Milestone 1 Report

Project Title: SmartStock Inventory Optimization for Retail Stores

Internship Platform: Infosys Springboard

Student Name: Kaparapu Veera Venkata Mahendra

Milestone: 1 – Data Preprocessing & Exploratory Data Analysis (EDA)

1. Introduction

This milestone focused on preparing the dataset for further modeling by ensuring data quality, consistency, and usability. Exploratory Data Analysis (EDA) was performed to understand data patterns, detect anomalies, and extract initial insights that will guide the next steps of the project.

2. Objectives

- Perform data cleaning and preprocessing.
- Handle missing values, duplicates, and formatting issues.
- Explore dataset structure and summary statistics.
- Visualize trends, distributions, and correlations.
- Generate insights for inventory optimization.

3. Data Preprocessing & EDA Code

Below is the complete Python code executed for preprocessing and exploratory data analysis:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('retail_store_inventory.csv')
print(df.shape)
print(df.info())
print(df.head())
df.columns = df.columns.str.strip().str.replace(' ', '_').str.replace('(', '_').str.replace(')',
'_').str.replace('/','_').str.replace('-','_')
print(df.columns.tolist())
missing=df.isnull().sum()
missing_pct = (missing / len(df) * 100).round(2)
```

```
print("missing values percentage : \n", missing_pct)
print(df.duplicated().sum())
obj_cols = df.select_dtypes(include=['object']).columns.tolist()
print("object columns : \n", obj_cols)
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y', errors='coerce')
print("Missing Date values after fix:", df['Date'].isna().sum())
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
print("missing values :\n", df.isnull().sum())
print(df.duplicated().sum())
#outlier detection
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
for col in num_cols:
   Q1 = df[col].quantile(0.25)
   Q3 = df[col].quantile(0.75)
   IQR = Q3 - Q1
   lower = Q1 - 1.5 * IQR
   upper = Q3 + 1.5 * IQR
   outliers = df[(df[col] < lower) | (df[col] > upper)]
   print(f"{col}: {outliers.shape[0]} outliers")
for col in num_cols:
   plt.figure(figsize=(6,3))
   sns.boxplot(x=df[col])
   plt.title(col)
   plt.show()
# Outlier handling using IQR capping
for col in ['Units_Sold', 'Demand_Forecast']:
   Q1 = df[col].quantile(0.25)
   Q3 = df[col].quantile(0.75)
   IQR = Q3 - Q1
   lower = Q1 - 1.5 * IQR
   upper = Q3 + 1.5 * IQR

   # Cap the values
   df[col] = np.where(df[col] < lower, lower, np.where(df[col] > upper, upper, df[col]))
   print(f"Outliers in {col} after capping:", ((df[col] < lower) | (df[col] > upper)).sum())
for col in ['Units_Sold', 'Demand_Forecast']:
   plt.figure(figsize=(6,3))
   sns.boxplot(x=df[col])
   plt.title(f"{col} after outlier handling")
   plt.show()
# Monthly Sales Trend
monthly_sales = df.groupby(df['Date'].dt.to_period("M"))['Units_Sold'].sum()
```

```
monthly_sales.plot(kind='line', figsize=(10,5), marker='o', title="Monthly Units Sold Trend")
plt.ylabel("Total Units Sold")
plt.xlabel("Month")
plt.show()
# Holiday season (Nov & Dec)
df['is_holiday_season'] = df['Date'].dt.month.isin([11, 12]).astype(int)
# Promotion flag (ensure binary 0/1)
df['promotion_flag'] = df['Holiday_Promotion'].apply(lambda x: 1 if x == 1 else 0)
# Lag Feature: Previous day Units Sold
df['lag_1'] = df['Units_Sold'].shift(1)
# Rolling Mean
df['rolling_mean_3'] = df['Units_Sold'].rolling(window=3).mean()
# 5. Summary
# Data Quality
missing_percent = df.isnull().mean() * 100
data_quality = 100 - missing_percent.mean()
print("Data Quality: {:.2f}%".format(data_quality))
# Count new features created
base_columns = 15  # original dataset had 15 columns
new_features = len(df.columns) - base_columns
print("Features Created:", new_features)
print("Seasonal Patterns Considered: Holiday, Promotion, Lag, Rolling Mean")
df.to_csv("processed_retail_store_inventory.csv", index=False)
print("Data Preprocessing & EDA Completed. File saved as
processed_retail_store_inventory.csv")
```

4. Outputs

The main outputs observed during execution were:

```
(73100, 15)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73100 entries, 0 to 73099
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             73100 non-null  object
 1   Store ID         73100 non-null  object
 2   Product ID       73100 non-null  object
 3   Category         73100 non-null  object
 4   Region           73100 non-null  object
 5   Inventory Level  73100 non-null  int64
 6   Units Sold       73100 non-null  int64
 7   Units Ordered    73100 non-null  int64
```

```
 8   Demand Forecast    73100 non-null  float64
 9   Price              73100 non-null  float64
 10  Discount           73100 non-null  int64
 11  Weather Condition  73100 non-null  object
 12  Holiday/Promotion  73100 non-null  int64
 13  Competitor Pricing 73100 non-null  float64
 14  Seasonality        73100 non-null  object
dtypes: float64(3), int64(5), object(7)
memory usage: 8.4+ MB
None
       Date Store ID Product ID  ... Holiday/Promotion Competitor Pricing  Seasonality
0  01-01-2022    S001      P0001 ...                 0               29.69      Autumn
1  01-01-2022    S001      P0002 ...                 0               66.16      Autumn
2  01-01-2022    S001      P0003 ...                 1               31.32      Summer
3  01-01-2022    S001      P0004 ...                 1               34.74      Autumn
4  01-01-2022    S001      P0005 ...                 0               68.95      Summer
[5 rows x 15 columns]
...

