ESTIMATING STOCK KEEPING UNIT USING ML

Data Exploration and Preprocessing

Dataset Overview:
The dataset used for this project contains over 150,000 records and includes 9 initial features:

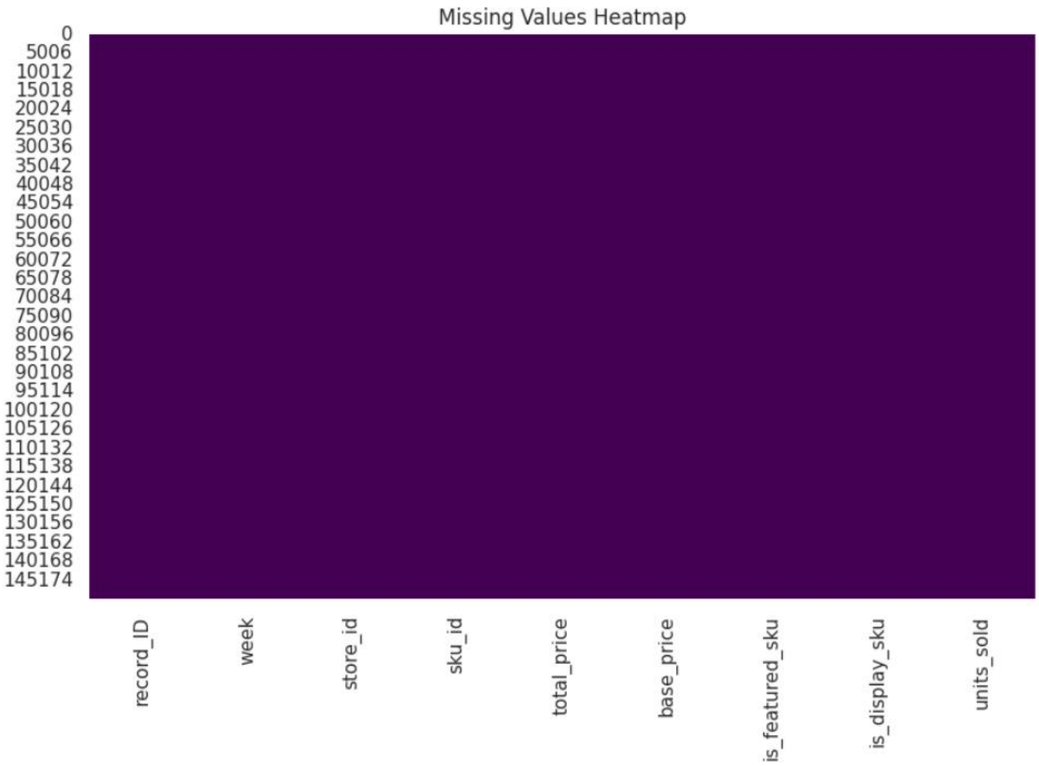
- record_ID
- week (string format)
- store_id
- sku_id
- total_price
- base_price
- is_featured_sku
- is_display_sku
- units_sold (target variable)

