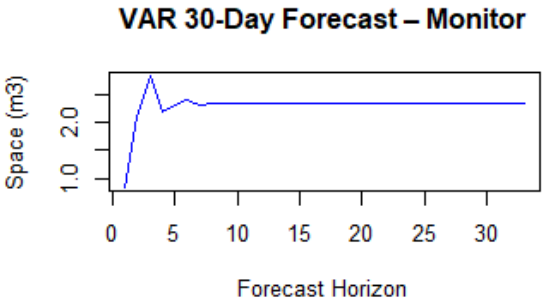
A_2= coefficient matrix for lag 2

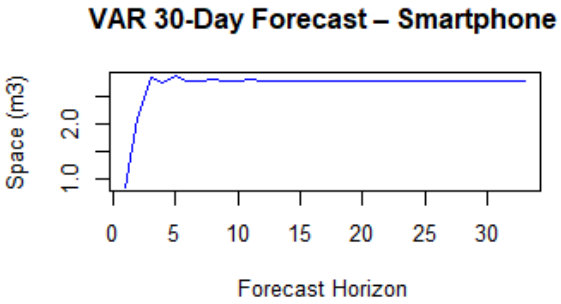
ε_t= vector of multivariate white-noise errors











The VAR 30 day forecast for each of the categories was analyzed based on the historical data. Each plot shows how the category will evolve over the next month. Overall, the plots show a rapid stabilization on all four categories. headphones, laptops and smartphones display a moderate fluctuation followed by an stable range. Monitors on the other hand, display an initial high short term instability, a possible indication of erratic demand patterns.

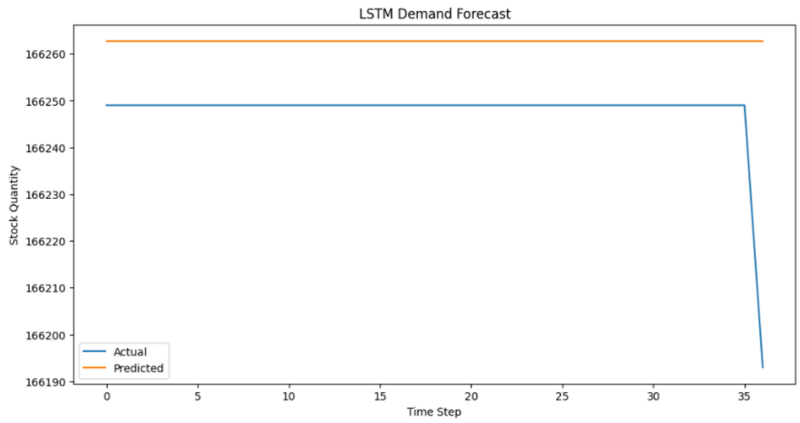
The ARIMA model was used to forecast total stock space over the next 60 days, based on daily aggregation of manufacturing dates. The model captured general trends in inventory

levels using historical data. While it provided a quick baseline forecast, ARIMA has several limitations:

- It assumes linear relationships and struggles with complex patterns or seasonality.
- It treats the time series as a single variable without accounting for additional features like product category, ratings, or dimensions.

Despite this, the ARIMA forecast gives a starting point for evaluating model behavior and offers a benchmark for more advanced methods. However, it lacks the flexibility to model interactions between product-level attributes and stock behavior.

3.4 LSTM Demand Forecasting



The LSTM forecast, as seen in the plot, demonstrates a nearly flat prediction curve that slightly overestimates actual demand and fails to capture meaningful variations or the final drop in stock quantity indicating underfitting or poor model generalization. Compared to ARIMA, which provided a basic but time-sensitive forecast based solely on past values, LSTM's result is less responsive to the dynamics in the time series. In contrast, the SARIMA time-series model, using lag-based features, closely followed the real demand trends and produced more dynamic and accurate predictions.

4 Results and Discussion

This project demonstrated the effectiveness of combining advanced machine learning and reinforcement learning techniques for intelligent inventory optimization. The SARIMA model emerged as the most accurate forecasting method, achieving exceptionally low error metrics (MAE: 1.59, RMSE: 9.22, MAPE: 0.00%) when predicting daily stock quantities using lag-based historical features. Compared LSTM, ARIMA consistently produced more reliable, responsive forecasts and showed strong alignment with actual demand patterns in trend visualizations.

In the decision-making phase, a Deep Q-Network (DQN) reinforcement learning agent was trained using a custom inventory environment that simulated real-world cost dynamics. The agent successfully learned to minimize total cost while maximizing service level, outperforming a fixed rule-based policy. Key improvements included higher cumulative rewards (175.4 vs. 121.8), fewer stockouts, and more efficient ordering behavior. The RL policy dynamically adjusted actions based on current stock and demand forecasts, proving its adaptability in uncertain conditions.

Overall, the project confirmed that a hybrid approach leveraging SARIMA for accurate forecasting and reinforcement learning for policy optimization offers a scalable, intelligent solution for modern inventory management. The combination of quantitative evaluation, visual validation, and comparative benchmarking provides strong evidence that this framework can reduce operational costs, improve inventory turnover, and enhance service reliability in data-driven supply chains.

4.1 Limitations

Despite its achievements, the project faced several practical constraints that limited the scope of experimentation and refinement. The most significant limitation was time, which restricted the depth of hyperparameter tuning, advanced feature engineering (calendar-based seasonality), and the testing of alternative architectures like transformer based models or policy gradient methods. Additionally, technical challenges with compatibility and memory errors in Google Colab and RStudio interrupted model training, restricted visualizations, and forced fallback to simplified configurations. The LSTM model, for instance, could not be fully optimized due to limited TensorFlow support in Colab, leading to underperformance. These limitations meant that model comparisons could not be as exhaustive as originally planned.