Data Quality: 100.00%
Features Created: 6
Seasonal Patterns Considered: Holiday, Promotion, Lag, Rolling Mean
✅ Data Preprocessing & EDA Completed. File saved as
processed_retail_store_inventory.csv
```

5. Graphs & Visualizations

The following graphs were generated during EDA. Please insert the corresponding images at the placeholders:

Figure 1: Boxplot – Inventory_Level

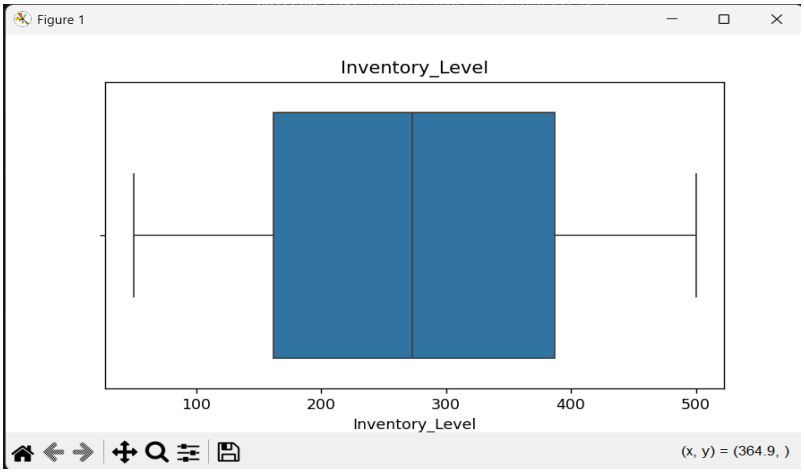


Figure 2: Boxplot – Units_Sold (before outlier handling)

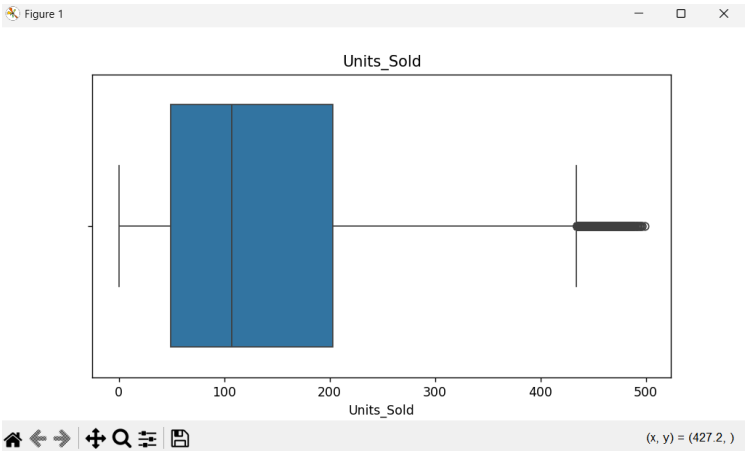