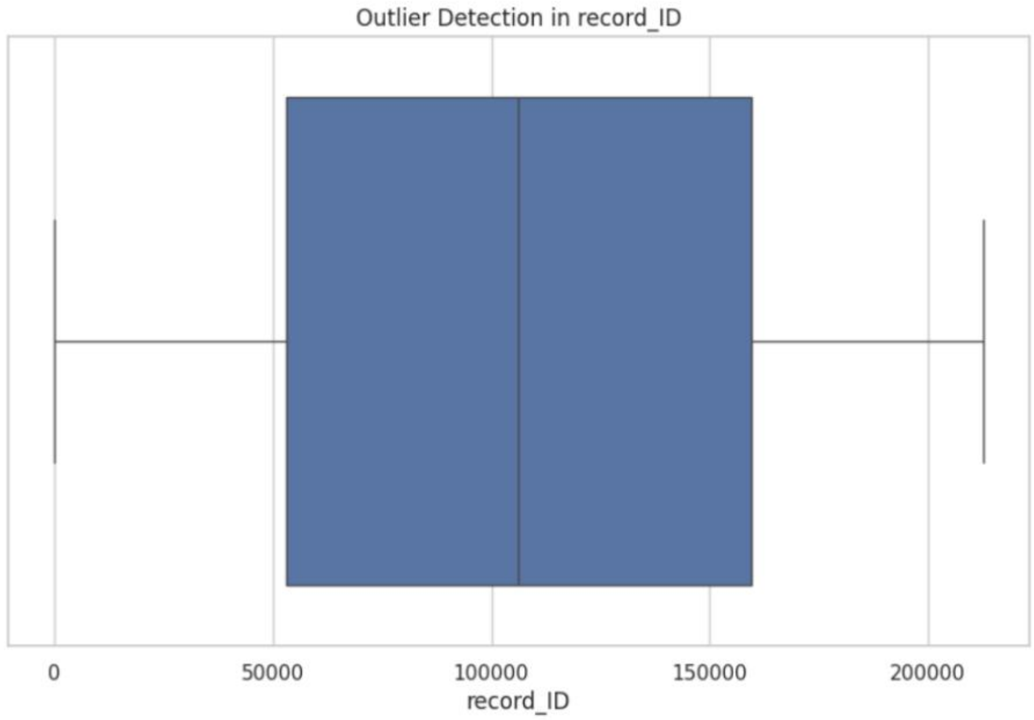
Exploratory Data Analysis (EDA):

- `.info()` revealed all data types and missing values. Only one missing value was found in `total_price`.
- `.describe()` showed summary statistics. The `units_sold` column had a high standard deviation and outliers.
- Seaborn heatmap was used to visualize missing data.

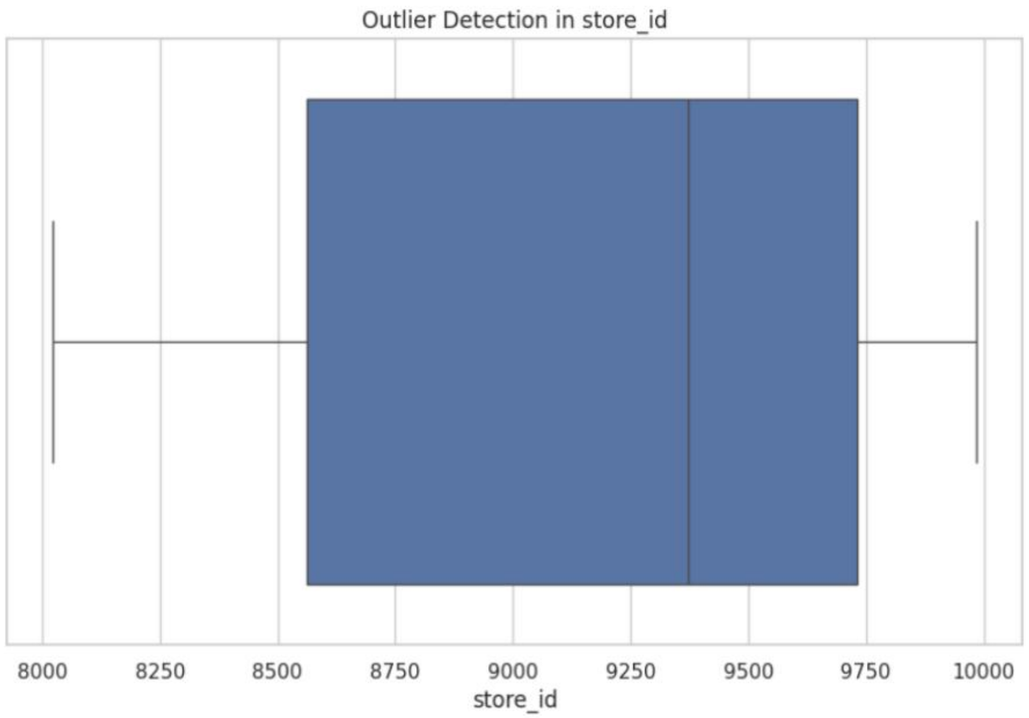
- Histograms and countplots helped analyze distributions.
- Boxplots helped detect outliers in continuous features.
- A correlation heatmap showed weak to moderate correlations among price and units_sold.

```
Missing Values:
record_ID          0
week               0
store_id           0
sku_id             0
total_price        1
base_price         0
is_featured_sku    0
is_display_sku     0
units_sold         0
dtype: int64
```