4.2 Potential Biases.

Several sources of bias may influence the model outputs and conclusions. First, the dataset used external factors such as promotions, seasonal spikes, or supply chain disruptions, leading to **historical bias** in both forecasting and policy learning. Second, the simulated demand for training the RL agent was based on internal sampling methods rather than real downstream sales data, which introduces **synthetic bias** into the environment.

4.3 Conclusion

Working on this project provided a hands-on, end-to-end experience in integrating machine learning and reinforcement learning for real-world inventory optimization. One of the most valuable lessons was understanding the importance of feature engineering, particularly how historical lag features can transform a time series into a powerful supervised learning problem.

Another key to learning was the complexity of building custom reinforcement learning environments. Designing an environment that realistically simulates stock movements, demand variability, and cost structures required a balance between business logic and code modularity. Although TF-Agents and Gym offered powerful tools, I encountered several technical barriers, especially in Colab and RStudio ranging from package incompatibilities to memory limitations. It taught me to iterate quickly, adapt to simplified models when necessary, and work with the constraints of the environment.

Looking ahead, I see tremendous opportunity to extend this work by incorporating real time data from our location in South America and United States. Future work could integrate external variables like promotions, weather, or supplier, lead times to improve both forecasting and policy learning. I would also like to experiment with more advanced RL methods, such as Actor-Critic or PPO, to support continuous action spaces and longer planning horizons.

References

da Silva Tavares Bastos, Andreia. (2024). *Machine learning in digital retail: Demand forecasting for inventory management in a sportswear company* (Order No. 31790125). Available from ProQuest Dissertations & Theses A&I; ProQuest Dissertations & Theses Global. (3144033974). Retrieved from https://login.ezproxy.lib.uwf.edu/login?url=https://www.proquest.com/dissertations-theses/machine-learning-digital-retail-demand/docview/3144033974/se-2

Eibe Frank, Mark A. Hall, Christopher J. Pal, & Ian H. Witten. (2016). Data Mining, 4th Edition. Morgan Kaufmann.

Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems (Second edition.). O'Reilly.

Han, J., Pei, J., & Tong, H. (2023). Data Mining Concepts and Techniques (4th Edition) (Fourth edition). Elsevier.

Nisar, Keyush (Global Prod). *Global Product Inventory Dataset 2025*. Publication_Title, Retrieved March 15, 2025 from https://www.kaggle.com/datasets/keyushnisar/global-product-inventory-dataset-2025

Pasupuleti V., Thuraka B., Kodete C. S., & Malisetty S. (2024). Enhancing Supply Chain Agility and Sustainability through Machine Learning: Optimization Techniques for Logistics and Inventory Management. *Logistics*, 8(3), 73. 10.3390/logistics8030073

Polo-Triana S., Gutierrez J. C., & Leon-Becerra J. (2024). Integration of machine learning in the supply chain for decision making: A systematic literature review. *Journal of Industrial Engineering and Management*, 17(2), 344. 10.3926/jiem.6403

R P., Kumar S., Bhardwaj S., Agrahari N., Pandey S., & Harakannanavar S. S. (2023). E-Commerce Inventory Management System Using Machine Learning Approach. *2023 International Conference on Data Science and Network Security (ICDSNS)*, Volume(Issue), 1-7. 10.1109/icdsns58469.2023.10245500

Sathish T., SaiKumar D., Patil S., Saravanan R., Giri J., & Aly A. A. (2024). Exponential smoothing method against the gradient boosting machine learning algorithm-based model for materials forecasting to minimize inventory. *AIP Advances*, 14(6), Page. 10.1063/5.0208491

