APS 4742
STATISTICAL APPLICATIONS IN INDUSTRY AND PROJECT PRESENTATION
GROUP NO: 4

RETAIL DEMAND ANALYSIS

(A six sigma DMAIC Approach to analyzing and reducing Stockouts in retail)

Table of Contents

1.Introduction

In the fast-moving world of retail, one of the most frustrating experiences—for both customers and store managers is when a product is out of stock. Whether it's a popular item during the holiday season or an essential good during a promotion, stockouts lead to missed sales, disappointed customers, and lost trust.

The key problem we're addressing in this study is frequent and avoidable stockouts in retail stores. These stockouts often happen not because the product is unavailable in the supply chain, but due to ineffective demand forecasting, poor inventory planning, or slow response to external changes like weather, promotions, or unexpected events.

We're trying to solve this problem because the cost of doing nothing is high. When a product isn't available, a customer might not just leave the store—they might not come back at all. For the business, this results in lost revenue, reduced market competitiveness, and operational inefficiencies. Worse still, repeated stockouts can signal deeper issues within the inventory and supply chain systems that hinder long-term growth.

This report explores a structured, analytical approach to understanding and addressing these inefficiencies. Using the DMAIC (Define, Measure, Analyze, Improve, Control) framework from Six Sigma, we aim to uncover the underlying causes of stockouts and offer practical recommendations that can help improve inventory accuracy, demand planning, and overall retail performance.

2.Litreture Review

Retail demand analysis is a critical function in modern inventory and supply chain management. It enables retailers to anticipate customer demand, minimize stockouts, manage seasonal variations, and implement proactive replenishment strategies. With the growing availability of large datasets and advanced analytics tools such as Python, retail organizations now utilize data-driven approaches to enhance demand forecasting and improve operational efficiency. This literature review presents key findings from previous studies relevant to the analysis conducted in this project, with a focus on five thematic areas: retail analytics and forecasting, inventory management, the DMAIC framework, promotions and seasonality, and technology-based improvements.

<u>Retail Analytics and Demand Forecasting</u>

Retail demand forecasting has evolved from basic historical sales trend analysis to the incorporation of external influencing factors such as weather, competitor activity, and promotional events. Chopra and Meindl (2019) emphasized that effective forecasting models combine internal and external variables to better predict consumer behavior. Fildes, Goodwin, Lawrence, and Nikolopoulos (2008) further argued that judgmental adjustments and contextual variables improve forecast accuracy, especially in retail environments with fluctuating demand.

In this study, a simulated dataset from Kaggle was used, which spans over two years and includes multiple product categories (electronics, clothing, groceries, toys, and furniture), as well as external variables such as weather and promotions. The dataset's richness reflects the growing trend in retail analytics to build comprehensive forecasting models using multi-dimensional data.

Inventory Management and Stockouts

Inventory control is vital for ensuring product availability and reducing holding costs. One of the most persistent challenges in retail is managing stockouts, which can lead to customer dissatisfaction and lost sales. Silver, Pyke, and Thomas (2016) highlighted that key contributors to stockouts include understocking, poor inventory flow, absence of reorder triggers, and neglect of seasonal demand fluctuations.

In line with their findings, this project applied inventory performance metrics such as the inventory turnover ratio and used visualization tools like Pareto charts and fishbone diagrams to identify the root causes of stockouts. These tools helped uncover inefficiencies such as promotion mismatches, poor reorder planning, and seasonal misalignment.

Six Sigma and the DMAIC Framework in Retail

The Six Sigma methodology, especially the DMAIC (Define, Measure, Analyze, Improve, Control) framework, has been widely adopted in retail to improve inventory and process efficiency. According to Laureani and Antony (2012), Six Sigma provides a structured approach to problem-solving and has been successful in reducing variability and improving customer service levels.

This project followed the DMAIC approach to systematically analyze stockout issues and propose improvements. For example, root causes identified during the analysis stage informed the design of dynamic safety stock policies and automated replenishment mechanisms during the improvement stage. The control phase included suggestions such as weekly stockout KPI monitoring using p-charts.

Promotions, Seasonality, and Demand Variability

Promotions and seasonal demand shifts significantly impact sales volumes. If not managed properly, these factors can lead to overstocking or understocking. Blattberg, Briesch, and Fox (1995) found that while promotional events boost short-term sales, they can distort long-term demand forecasts unless properly accounted for in planning. Similarly, failure to recognize seasonal patterns may result in frequent stockouts or excess inventory.

This study addressed these issues by using Python tools to visualize seasonal trends and assess the effectiveness of promotional campaigns. These insights were used to recommend more responsive inventory strategies and alignment of forecasting models with promotion calendars.

Technology-Driven Improvement Strategies

With advancements in computing and data science, retailers now rely on machine learning and real-time analytics to improve their inventory and demand management systems. Python has emerged as one of the most powerful tools for data analysis in retail due to its flexibility, open-source nature, and extensive library ecosystem. According to Zhang, Eddy Patuwo, and Hu (2007), artificial neural networks and machine learning models are capable of capturing nonlinear demand patterns and significantly improving forecast accuracy.

In this project, Python was used for data cleaning, visualization, inventory analysis, and performance monitoring. Techniques such as SKU-level segmentation, dynamic safety stock calculation, and control chart monitoring were implemented as technology-based solutions to enhance forecasting precision and improve operational responsiveness.

3.Methodology

3.1. Dataset Description

The analysis utilized a simulated retail store inventory dataset covering the period from January 1, 2022, to January 30, 2024 from https://www.kaggle.com/datasets/atomicd/retail-store-inventory-and-demand-forecasting?resource=download. It includes data from five major product categories such as Electronics, Clothing, Groceries, Toys, and Furniture.Also, it incorporates several external influencing factors such as Weather conditions,Promotional campaigns,Competitor pricing and activities,Seasonal effects,Epidemic-related scenarios.Also, It comprises 76,000 observations (rows) and 17 variables. This analysis focuses on understanding the relationship between stockouts and three key factors: Seasonality, Product Categories, and Promotional Status.