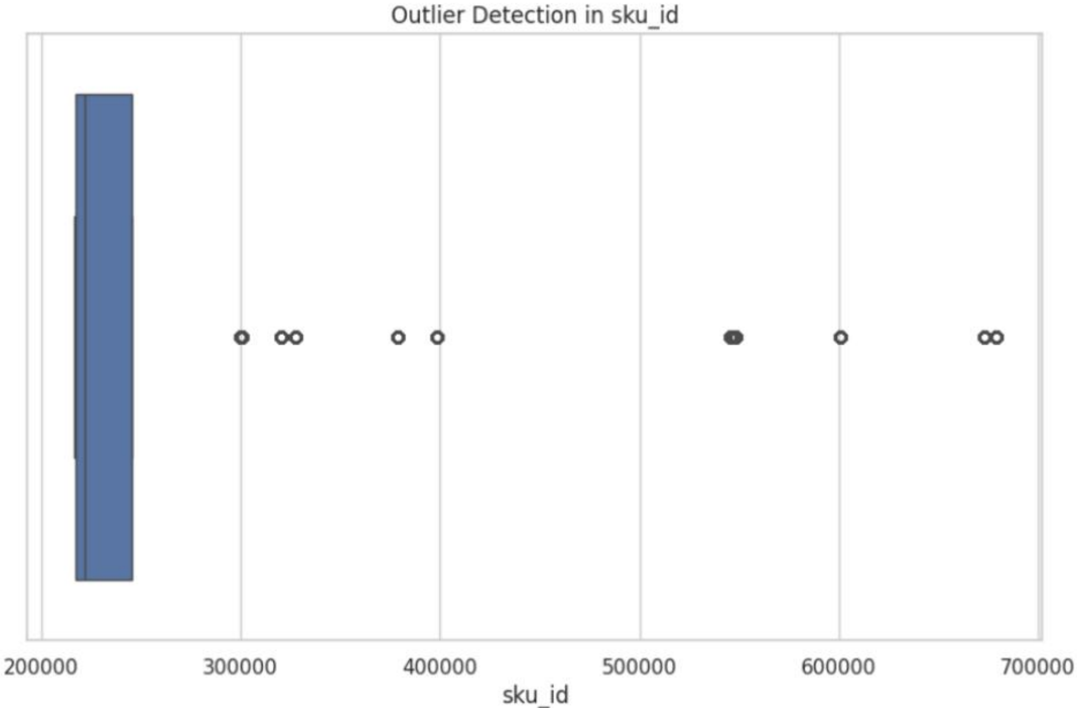


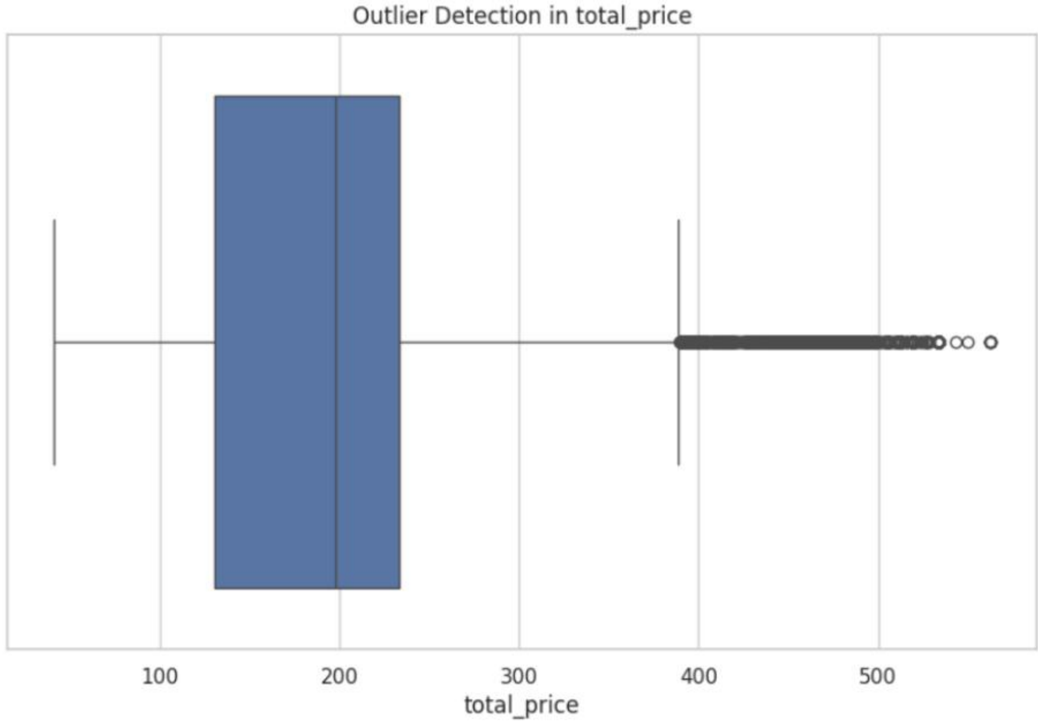