Appendix I

R Code

```
# ------------------------------
# 1. Load Libraries
# ------------------------------

knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
library(tidyverse)
library(lubridate)
library(forecast)
library(tseries)
library(vars)
library(gridExtra)
library(kableExtra)
library(tsibble)
library(fable)
library(DT)

# ------------------------------
# 2. Load and Prepare Data
# ------------------------------
library(readr)
raw <- read_csv("products_02.csv",
    col_types = cols(stock_xfer_date = col_date(format = "%m/%d/%Y")))

head(raw,5)


# ------------------------------
# 3 Clean and split dimensions (remove " cm")
# ------------------------------

raw <- raw %>%
  mutate(
    stock_xfer_date = ymd(stock_xfer_date),
```

```
    prod_dimensions = str_remove(prod_dimensions, " cm")
  ) %>%
  separate(prod_dimensions, into = c("len_cm", "width_cm", "height_cm"), sep = "x", convert =

# Compute volume per item in cm3, convert to m3, compute space used
raw <- raw %>%
  mutate(
    volume_cm3 = len_cm * width_cm * height_cm,
    space_used_cm3 = volume_cm3 * stock_qty,
    space_used_m3 = space_used_cm3 / 1e6
  )

# ------------------------------
# 3. Create Daily Aggregated Time Series
# ------------------------------

daily_totals <- raw %>%
  group_by(stock_xfer_date) %>%
  summarise(
    total_stock_qty = sum(stock_qty),
    total_space_m3 = sum(space_used_m3)
  ) %>%
  arrange(stock_xfer_date)

# Create ts object daily frequency assumed to be 365
ts_total_space <- ts(daily_totals$total_space_m3, frequency = 7)    # weekly seasonality
ts_total_stock <- ts(daily_totals$total_stock_qty, frequency = 7)


ggplot(daily_totals, aes(stock_xfer_date, total_space_m3)) +
geom_line(color = "steelblue") +
labs(title="Total Space Used (m³) Over Time",
x="Date", y="Space Used (m³)") +
theme_minimal()

# ------------------------------
# 4. Create Category-Level Time Series
# ------------------------------

cat_daily <- raw %>%
  group_by(stock_xfer_date, prod_cat) %>%
  summarise(space_m3 = sum(space_used_m3), stock_qty = sum(stock_qty)) %>%
  ungroup()
```

```r
cat_space_wide <- cat_daily %>%
  dplyr::select(stock_xfer_date, prod_cat, space_m3) %>%
  tidyr::pivot_wider(names_from = prod_cat, values_from = space_m3) %>%
  dplyr::arrange(stock_xfer_date)


# Replace NA (missing categories on some days)
cat_space_wide[is.na(cat_space_wide)] <- 0

cat("Category-wide dataset created:\n")
print(head(cat_space_wide))
print(dim(cat_space_wide))

# Convert to time series matrix
ts_cat_space <- ts(cat_space_wide[, -1], frequency = 7)

cat("ts_cat_space created successfully:\n")
print(dim(ts_cat_space))
print(colnames(ts_cat_space))



ggplot(cat_daily, aes(stock_xfer_date, space_m3, color=prod_cat)) +
geom_line() +
theme_minimal() +
labs(title="Daily Space Used by Category (m³)", x="Date", y="Space (m³)")

# --------------------------------
# 5. Stationarity Tests (ADF)
# --------------------------------

adf_total <- adf.test(ts_total_space)
adf_total

# First difference
adf_total_diff <- adf.test(diff(ts_total_space))
adf_total_diff

# Category ADF tests
apply(ts_cat_space, 2, adf.test)

adf_total_diff <- adf.test(diff(ts_total_space))
```

```r
# ------------------------------
# 6. Fit SARIMA Model for Total Space
# ------------------------------

sarima_model <- Arima(ts_total_space, order = c(0,1,2),
                      seasonal = list(order = c(2,0,2), period = 7))

summary(sarima_model)



p1 <- ggAcf(residuals(sarima_model)) + ggtitle("ACF of SARIMA Residuals")
p2 <- ggPacf(residuals(sarima_model)) + ggtitle("PACF of SARIMA Residuals")
grid.arrange(p1, p2, ncol=2)

# Forecast next 60 days
sarima_fc <- forecast(sarima_model, h=60)

autoplot(sarima_fc) +
ggtitle("SARIMA Forecast - Total Space Used (m³)") +
theme_minimal()


# ------------------------------
# 7. Train/Test Accuracy Evaluation
# ------------------------------

n <- length(ts_total_space)

# At least 60 points
h <- min(60, floor(n * 0.2))    # 20% test set if data is shorter

cat("Series length:", n, "\n")
cat("Forecast horizon:", h, "\n")

# Safe slicing using vector positions
train <- ts_total_space[1:(n - h)]
test  <- ts_total_space[(n - h + 1):n]

sarima_train_model <- Arima(
  train,
  order = c(0,1,2),
  seasonal = list(order = c(2,0,2), period = 7)
```

```
)

pred <- forecast(sarima_train_model, h = h)

cat("SARIMA Accuracy (Safe Train/Test Split):\n")
print(accuracy(pred, test))



#Box.test(sarima_resid, lag = 10, type = "Ljung-Box")


# ------------------------------
# 8. Fit VAR Model for Category-Level Space Usage
# ------------------------------

# First difference for stationarity
ts_cat_space_diff <- diff(ts_cat_space)

# Select optimal lag (VAR order)
lag_selection <- VARselect(ts_cat_space_diff, lag.max = 15, type = "const")
lag_selection

# Fit VAR(2)
var_model <- VAR(ts_cat_space_diff, p = 2, type = "const")
var_model


# ------------------------------
# Correction for VAR Forecasting
# ------------------------------

cat_matrix <- data.frame(cat_space_wide[, -1])

for(i in seq_along(cat_matrix)){
  cat_matrix[[i]] <- as.numeric(cat_matrix[[i]])
}

cat_matrix_diff <- diff(as.matrix(cat_matrix))
colnames(cat_matrix_diff) <- colnames(cat_matrix)
cat_df_diff <- data.frame(cat_matrix_diff)

library(vars)
var_model <- vars::VAR(cat_df_diff, p = 2, type = "const")
```

```r
class(var_model)

var_raw <- stats::predict(var_model, n.ahead = 30)

fcst_means <- lapply(var_raw$fcst, function(x) as.numeric(x[,1]))

var_fc_diff <- do.call(cbind, fcst_means)
colnames(var_fc_diff) <- names(var_raw$fcst)