3.2. Define and Measure Phase

Overall Stockout Rate:

Stockout Rate (%) = (Total number of stockout rows ÷ Total rows) × 100

Descriptive statistics were calculated for all factors, etc., using .describe().

3.3. Analyze Phase

The stockout and sales behavior were analyzed using group-based calculations and visualizations.

Seasonal Impact

Total sales = Sum of Units sold where Seasonality = that season

Total stockout = Sum of stockout values where Seasonality = that season

Category-wise Stockouts

Total Units sold = Sum of Units sold where Category = that category

Total stockouts = Sum of stockout values where Category = that category

Promotional Status vs Sales

Total Units sold = Sum of Units sold per each promotional status

Average Units Sold = (Total Units Sold ÷ Number of records in that group)

Total stockouts = Sum of stockout values per each promotional status

4.Implementation

4.1. Dataset Description

4.1.1 Source

import pandas as pd

```
import pandas as pd

df = pd.read_csv('/content/retail_data.csv')
```

4.1.2 Column Descriptions

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76000 entries, 0 to 75999
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                76000 non-null  object
 1   Store ID            76000 non-null  object
 2   Product ID          76000 non-null  object
 3   Category            76000 non-null  object
 4   Region              76000 non-null  object
 5   Inventory Level     76000 non-null  int64
 6   Units Sold          76000 non-null  int64
 7   Units Ordered       76000 non-null  int64
 8   Price               76000 non-null  float64
 9   Discount            76000 non-null  int64
 10  Weather Condition   76000 non-null  object
 11  Promotion           76000 non-null  int64
 12  Competitor Pricing  76000 non-null  float64
 13  Seasonality         76000 non-null  object
 14  Epidemic            76000 non-null  int64
 15  Demand              76000 non-null  int64
dtypes: float64(2), int64(7), object(7)
memory usage: 9.3+ MB
```

Observations:

All columns have 76,000 non-null counts, indicating no missing data.

4.2 Baseline Metrics Calculation

4.2.1 Descriptive Statistics

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Inventory Level	76000.0	301.062842	226.510161	0.00	136.0000	227.0	408.0000	2267.00
Units Sold	76000.0	88.827316	43.994525	0.00	58.0000	84.0	114.0000	426.00
Units Ordered	76000.0	89.090645	162.404627	0.00	0.0000	0.0	121.0000	1616.00
Price	76000.0	67.726028	39.377899	4.74	31.9975	64.5	95.8300	228.03
Discount	76000.0	9.087039	7.475781	0.00	5.0000	10.0	10.0000	25.00
Promotion	76000.0	0.328947	0.469834	0.00	0.0000	0.0	1.0000	1.00
Competitor Pricing	76000.0	69.454029	40.943818	4.29	32.6200	65.7	97.9325	261.22
Epidemic	76000.0	0.200000	0.400003	0.00	0.0000	0.0	0.0000	1.00
Demand	76000.0	104.317158	46.964801	4.00	71.0000	100.0	133.0000	430.00

4.3 Definition and Measurement of Stockouts

4.3.1 Definition

In this analysis, a stockout is defined as a situation where the 'Inventory Level' at the start of the day for a specific product at a specific store on a given date is zero.

4.3.2 Creation of the 'stockout' Column

```
df['stockout'] = df['Inventory Level'].apply(lambda x: 1 if x == 0 else 
0)
```

- If 'Inventory Level' == 0, 'stockout' = 1 (stockout occurred).
- If 'Inventory Level' > 0, 'stockout' = 0 (no stockout).

4.3.3 Overall Stockout Rate

```
total_stockouts = df['stockout'].sum()
total_observations = len(df)
overall_stockout_rate = (total_stockouts / total_observations) * 100
print(f"Total stockouts: {total_stockouts}")
print(f"Total observations: {total_observations}")
```

```
print(f"Overall stockout rate: {overall_stockout_rate:.4f}%")
```

```
Total stockouts: 406
Total observations: 76000
Overall stockout rate: 0.5342%
```

4.4 Measure Stockouts by Product and Store

<u>Stockouts per Product</u>

```
product_store_stockout_rate = df.groupby(['Product ID', 'Store
ID'])['stockout'].mean() * 100
print("\nStockout rate per product and store:")
product_store_stockout_rate
```

Stockout rate per product and store:

Product ID	Store ID	stockout
P0001	S001	0.000000
	S002	0.657895
	S003	1.052632
	S004	1.184211
	S005	1.184211
...	...	...

<u>Stockouts per Store</u>

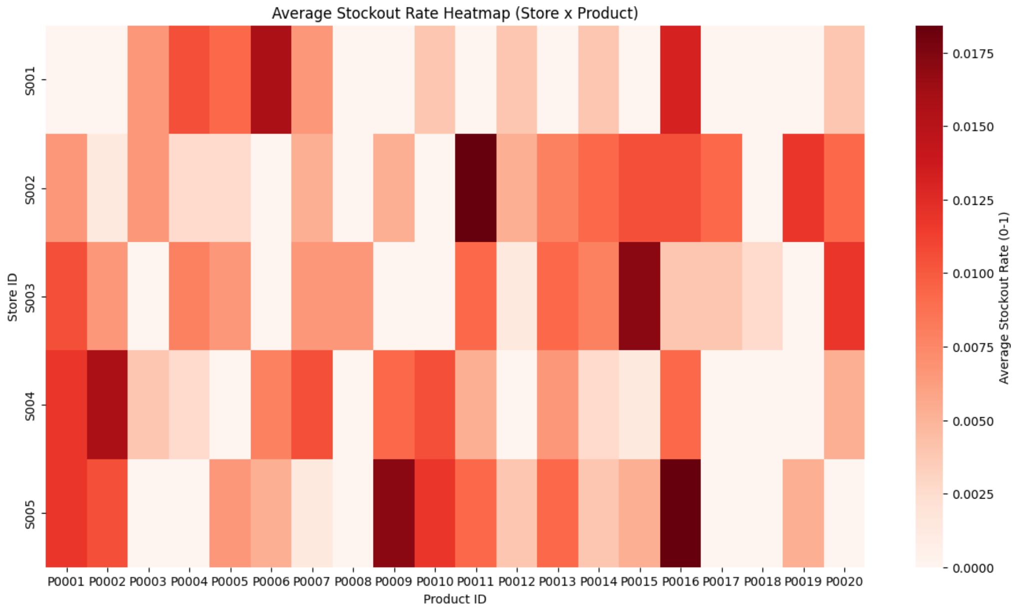
```
store_stockouts = df.groupby('Store
ID')['stockout'].sum().reset_index()
store_stockouts_sorted = store_stockouts.sort_values(by='stockout',
ascending=False)
print(store_stockouts_sorted)
```