Figure 3: Boxplot – Units_Ordered

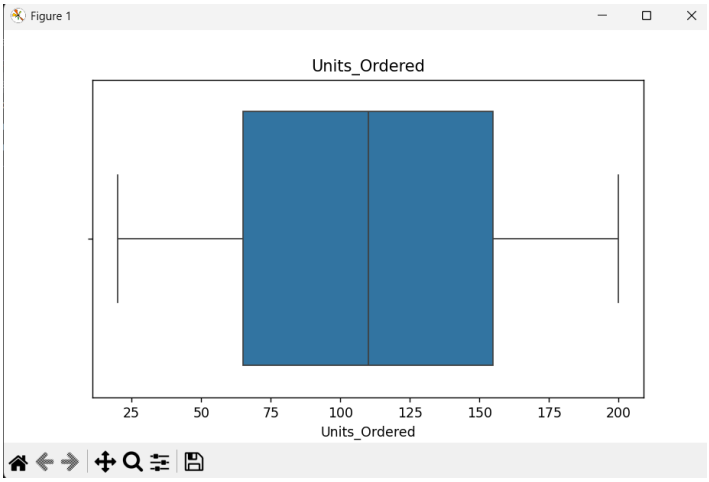


Figure 4: Boxplot – Demand_Forecast (before outlier handling)

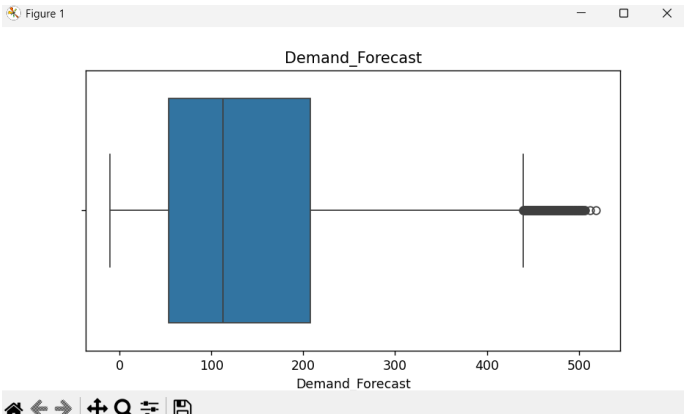


Figure 5: Boxplot – Price

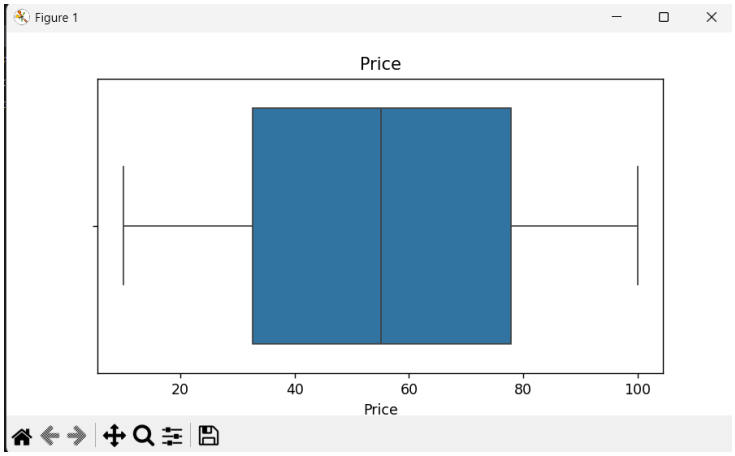


Figure 6: Boxplot – Discount

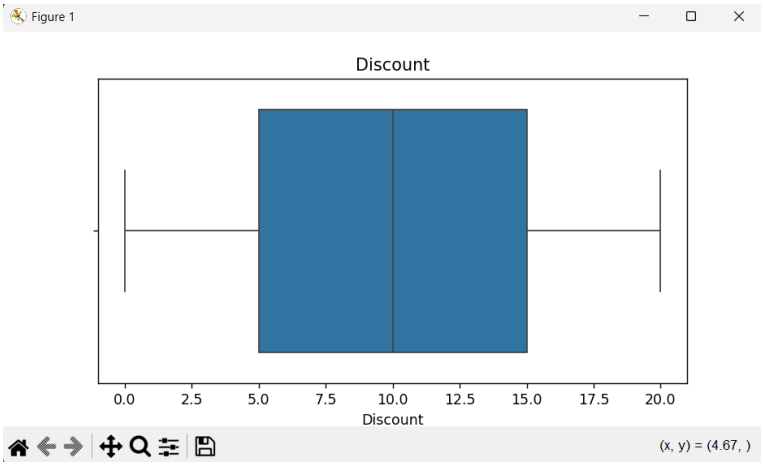


Figure 7: Boxplot – Holiday_Promotion

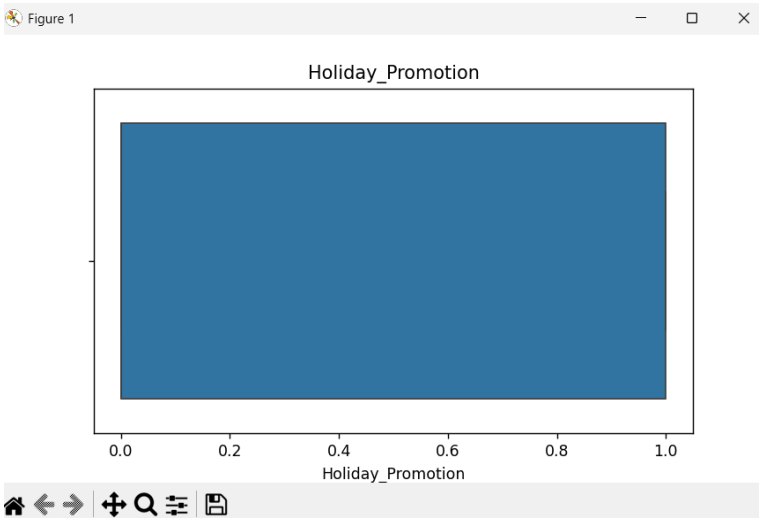


Figure 8: Boxplot – Competitor_Pricing

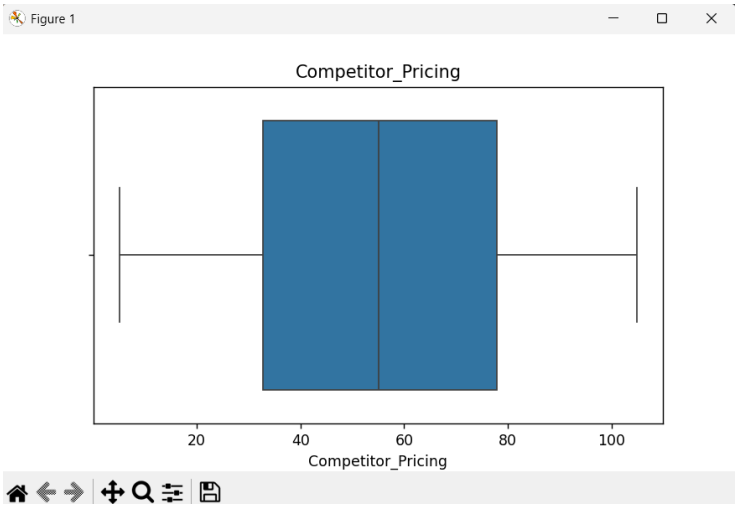