last_levels <- as.numeric(tail(cat_matrix, 1))

var_fc_levels <- sweep(apply(var_fc_diff, 2, cumsum), 2, last_levels, "+")
var_fc_levels <- as.data.frame(var_fc_levels)

# ------------------------------
# 9. VAR Forecasting (Category Space)
# ------------------------------

var_forecast <- predict(var_model, n.ahead = 30)

# Rebuild into levels
initial_values <- tail(cat_space_wide[, -1], 1)

var_fc_levels <- sapply(var_forecast$fcst, function(x) {
  cumsum(x[, 1]) + as.numeric(initial_values)
})

var_fc_levels <- as.data.frame(var_fc_levels)

cat("Category-level VAR forecast (levels):\n")
print(var_fc_levels[1:5, ])


var_fc <- predict(var_model, n.ahead = 30)

initial_values <- tail(cat_space_wide[, -1], 1)

var_fc_levels <- sapply(var_fc$fcst, function(x) {
cumsum(x[,1]) + as.numeric(initial_values)
})

var_fc_levels <- as.data.frame(var_fc_levels)
head(var_fc_levels)
```

```r
matplot(var_fc_levels, type="l", lwd=2, lty=1,
col=c("purple","blue","green","red"),
ylab="Space (m³)", xlab="Day Ahead",
main="Category-Level VAR Forecast (Space Used)")
legend("topleft", legend=colnames(var_fc_levels),
col=c("purple","blue","green","red"), lty=1, lwd=2)

# ------------------------------
# 10. Hierarchical Reconciliation
# ------------------------------

# Scale category forecasts to match SARIMA total forecast
cat_matrix <- cat_space_wide[, -1]              # remove date
cat_matrix <- data.frame(cat_matrix)            # remove tibble class

# Force everything numeric
for(i in seq_along(cat_matrix)){
  cat_matrix[[i]] <- as.numeric(cat_matrix[[i]])
}

# Check structure
str(cat_matrix)

# Differencing
cat_matrix_diff <- diff(as.matrix(cat_matrix))  # returns pure numeric matrix
colnames(cat_matrix_diff) <- colnames(cat_matrix)

# Convert to dataframe for VAR
cat_df_diff <- as.data.frame(cat_matrix_diff)

# Final check before VAR
cat("CLEAN cat_df_diff:\n")
str(cat_df_diff)

# Number of forecast steps in VAR output
h_var <- nrow(var_fc_levels)

# Take the first h_var SARIMA forecasts and coerce to numeric
sarima_vals <- as.numeric(sarima_fc$mean[seq_len(h_var)])

# Sum of category forecasts per horizon
cat_sums <- rowSums(var_fc_levels)
```

```r
# Scaling factor so that row sums match SARIMA totals
scale_factor <- sarima_vals / cat_sums

# Apply scaling to each row (each horizon)
reconciled_cat <- sweep(var_fc_levels, 1, scale_factor, "*")

# Inspect first few reconciled rows
head(reconciled_cat) %>% kable() %>% kable_styling()

sarima_vals <- as.numeric(sarima_fc$mean[seq_len(h_var)])

#`````````````````````````````````````````````````````````
# SUMMARY STATISTICS FOR INVENTORY & SPACE DATA
#`````````````````````````````````````````````````````````

library(dplyr)
library(tidyr)

# ---------------------------------------------------------
# 1. Summary of the raw dataset
# ---------------------------------------------------------
cat("==== RAW DATA SUMMARY ====\n")
print(summary(raw))

cat("\n==== RAW DATA STRUCTURE ====\n")
str(raw)

# ---------------------------------------------------------
# 2. Summary of per-unit volume and space
# ---------------------------------------------------------
#cat("\n==== VOLUME & SPACE SUMMARY (PER SKU TRANSFER) ====\n")
#raw %>%
#  select(volume_cm3, space_used_cm3, space_used_m3, stock_qty) %>%
# summary() %>%
#  print()

# ---------------------------------------------------------
# 3. Summary statistics for daily totals
# ---------------------------------------------------------
daily_summary <- daily_totals %>%
  summarise(
    mean_stock = mean(total_stock_qty),
    sd_stock = sd(total_stock_qty),
```

```
    min_stock = min(total_stock_qty),
    max_stock = max(total_stock_qty),
    median_stock = median(total_stock_qty),
    mean_space = mean(total_space_m3),
    sd_space = sd(total_space_m3),
    min_space = min(total_space_m3),
    max_space = max(total_space_m3),
    median_space = median(total_space_m3)
  )

cat("\n==== DAILY TOTAL SUMMARY ====\n")
print(daily_summary)

# ------------------------------------------------------------
# 4. Summary by product category (stock + space)
# ------------------------------------------------------------
cat("\n==== CATEGORY-LEVEL SUMMARY ====\n")

category_summary <- raw %>%
  group_by(prod_cat) %>%
  summarise(
    total_stock = sum(stock_qty),
    avg_stock = mean(stock_qty),
    sd_stock = sd(stock_qty),
    total_space_m3 = sum(space_used_m3),
    avg_space_m3 = mean(space_used_m3),
    sd_space_m3 = sd(space_used_m3),
    avg_volume_cm3 = mean(volume_cm3),
    sd_volume_cm3 = sd(volume_cm3)
  )

print(category_summary)

# ------------------------------------------------------------
# 5. Summary of daily category-level space
# ------------------------------------------------------------
cat("\n==== DAILY CATEGORY SPACE SUMMARY ====\n")

daily_cat_summary <- cat_space_wide %>%
  summarise(
    across(-stock_xfer_date,
           list(
             mean = mean,
```

```
            sd = sd,
            min = min,
            max = max,
            median = median
          ),
          .names = "{.col}_{.fn}"
    )
  )

print(daily_cat_summary)

# ------------------------------------------------------------
# 6. Correlation analysis
# ------------------------------------------------------------
cat("\n==== CORRELATIONS ====\n")

cor_total <- cor(daily_totals$total_stock_qty, daily_totals$total_space_m3)
cat("Correlation (Total Stock vs Total Space):", cor_total, "\n")

cor_categories <- cor(cat_space_wide[, -1])
cat("\nCorrelation Matrix (Category Space m³):\n")