```
  Store ID  stockout
1     S002        93
4     S005        91
2     S003        85
3     S004        78
0     S001        59
```

4.5 Visual Analysis of Stockout Patterns

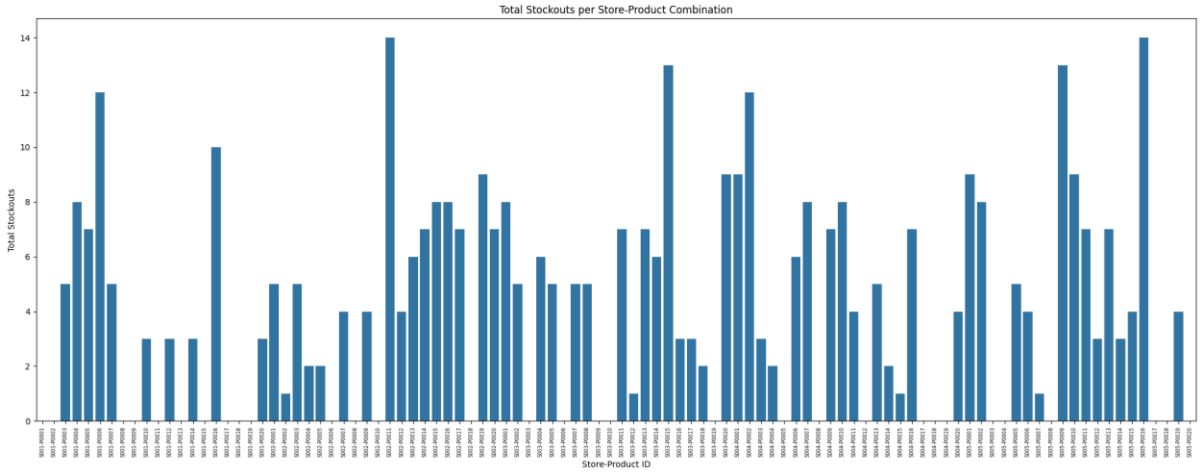
4.5.1 Heatmap: Average Stockout Rate (Store x Product)

```
import matplotlib.pyplot as plt
import seaborn as sns

stockout_matrix = df.pivot_table(index='Store ID', columns='Product
ID', values='stockout', aggfunc='mean', fill_value=0)
plt.figure(figsize=(15, 8))
sns.heatmap(stockout_matrix, cmap='Reds', cbar_kws={'label': 'Average
Stockout Rate (0-1)'})
plt.title('Average Stockout Rate Heatmap (Store x Product)')
plt.xlabel('Product ID')
plt.ylabel('Store ID')
plt.show()
```



4.5.2 Histogram/Bar Plot: Stockouts per Store-Product Combination