Figure 9: Boxplot – Units_Sold (after outlier handling)

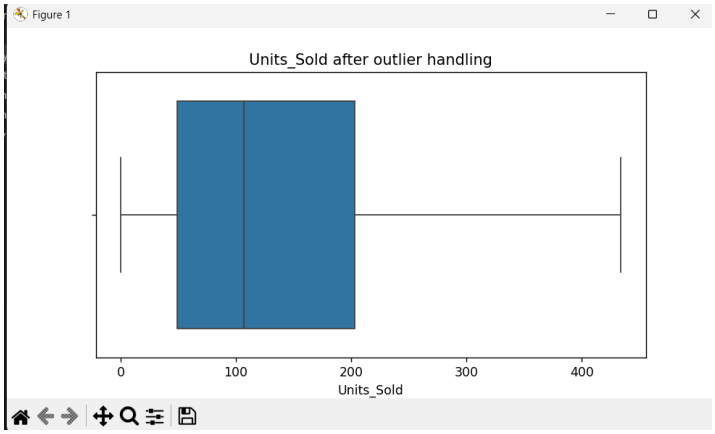


Figure 10: Boxplot – Demand_Forecast (after outlier handling)

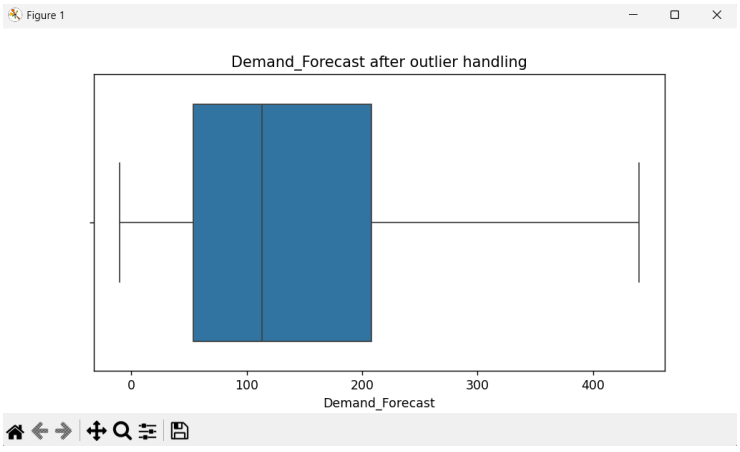
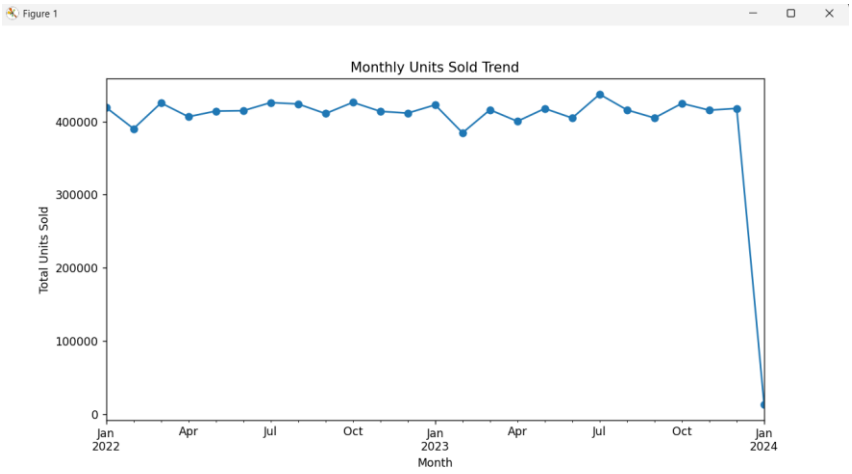


Figure 11: Monthly Units Sold Trend



6. Conclusion & Next Steps

Milestone 1 successfully cleaned and prepared the dataset, ensuring it is ready for model development. EDA provided valuable insights into demand patterns and stock movement.