print(round(cor_categories, 3))

#``````````````````````````````````````````````````````````
# QQ PLOTS FOR INVENTORY & SPACE TIME SERIES ANALYSIS
#``````````````````````````````````````````````````````````

library(ggplot2)

# ------------------------------------------------------------
# 1. QQ Plot for SARIMA residuals (Total Space)
# ------------------------------------------------------------
sarima_resid <- residuals(sarima_model)

# Base R QQ plot
qqnorm(sarima_resid, main = "QQ Plot of SARIMA Residuals (Total Space)")
qqline(sarima_resid, col = "red", lwd = 2)

# ggplot version (optional)
sarima_df <- data.frame(resid = sarima_resid)

ggplot(sarima_df, aes(sample = resid)) +
  stat_qq() +
```

```
  stat_qq_line(col = "red", size = 1) +
  ggtitle("QQ Plot - SARIMA Residuals (Total Space)") +
  theme_minimal()

# ------------------------------------------------------------
# 2. QQ Plot for Total Space Series (Raw or Differenced)
# ------------------------------------------------------------
# Raw series
qqnorm(ts_total_space, main = "QQ Plot - Total Space (Raw Data)")
qqline(ts_total_space, col = "red", lwd = 2)

# Differenced series
qqnorm(diff(ts_total_space), main = "QQ Plot - Differenced Total Space")
qqline(diff(ts_total_space), col = "blue", lwd = 2)

# ------------------------------------------------------------
# 3. QQ Plots for Category-Level Space (Each Category)
# ------------------------------------------------------------
cat_names <- colnames(ts_cat_space)

par(mfrow = c(2, 2))  # 4 plots together

for (i in 1:ncol(ts_cat_space)) {
  qqnorm(ts_cat_space[, i],
         main = paste("QQ Plot -", cat_names[i]),
         cex = 0.8)
  qqline(ts_cat_space[, i], col = "red", lwd = 2)
}

par(mfrow = c(1, 1))  # reset layout

# ------------------------------------------------------------
# 4. QQ Plots for VAR Residuals (Category Model)
# ------------------------------------------------------------
var_resid <- resid(var_model)

cat("VAR residuals structure:\n")
print(str(var_resid))

# var_resid is a list with one matrix per equation - we loop through them
par(mfrow = c(2, 2))

# for (i in 1:length(var_resid)) {
```

```
#   v <- var_resid[[i]][, 1]  # extract residuals for equation i
#
#   qqnorm(v,
#          main = paste("QQ Plot - VAR Residuals:", names(var_resid)[i]),
#          cex = 0.8)
#   qqline(v, col = "red", lwd = 2)
# }
#
# par(mfrow = c(1, 1))

#``````````````````````````````````````````````````````````
# TOTAL SPACE USAGE RESULTS (m³)
#``````````````````````````````````````````````````````````

library(dplyr)
library(ggplot2)
library(lubridate)
library(forecast)

# ----------------------------------------------------------
# 1. Compute Daily Total Space Usage
# ----------------------------------------------------------

daily_totals <- raw %>%
  group_by(stock_xfer_date) %>%
  summarise(
    total_space_m3 = sum(space_used_m3),
    total_stock_qty = sum(stock_qty),
    .groups = "drop"
  ) %>%
  arrange(stock_xfer_date)

head(daily_totals)

# ----------------------------------------------------------
# 2. Summary Statistics for Total Space
# ----------------------------------------------------------

total_space_summary <- daily_totals %>%
  summarise(
    mean_space = mean(total_space_m3),
    sd_space = sd(total_space_m3),
    min_space = min(total_space_m3),
```

```
    max_space = max(total_space_m3),
    median_space = median(total_space_m3),
    q1_space = quantile(total_space_m3, 0.25),
    q3_space = quantile(total_space_m3, 0.75),
    iqr_space = IQR(total_space_m3)
  )

cat("==== Total Space Usage Summary (m³/day) ====\n")
print(total_space_summary)

# ------------------------------------------------------------
# 3. Summary Statistics for Total Stock Transfers
# ------------------------------------------------------------

total_stock_summary <- daily_totals %>%
  summarise(
    mean_stock = mean(total_stock_qty),
    sd_stock = sd(total_stock_qty),
    min_stock = min(total_stock_qty),
    max_stock = max(total_stock_qty),
    median_stock = median(total_stock_qty),
    q1_stock = quantile(total_stock_qty, 0.25),
    q3_stock = quantile(total_stock_qty, 0.75),
    iqr_stock = IQR(total_stock_qty)
  )

cat("\n==== Total Stock Transfer Summary (units/day) ====\n")
print(total_stock_summary)

# ------------------------------------------------------------
# 4. Correlation Between Stock & Space
# ------------------------------------------------------------

cor_stock_space <- cor(daily_totals$total_stock_qty,
                       daily_totals$total_space_m3)

cat("\nCorrelation between Stock Transfers and Space Used:",
    round(cor_stock_space, 4), "\n")

# ------------------------------------------------------------
# 5. Plot: Total Daily Space Usage (m³)
# ------------------------------------------------------------
```

```r
ggplot(daily_totals, aes(x = stock_xfer_date, y = total_space_m3)) +
  geom_line(color = "steelblue", linewidth = 1) +
  labs(
    title = "Daily Total Space Usage (m³) - 2023",
    x = "Date",
    y = "Space Used (m³)"
  ) +
  theme_minimal()

# ---------------------------------------------------------
# 6. Histogram + Density of Total Space Usage
# ---------------------------------------------------------

ggplot(daily_totals, aes(x = total_space_m3)) +
  geom_histogram(binwidth = 0.3, fill = "skyblue", color = "black") +
  geom_density(color = "red", linewidth = 1) +
  labs(
    title = "Distribution of Daily Space Usage",
    x = "Space Used (m³)",
    y = "Frequency"
  ) +
  theme_minimal()