```
stockout_counts = df.groupby(['Store ID', 'Product
ID'])['stockout'].sum().reset_index()
stockout_counts['Store-Product'] = stockout_counts['Store
ID'].astype(str) + '-' + stockout_counts['Product ID'].astype(str)
plt.figure(figsize=(20, 8))
sns.barplot(x='Store-Product', y='stockout', data=stockout_counts)
plt.title('Total Stockouts per Store-Product Combination')
plt.xlabel('Store-Product ID')
plt.ylabel('Total Stockouts')
plt.xticks(rotation=90, fontsize=6)
plt.tight_layout()
plt.show()
```

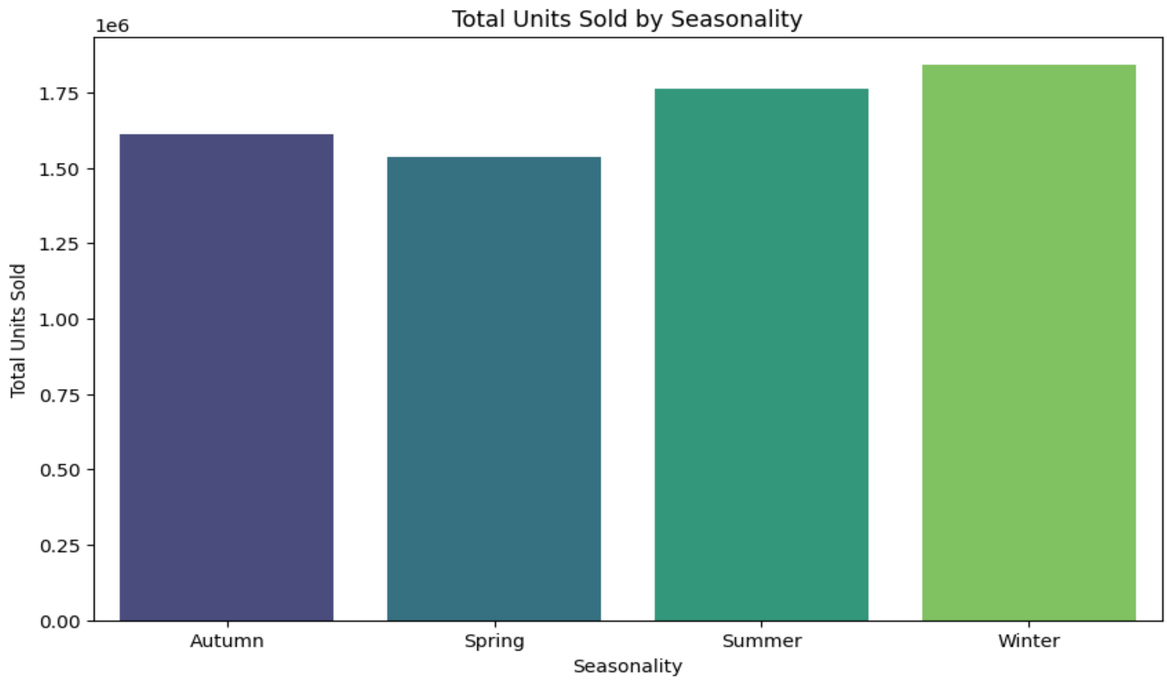


4.5.3 Analysis of Total units sold and stockout relationships with key factors (seasonality, promotion, category)

<u>Seasonality</u>

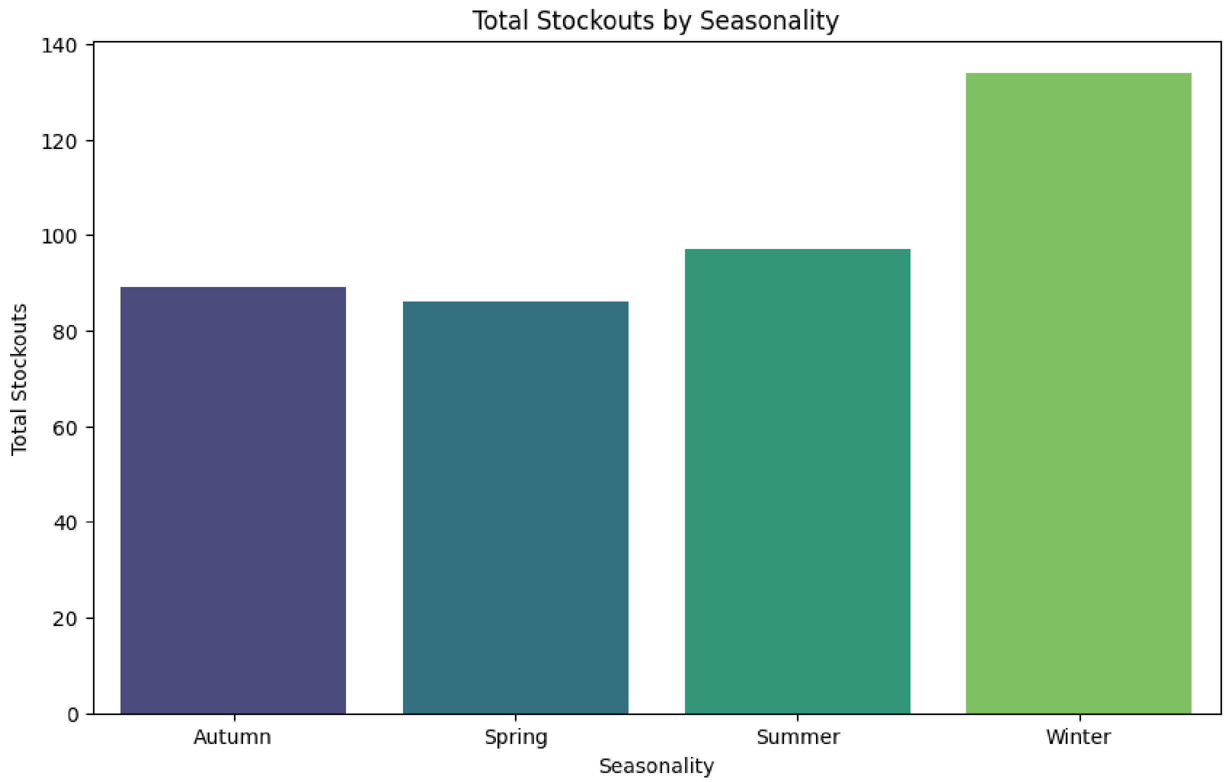
Total Units sold:

```
units_sold_by_seasonality = df.groupby('Seasonality')['Units
Sold'].sum().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Seasonality', y='Units Sold',
data=units_sold_by_seasonality, palette='viridis')
plt.title('Total Units Sold by Seasonality')
plt.xlabel('Seasonality')
plt.ylabel('Total Units Sold')
plt.show()
```



Total Stockouts:

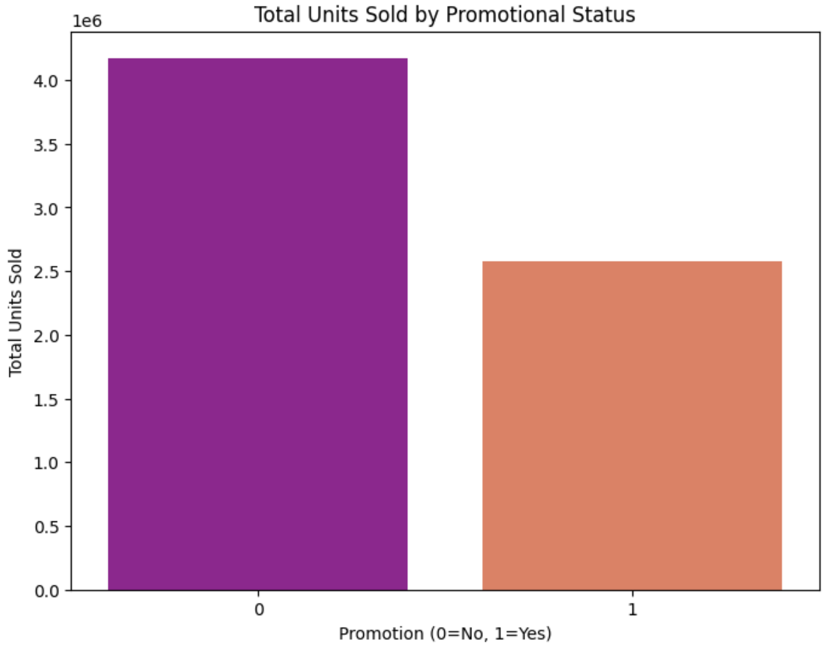
```
stockout_by_seasonality =
df.groupby('Seasonality')['stockout'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Seasonality', y='stockout',
data=stockout_by_seasonality, palette='viridis')
plt.title('Total Stockouts by Seasonality')
plt.xlabel('Seasonality')
plt.ylabel('Total Stockouts')
plt.show()
```



Promotional status

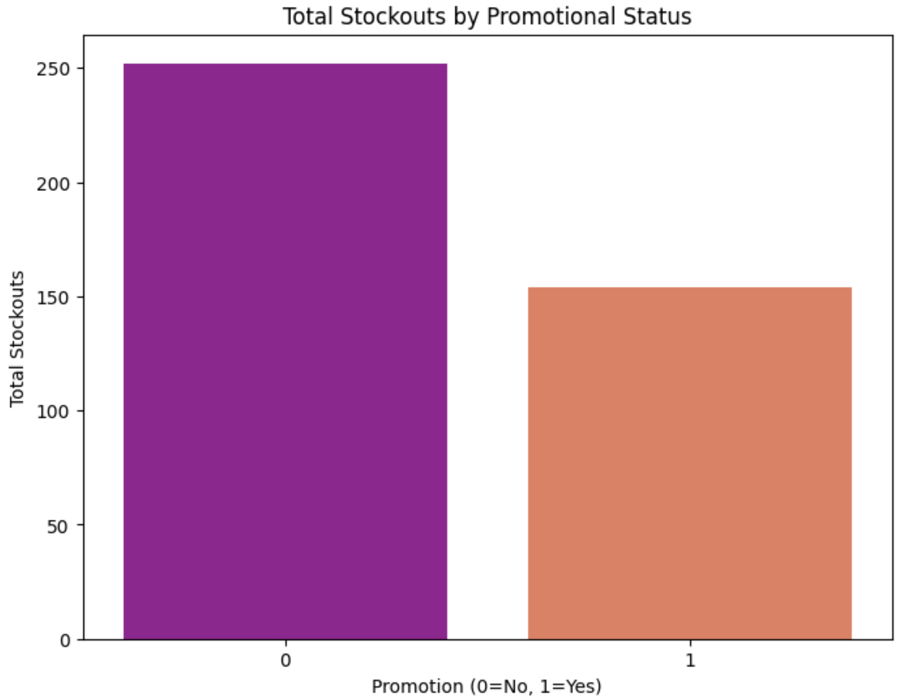
Total Units sold:

```
units_sold_by_promotion = df.groupby('Promotion')['Units
Sold'].sum().reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='Promotion', y='Units Sold',
data=units_sold_by_promotion, palette='plasma')
plt.title('Total Units Sold by Promotional Status')
plt.xlabel('Promotion (0=No, 1=Yes)')
plt.ylabel('Total Units Sold')
plt.show()
```



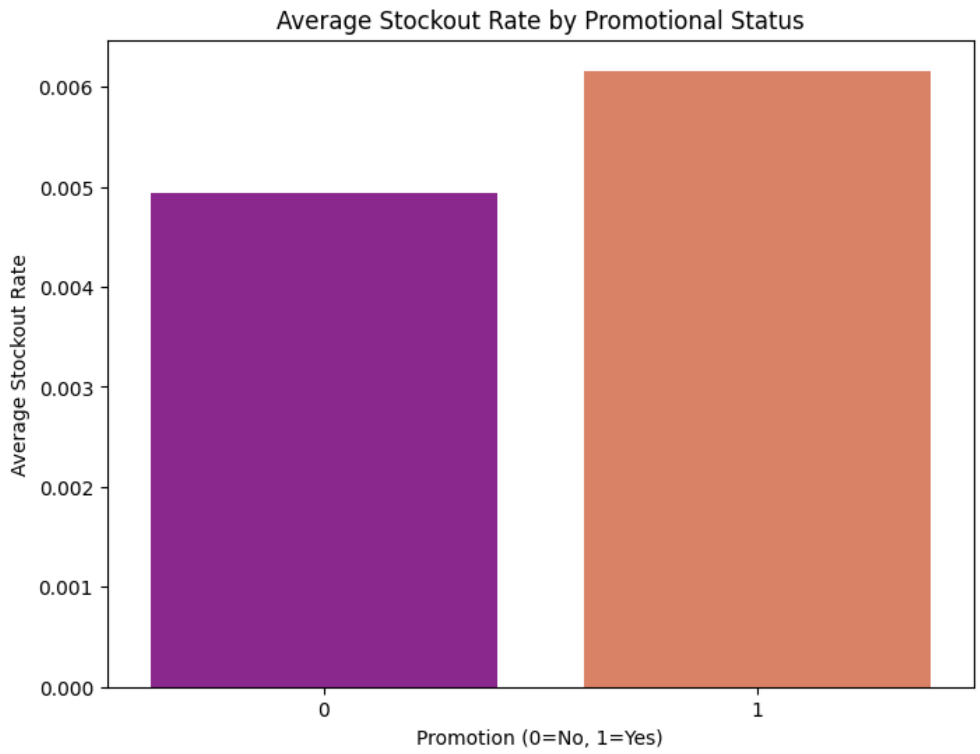
Total stockouts:

```
stockout_by_promotion =
df.groupby('Promotion')['stockout'].sum().reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='Promotion', y='stockout', data=stockout_by_promotion,
palette='plasma')
plt.title('Total Stockouts by Promotional Status')
plt.xlabel('Promotion (0=No, 1=Yes)')
plt.ylabel('Total Stockouts')
plt.show()
```



Average Stockouts:

```
avg_stockout_by_promotion =
df.groupby('Promotion')['stockout'].mean().reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='Promotion', y='stockout',
data=avg_stockout_by_promotion, palette='plasma')
plt.title('Average Stockout Rate by Promotional Status')
plt.xlabel('Promotion (0=No, 1=Yes)')
plt.ylabel('Average Stockout Rate')
plt.show()
```

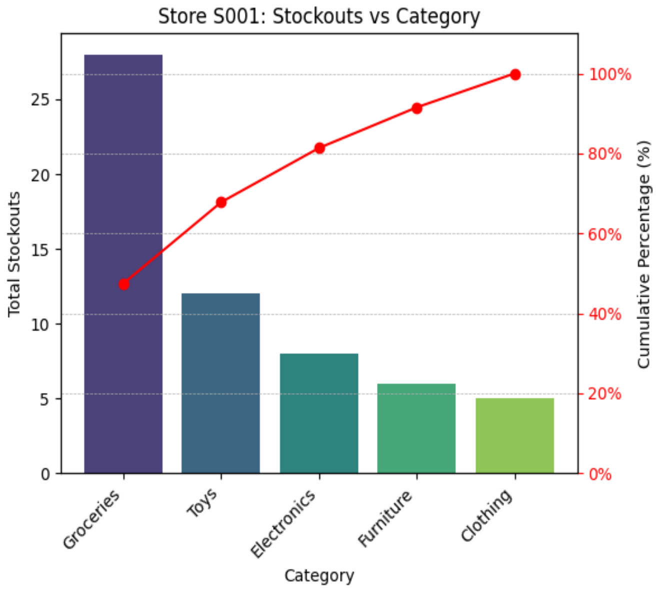


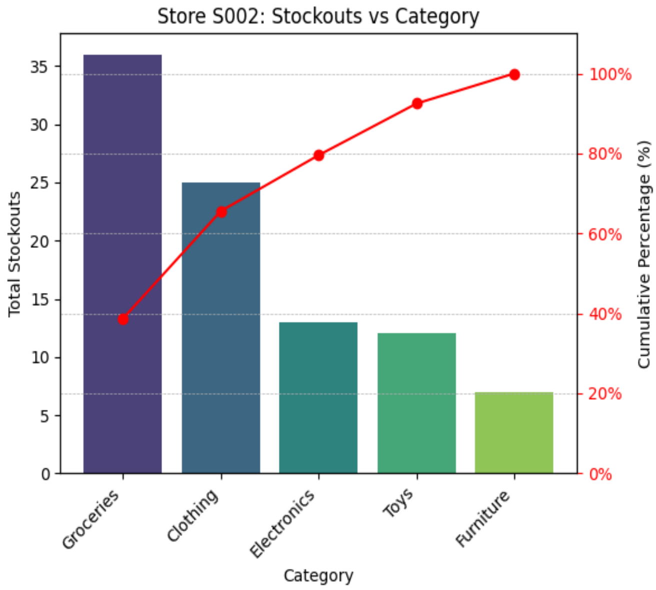
Category

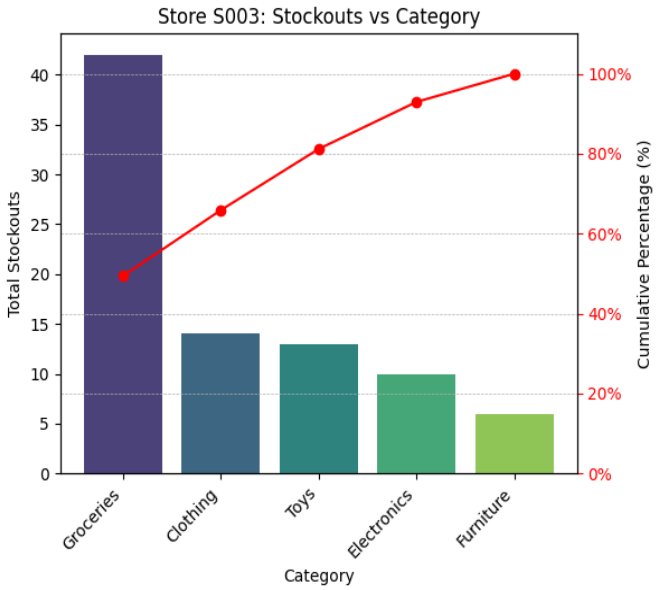
Total Stockouts:

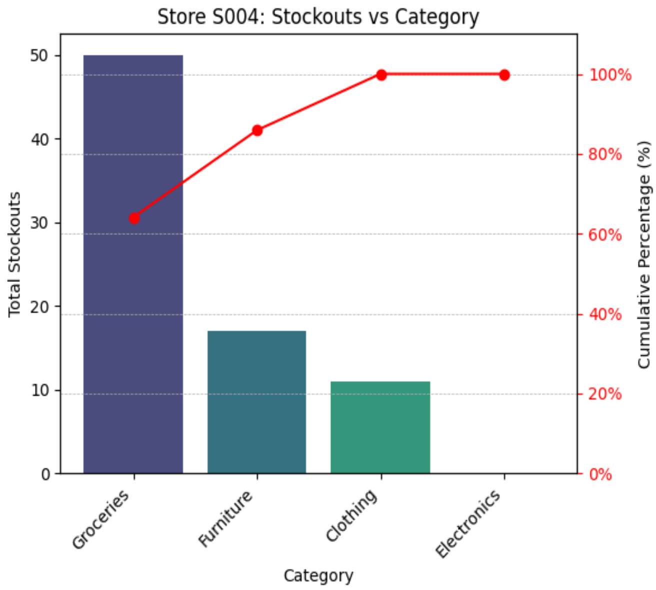
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from matplotlib.ticker import PercentFormatter


required_columns_pareto = ['stockout', 'Category', 'Store ID']
if not all(col in df.columns for col in required_columns_pareto):
  print("Error: Missing required columns for Pareto chart in the
dataframe.")
else:

  stores = df['Store ID'].unique()

  n_stores = len(stores)
  n_cols = min(2, n_stores)
  n_rows = (n_stores + n_cols - 1) // n_cols

  fig, axes = plt.subplots(n_rows, n_cols, figsize=(n_cols * 6, n_rows
* 5)) # Adjust overall figure size
  axes = axes.flatten()

  for i, store in enumerate(stores):
    ax1 = axes[i]


    store_data = df[df['Store ID'] == store].copy()

    stockout_by_category_store =
store_data.groupby('Category')['stockout'].sum().sort_values(ascending=
False).reset_index()

    if not stockout_by_category_store.empty and
stockout_by_category_store['stockout'].sum() > 0:
      stockout_by_category_store['cumulative_stockout'] =
stockout_by_category_store['stockout'].cumsum()
      stockout_by_category_store['cumulative_percentage'] =
(stockout_by_category_store['cumulative_stockout'] /
stockout_by_category_store['stockout'].sum()) * 100

      sns.barplot(x='Category', y='stockout',
data=stockout_by_category_store, ax=ax1, palette='viridis')
```