# ---------------------------------------------------------
# 7. Prepare Time-Series Object for Modeling
# ---------------------------------------------------------

ts_total_space <- ts(daily_totals$total_space_m3, frequency = 7)

cat("\n==== Time Series Object for Total Space ====\n")
print(ts_total_space)


#``````````````````````````````
# STOCK VS. SPACE CORRELATION
#``````````````````````````````

library(dplyr)
library(ggplot2)

# ---------------------------------------------------------
# 1. Calculate correlation coefficient
# ---------------------------------------------------------
```

```r
cor_stock_space <- cor(daily_totals$total_stock_qty,
                       daily_totals$total_space_m3)

cat("Correlation between Stock Transfers and Space Used (m³):",
    round(cor_stock_space, 4), "\n")

# ---------------------------------------------------------
# 2. Perform a correlation test (p-value, CI)
# ---------------------------------------------------------

cor_test_result <- cor.test(daily_totals$total_stock_qty,
                            daily_totals$total_space_m3)

cat("\n==== Correlation Test Results ====\n")
print(cor_test_result)

# ---------------------------------------------------------
# 3. Scatterplot of Stock vs Space
# ---------------------------------------------------------

ggplot(daily_totals, aes(x = total_stock_qty, y = total_space_m3)) +
  geom_point(color = "steelblue", alpha = 0.6) +
  geom_smooth(method = "lm", se = TRUE, color = "red", linewidth = 1) +
  labs(
    title = "Relationship Between Stock Transfers and Space Used",
    subtitle = paste("Correlation =", round(cor_stock_space, 3)),
    x = "Total Stock Transfers (Units)",
    y = "Total Space Used (m³)"
  ) +
  theme_minimal()

# ---------------------------------------------------------
# 4. Display scatterplot with log scaling
# ---------------------------------------------------------

ggplot(daily_totals, aes(x = total_stock_qty, y = total_space_m3)) +
  geom_point(color = "darkgreen", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red") +
  scale_x_continuous(trans = "log10") +
  labs(
    title = "Stock vs Space (Log-Scaled Stock)",
    x = "Total Stock Transfers (log10)",
    y = "Total Space Used (m³)"
```

```
  ) +
  theme_minimal()

#````````````````````````````````````````
# CATEGORY-LEVEL RESULTS (SPACE + STOCK)
#````````````````````````````````````````

library(dplyr)
library(tidyr)
library(ggplot2)

# ---------------------------------------------------------
# 1. Create daily category-level dataset
# ---------------------------------------------------------

cat_daily <- raw %>%
  group_by(stock_xfer_date, prod_cat) %>%
  summarise(
    space_m3 = sum(space_used_m3),
    stock_qty = sum(stock_qty),
    .groups = "drop"
  ) %>%
  arrange(stock_xfer_date)

head(cat_daily)

# ----------------------------------
# 2. Summary statistics per category
# ----------------------------------

cat_summary <- cat_daily %>%
  group_by(prod_cat) %>%
  summarise(
    mean_space = mean(space_m3),
    sd_space = sd(space_m3),
    min_space = min(space_m3),
    max_space = max(space_m3),
    median_space = median(space_m3),

    mean_stock = mean(stock_qty),
    sd_stock = sd(stock_qty),
    min_stock = min(stock_qty),
    max_stock = max(stock_qty),
```

```
    median_stock = median(stock_qty)
  )

cat("\n==== CATEGORY-LEVEL SUMMARY ====\n")
print(cat_summary)

# ---------------------------------
# 3. Category Share of Total Space
# ---------------------------------

cat_share <- cat_daily %>%
  group_by(prod_cat) %>%
  summarise(total_space = sum(space_m3)) %>%
  mutate(share = total_space / sum(total_space))

cat("\n==== CATEGORY SHARE OF TOTAL SPACE ====\n")
print(cat_share)

# ---------------------------------------------------------
# 4. Create wide category space matrix for correlation + VAR
# ---------------------------------------------------------

# cat_space_wide <- cat_daily %>%
#   select(stock_xfer_date, prod_cat, space_m3) %>%
#   pivot_wider(names_from = prod_cat,
#               values_from = space_m3,
#               values_fill = 0) %>%
#   arrange(stock_xfer_date)
#
# cat("\n==== CATEGORY SPACE (WIDE FORMAT) ====\n")
# print(head(cat_space_wide))

# ---------------------------------------------------------
# 5. Correlation Across Categories (space)
# ---------------------------------------------------------

cor_cat_space <- cor(cat_space_wide[, -1])
cat("\n==== CATEGORY SPACE CORRELATION MATRIX ====\n")
print(round(cor_cat_space, 3))

# ---------------------------------------------------------
# 6. Plot: Daily Category Space Usage (m^3)
# ---------------------------------------------------------
```

```r
ggplot(cat_daily, aes(x = stock_xfer_date, y = space_m3, color = prod_cat)) +
  geom_line(linewidth = 0.8) +
  labs(
    title = "Daily Space Usage by Product Category",
    x = "Date",
    y = "Space Used (m³)",
    color = "Product Category"
  ) +
  theme_minimal()

# ------------------------------------------------------------
# 7. Plot: Category Space Distribution
# ------------------------------------------------------------

ggplot(cat_daily, aes(x = prod_cat, y = space_m3, fill = prod_cat)) +
  geom_boxplot(alpha = 0.8) +
  labs(
    title = "Distribution of Daily Space Usage by Category",
    x = "Category",
    y = "Space (m³)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")

# ------------------------------------------------------------
# 8. Plot: Daily Category Stock Transfers
# ------------------------------------------------------------

ggplot(cat_daily, aes(x = stock_xfer_date, y = stock_qty, color = prod_cat)) +
  geom_line(linewidth = 0.8) +
  labs(
    title = "Daily Stock Transfers by Product Category",
    x = "Date",
    y = "Units Transferred"
  ) +
  theme_minimal()

# ------------------------------------------------------------
# 9. Category-level summary table for the report
# ------------------------------------------------------------

category_summary_table <- cat_summary %>%
  mutate(
```

```
    avg_daily_share = cat_share$share[match(prod_cat, cat_share$prod_cat)]
  )

cat("\n==== FINAL CATEGORY SUMMARY (SPACE + STOCK + SHARE) ====\n")
print(category_summary_table)

#````````````````````````````````````````````````````````
# STATIONARITY RESULTS FOR TOTAL SPACE + CATEGORY SERIES
#````````````````````````````````````````````````````````

library(tseries)
library(forecast)
library(ggplot2)

# ---------------------------------------------------------
# 1. ADF Test for Total Space Used (m³)
# ---------------------------------------------------------

cat("==== ADF Test: Total Space (Level) ====\n")
adf_level <- adf.test(ts_total_space)
print(adf_level)

cat("\n==== ADF Test: Total Space (First Difference) ====\n")
adf_diff <- adf.test(diff(ts_total_space))
print(adf_diff)

# ---------------------------------------------------------
# 2. ACF & PACF Plots for Total Space Used
# ---------------------------------------------------------

par(mfrow = c(2, 1))

acf(ts_total_space, main = "ACF - Total Space (Level)")
pacf(ts_total_space, main = "PACF - Total Space (Level)")

par(mfrow = c(2, 1))

acf(diff(ts_total_space), main = "ACF - Total Space (First Difference)")
pacf(diff(ts_total_space), main = "PACF - Total Space (First Difference)")

par(mfrow = c(1, 1))

# ---------------------------------------------------------
# 3. ADF Tests for Category-Level Series
```

```r
# ----------------------------------------------------------

cat("\n==== ADF Tests: Category Space (Level Series) ====\n")

cat_names <- colnames(ts_cat_space)

for (i in 1:ncol(ts_cat_space)) {
  cat("\nCategory:", cat_names[i], "\n")
  print(adf.test(ts_cat_space[, i]))
}

cat("\n==== ADF Tests: Category Space (First Differences) ====\n")

for (i in 1:ncol(ts_cat_space)) {
  cat("\nCategory:", cat_names[i], " (Differenced)\n")
  print(adf.test(diff(ts_cat_space[, i])))
}

# ----------------------------------------------------------
# 4. ACF & PACF Plots for Category-Level Series
# ----------------------------------------------------------

par(mfrow = c(2, 2))

for (i in 1:ncol(ts_cat_space)) {
  acf(ts_cat_space[, i],
      main = paste("ACF -", cat_names[i], "(Level)"))
}

par(mfrow = c(2, 2))

for (i in 1:ncol(ts_cat_space)) {
  acf(diff(ts_cat_space[, i]),
      main = paste("ACF -", cat_names[i], "(First Difference)"))
}

par(mfrow = c(2, 2))

for (i in 1:ncol(ts_cat_space)) {
  pacf(ts_cat_space[, i],
       main = paste("PACF -", cat_names[i], "(Level)"))
}
```

```r
par(mfrow = c(2, 2))

for (i in 1:ncol(ts_cat_space)) {
  pacf(diff(ts_cat_space[, i]),
       main = paste("PACF -", cat_names[i], "(First Difference)"))
}

par(mfrow = c(1, 1))


acf(ts_cat_space)

#````````````````````````````````````````````````````````````````
#   FORECASTED RESULTS FOR VAR MODEL
#````````````````````````````````````````````````````````````````

library(vars)
library(tidyverse)

# --------------------------------
# 1. Forecast 30 days ahead
# --------------------------------
var_fc <- predict(var_model, n.ahead = 30)

# View raw forecast structure
print(var_fc)

# --------------------------------
# 2. Extract forecasted changes (differenced predictions)
# --------------------------------
# var_fc$fcst is a list with one entry per category
# Each contains mean, lower, upper forecasts

fc_means <- sapply(var_fc$fcst, function(x) x[, "fcst"])
colnames(fc_means) <- names(var_fc$fcst)

cat("=== Forecasted FIRST DIFFERENCES (VAR) ===\n")
print(round(fc_means, 4))

# --------------------------------
# 3. Convert differenced forecasts back to actual LEVELS
# --------------------------------
# Last observed category space values from 2023
last_obs <- tail(cat_space_wide[, -1], 1)
```

```r
# Cumulative sum of differenced forecasts + last observed values
fc_levels <- apply(fc_means, 2, cumsum) %>%
  sweep(1, as.numeric(last_obs), `+`)

fc_levels <- as.data.frame(fc_levels)
fc_levels$Day <- 1:nrow(fc_levels)

cat("\n=== Forecasted CATEGORY SPACE LEVELS (m³) ===\n")
print(round(fc_levels, 4))

# ------------------------------
# 4. Optional: plot VAR forecasts
# ------------------------------
fc_long <- fc_levels %>%
  pivot_longer(cols = -Day, names_to = "Category", values_to = "Forecast_m3")

ggplot(fc_long, aes(x = Day, y = Forecast_m3, color = Category)) +
  geom_line(size = 1.2) +
  theme_minimal() +
  ggtitle("VAR Forecast - Category-Level Space Usage (m³)") +
  labs(x = "Forecast Day", y = "Space Used (m³)") +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    legend.title = element_blank()
  )

#````````````````````````````````````````
# COMPLETE EXPLORATORY DATA ANALYSIS (EDA)
# Dataset: products_02.csv
#````````````````````````````````````````

library(tidyverse)
library(lubridate)
library(psych)
library(GGally)
library(gridExtra)

# ---------------------------------------------------------
# 1. Load Data & Initial Inspection
# ---------------------------------------------------------
raw <- read_csv("products_02.csv")
```

```r
cat("==== HEAD OF DATA ====\n")
print(head(raw))

cat("\n==== STRUCTURE OF DATA ====\n")