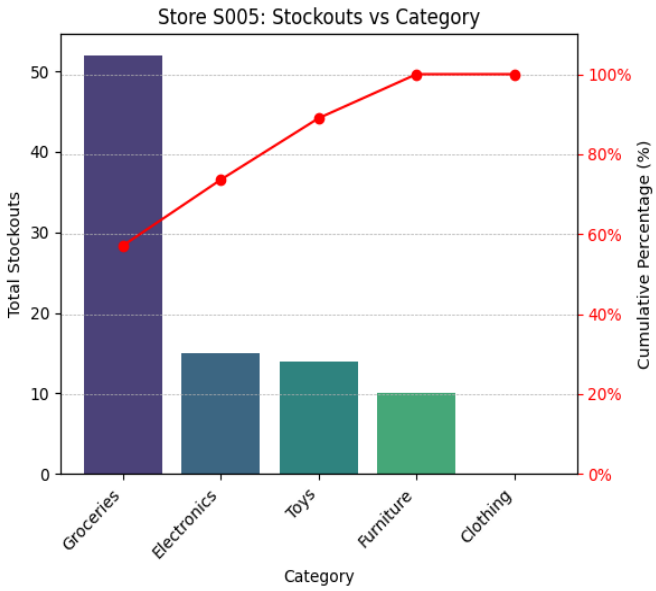
```
      ax1.set_title(f'Store {store}: Stockouts vs Category')
      ax1.set_xlabel('Category')
      ax1.set_ylabel('Total Stockouts')
      ax1.tick_params(axis='x', rotation=45)

      ax1.set_xticklabels(stockout_by_category_store['Category'],
rotation=45, ha='right')



      ax2 = ax1.twinx()
      ax2.plot(stockout_by_category_store['Category'],
stockout_by_category_store['cumulative_percentage'], color='red',
marker='o', linestyle='-')
      ax2.set_ylabel('Cumulative Percentage (%)')
      ax2.tick_params(axis='y', colors='red')
      ax2.set_ylim(0, 110)
      ax2.yaxis.set_major_formatter(PercentFormatter())

      ax2.grid(True, which='both', linestyle='--', linewidth=0.5)

    else:
      fig.delaxes(axes[i])
      print(f"No stockout data available or total stockouts are zero
for Store {store} to generate a Pareto chart.")

  for j in range(i + 1, len(axes)):
      fig.delaxes(axes[j])

  plt.tight_layout()
  plt.show()
```


Store S001: Stockouts vs Category
Total Stockouts
Cumulative Percentage (%)
0
5
10
15
20
25
0%
20%
40%
60%
80%
100%
Groceries
Toys
Electronics
Furniture
Clothing
Category


Store S002: Stockouts vs Category
Total Stockouts
Cumulative Percentage (%)
0
5
10
15
20
25
30
35
0%
20%
40%
60%
80%
100%
Groceries
Clothing
Electronics
Toys
Furniture
Category


Store S003: Stockouts vs Category
Total Stockouts
Cumulative Percentage (%)
0
5
10
15
20
25
30
35
40
0%
20%
40%
60%
80%
100%
Groceries
Clothing
Toys
Electronics
Furniture
Category


Store S004: Stockouts vs Category
Total Stockouts
Cumulative Percentage (%)
0
10
20
30
40
50
0%
20%
40%
60%
80%
100%
Groceries
Furniture
Clothing
Electronics
Category


Store S005: Stockouts vs Category
Total Stockouts
Cumulative Percentage (%)
0
10
20
30
40
50
0%
20%
40%
60%
80%
100%
Groceries
Electronics
Toys
Furniture
Clothing
Category

4. Discussion

The analysis of stockouts and sales across 76,000 observations reveals critical insights into the operational dynamics that influence product availability. While our data provides significant evidence of trends, there are several potential causes of stockouts that warrant further investigation, particularly in relation to inventory flow and reorder processes. These hypotheses are based on observable patterns in the data and could explain some of the challenges identified.

4.1 Analysis

4.1.1 Seasonality Impact

The seasonal patterns in both sales and stockouts highlight the cyclical nature of demand, particularly in Winter and Summer. However, the observed stockouts during these periods may also point to deeper underlying issues in inventory flow. Specifically, there could be instances where slow-selling items are overstocked, while fast-selling items are not adequately replenished to keep up with demand. This imbalance could exacerbate stockout incidents during peak sales periods.

4.1.2 Promotions and Demand Surges

Promotional events significantly boost sales but also lead to a rise in stockouts. While this is partly due to the surge in demand, another possible contributor could be a lack of reorder triggers in the current inventory system. Without an automated system to trigger replenishment orders when stock levels fall below a certain threshold, products may not be restocked in time, leading to stockouts.

4.1.3 Category-Level Disparities

At the category level, Groceries emerged as the top contributor to stockouts, with the data suggesting challenges related to demand volatility and perishability. However, it is also possible that the inventory flow issues (overstocking slow movers and understocking fast movers) are contributing to grocery stockouts. Replenishment inefficiencies could be exacerbating these issues, particularly for high-demand grocery items that require more frequent stock updates.

In other categories, such as Clothing, Electronics, and Furniture, stockouts vary by store, reflecting possible local supply chain issues or mismatched inventory strategies. While the observed stockouts in these categories can be linked to variations in demand patterns and promotional activity, inventory management practices, such as inconsistent reorder mechanisms or manual order forecasting, may also be influencing these discrepancies.