str(raw)

cat("\n==== SUMMARY OF RAW DATA ====\n")
summary(raw)

cat("\n==== CHECK FOR MISSING VALUES ====\n")
print(colSums(is.na(raw)))

# ------------------------------------------------------------
# 2. Clean & Prepare Variables
# ------------------------------------------------------------
raw <- raw %>%
  mutate(
    # stock_xfer_date = ymd(stock_xfer_date),
    prod_dimensions = str_remove(prod_dimensions, " cm")
  ) %>%
  separate(prod_dimensions, into = c("len_cm", "width_cm", "height_cm"),
           sep = "x", convert = TRUE) %>%
  mutate(
    volume_cm3 = len_cm * width_cm * height_cm,
    space_used_cm3 = volume_cm3 * stock_qty,
    space_used_m3 = space_used_cm3 / 1e6
  )

cat("\n==== AFTER CLEANING ====\n")
summary(raw)

# ------------------------------------------------------------
# 3. Numerical Summary Statistics
# ------------------------------------------------------------
num_vars <- raw %>%
  select(stock_qty, len_cm, width_cm, height_cm, volume_cm3, space_used_m3)

cat("\n==== SUMMARY STATISTICS (NUMERIC VARIABLES) ====\n")
print(describe(num_vars))

# ------------------------------------------------------------
# 4. Category-Level Summaries
# ------------------------------------------------------------
```

```r
cat("\n==== CATEGORY SUMMARY ====\n")
cat_summary <- raw %>%
  group_by(prod_cat) %>%
  summarise(
    n = n(),
    total_stock = sum(stock_qty),
    avg_stock = mean(stock_qty),
    total_space_m3 = sum(space_used_m3),
    avg_space_m3 = mean(space_used_m3),
    avg_volume_cm3 = mean(volume_cm3)
  )

print(cat_summary)

num_vars <- raw %>%
  select(stock_qty, len_cm, width_cm, height_cm, volume_cm3, space_used_m3)

describe(num_vars)

# ------------------------------------------------------------
# 5. Visual Distributions - Stock Transfers
# ------------------------------------------------------------
p1 <- ggplot(raw, aes(stock_qty)) +
  geom_histogram(fill="steelblue", color="black", bins=40) +
  theme_minimal() + ggtitle("Histogram: Stock Transfers")

p2 <- ggplot(raw, aes(y = stock_qty, x = prod_cat, fill = prod_cat)) +
  geom_boxplot() +
  theme_minimal() + ggtitle("Stock Transfers by Product Category")

grid.arrange(p1, p2, ncol = 2)

# ------------------------------------------------------------
# 6. Visual Distributions - Space Used
# ------------------------------------------------------------
p3 <- ggplot(raw, aes(space_used_m3)) +
  geom_histogram(fill="darkgreen", color="black", bins=40) +
  theme_minimal() + ggtitle("Histogram: Space Used (m3)")

p4 <- ggplot(raw, aes(x = prod_cat, y = space_used_m3, fill = prod_cat)) +
  geom_boxplot() +
  theme_minimal() + ggtitle("Space Used by Category (m3)")

grid.arrange(p3, p4, ncol = 2)
```

```r
# ------------------------------------------------------------
# 7. Time Series Aggregation
# ------------------------------------------------------------
daily <- raw %>%
  group_by(stock_xfer_date) %>%
  summarise(
    total_stock = sum(stock_qty),
    total_space_m3 = sum(space_used_m3)
  )

cat("\n==== DAILY SUMMARY ====\n")
print(head(daily))

# Time series plots
p5 <- ggplot(daily, aes(stock_xfer_date, total_stock)) +
  geom_line(color="steelblue") +
  theme_minimal() + ggtitle("Daily Total Stock Transfers")

p6 <- ggplot(daily, aes(stock_xfer_date, total_space_m3)) +
  geom_line(color="darkgreen") +
  theme_minimal() + ggtitle("Daily Total Space Used (m3)")

grid.arrange(p5, p6, ncol = 1)

# ------------------------------------------------------------
# 8. Category-Level Time Series
# ------------------------------------------------------------
cat_daily <- raw %>%
  group_by(stock_xfer_date, prod_cat) %>%
  summarise(space_m3 = sum(space_used_m3), stock_qty = sum(stock_qty),
            .groups = "drop")

p7 <- ggplot(cat_daily, aes(stock_xfer_date, space_m3, color = prod_cat)) +
  geom_line() +
  theme_minimal() +
  ggtitle("Daily Space Used by Category")

p8 <- ggplot(cat_daily, aes(stock_xfer_date, stock_qty, color = prod_cat)) +
  geom_line() +
  theme_minimal() +
  ggtitle("Daily Stock Transfers by Category")

grid.arrange(p7, p8, ncol = 1)
```

```r
# ------------------------------------------------------------
# 9. Correlation Analysis
# ------------------------------------------------------------
cor_vars <- raw %>%
  select(stock_qty, len_cm, width_cm, height_cm, volume_cm3, space_used_m3)

cat("\n==== CORRELATION MATRIX ====\n")
print(cor(cor_vars))

ggpairs(cor_vars) + ggtitle("Pairwise Relationships")

# ------------------------------------------------------------
# 10. Distribution of Item Dimensions & Volume
# ------------------------------------------------------------
p9 <- ggplot(raw, aes(volume_cm3)) +
  geom_histogram(fill="purple", color="black", bins=40) +
  theme_minimal() + ggtitle("Histogram: Item Volume (cm3)")

p10 <- ggplot(raw, aes(x = prod_cat, y = volume_cm3, fill = prod_cat)) +
  geom_boxplot() +
  theme_minimal() +
  ggtitle("Volume Distribution by Product Category")

grid.arrange(p9, p10, ncol = 2)

# ------------------------------------------------------------
# 11. Scatterplots (Stock vs Space, Volume vs Space)
# ------------------------------------------------------------
p11 <- ggplot(raw, aes(stock_qty, space_used_m3)) +
  geom_point(alpha=0.3, color="blue") +
  theme_minimal() +
  ggtitle("Stock Transfers vs Space Used (m3)")

p12 <- ggplot(raw, aes(volume_cm3, space_used_m3)) +
  geom_point(alpha=0.3, color="red") +
  theme_minimal() +
  ggtitle("Volume (cm3) vs Space Used (m3)")

grid.arrange(p11, p12, ncol = 2)
```