4.2 Improve

To address the challenges identified in the analysis, we propose several improvements to inventory management practices:

Seasonal Demand Alignment: To tackle predictable demand spikes during Winter and other high-demand seasons, we suggest implementing seasonal forecasting models. These models would anticipate seasonal variations and set auto-replenishment buffers to ensure stock levels are preemptively adjusted before peak periods. This would help avoid stockouts during crucial sales windows.

Promotion-Aware Forecasting Engine: Given the significant sales increase during promotional periods, we recommend developing a promotion-aware forecasting engine. This system would link promotion calendars to demand forecasts, utilizing machine learning to predict sales lifts and adjust inventory levels accordingly. By integrating promotional data with demand predictions, we can better align inventory with actual sales patterns during promotions.

Dynamic Safety Stock Calculation: Traditional inventory management often relies on fixed safety stock buffers. We propose shifting to a dynamic safety stock calculation approach, which calculates safety stock per product and store based on recent demand variability. This ensures more accurate replenishment and helps avoid both stockouts and overstocking, especially for slow-moving items.

SKU Segmentation Using ABC Analysis: To optimize inventory management, we suggest implementing SKU segmentation using the ABC method. This method classifies items into three categories based on sales velocity: A-class (fast-moving), B-class (moderate-moving), and C-class (slow-moving). By applying tighter control to A-class items and minimizing excess stock in C-class items, we can ensure better inventory allocation and reduce stockouts.

Weekly KPI Tracking Using P-Charts: To effectively monitor stockouts and forecast performance, we recommend introducing weekly KPI tracking using P-charts. These charts will help visualize stockout trends and provide early insights into inventory issues, enabling quicker corrective actions and more agile decision-making.

Closed-Loop Forecast Correction Mechanism: Lastly, we propose a closed-loop forecast correction mechanism. This system would continuously analyze forecast errors and adjust future predictions automatically, allowing the inventory management system to learn and improve over time. This approach can enhance long-term forecasting accuracy and make the system more responsive to changing demand.

Together, these improvements aim to create a more data-driven, proactive inventory management system that can better anticipate demand surges, optimize stock levels, and reduce stockouts.

4.3 Control

To ensure that the improvements are sustainable and effectively integrated into daily operations, we recommend the following control measures:

Make New Inventory Rules Part of Daily Work: The new inventory rules and processes should be embedded in the daily workflow. By making these changes part of the routine, we ensure consistency and alignment across all teams involved in inventory management.

Regular Stock Level Monitoring: Implement regular monitoring of stock levels using visual charts and dashboards. This will provide real-time visibility into inventory status and help identify potential stockout risks before they escalate.

Frequent Process Audits and Feedback: Conduct regular audits and gather feedback from inventory management staff. This will help identify any inefficiencies or gaps in the process, allowing for ongoing improvements and refinements.

Clear Instructions and Plans: Ensure that there are clear documentation and action plans for handling inventory replenishment, especially during peak demand periods and promotions. Well-defined guidelines will help ensure consistency in execution.

Proactive Alerts for Early Problem Detection: Set up alerts to notify inventory managers when stock levels fall below critical thresholds. These alerts will provide early warnings, allowing for quick intervention and restocking before stockouts occur.

Confirm Long-Term Effectiveness and Garner Support: After implementing the improvements, continuously monitor results to confirm that stockouts have been reduced and inventory management has improved. Additionally, ensure management buy-in and support to sustain these changes over the long term.

By combining these improvements with effective control measures, we can transition to a more data-driven, proactive approach in inventory management, ultimately reducing stockouts and boosting both sales and customer satisfaction.

5. Conclusion

The analysis of stockouts across 76,000 observations revealed a stockout rate of 0.5342%, highlighting the need for more effective inventory management. Significant seasonal peaks in Winter and Summer, especially during promotions, indicate a mismatch between demand surges and inventory replenishment. Groceries contributed the highest proportion of stockouts, while other categories like Clothing and Electronics showed varied contributions across stores.

Statistical insights suggest that inventory flow inefficiencies, such as overstocking slow-moving items and understocking high-demand products, are major causes of stockouts. Additionally, the lack of automated reorder systems during demand surges exacerbates these issues. However, this analysis is based on a subset of data, and other potential causes, such as supply chain disruptions, forecasting errors, manual ordering processes, and supplier delays, have not been fully explored.

To improve stock availability, we recommend implementing seasonal forecasting models, promotion-aware forecasting, and dynamic safety stock calculations. Adopting ABC SKU segmentation and weekly KPI tracking will further optimize inventory levels and reduce stockouts. By integrating these improvements and real-time monitoring tools, we aim to reduce stockout incidents, enhance sales efficiency, and increase customer satisfaction.

In conclusion, while the current stockout rate suggests significant opportunities for improvement, addressing both demand fluctuations and inventory control inefficiencies through data-driven strategies will mitigate these issues. Further investigation into additional causes, such as supply chain disruptions and manual processes, will provide a more comprehensive understanding and refine inventory practices across all stores.

6.References

Blattberg, R. C., Briesch, R., & Fox, E. J. (1995). How promotions work. *Marketing Science*, 14(3), G122–G132.

Chopra, S., & Meindl, P. (2019). *Supply Chain Management: Strategy, Planning, and Operation* (7th ed.). Pearson.

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2008). Effective forecasting and judgmental adjustments: An empirical evaluation. *International Journal of Forecasting*, 24(1), 3–22.

Laureani, A., & Antony, J. (2012). Critical success factors for the effective implementation of Lean Sigma: Results from an empirical study. *International Journal of Lean Six Sigma*, 3(4), 274–283.

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016). *Inventory and Production Management in Supply Chains* (4th ed.). CRC Press.

Taylor, D. H., & Fearne, A. (2009). Demand management in fresh food value chains: A framework for analysis and improvement. *Supply Chain Management: An International Journal*, 14(5), 379–392.

Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (2007). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35–62