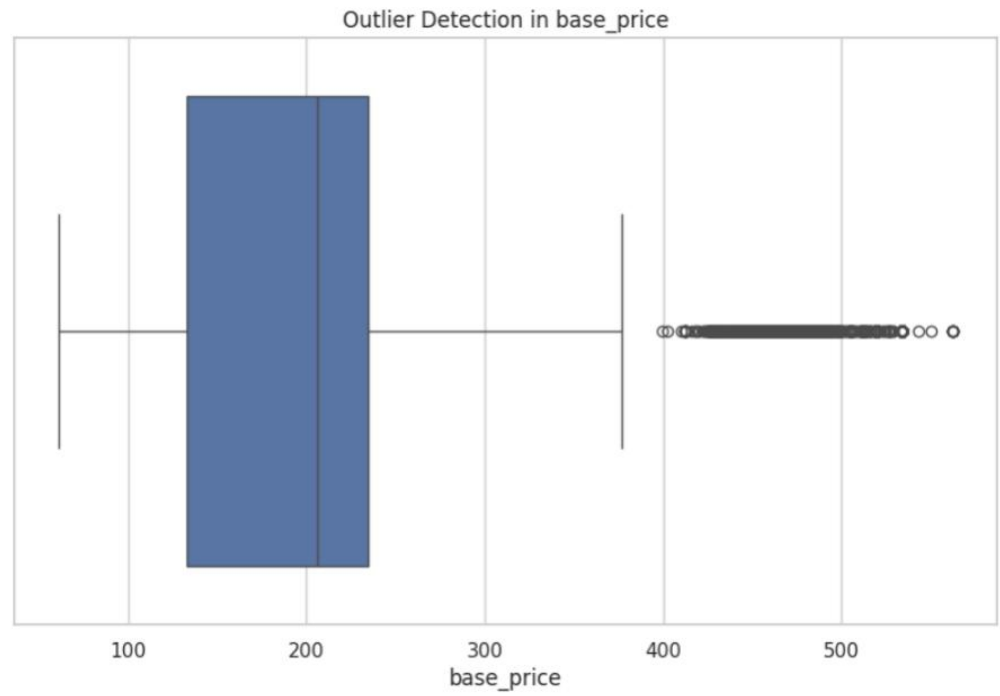
Outlier Detection in record_ID
0
50000
100000
150000
200000
record_ID

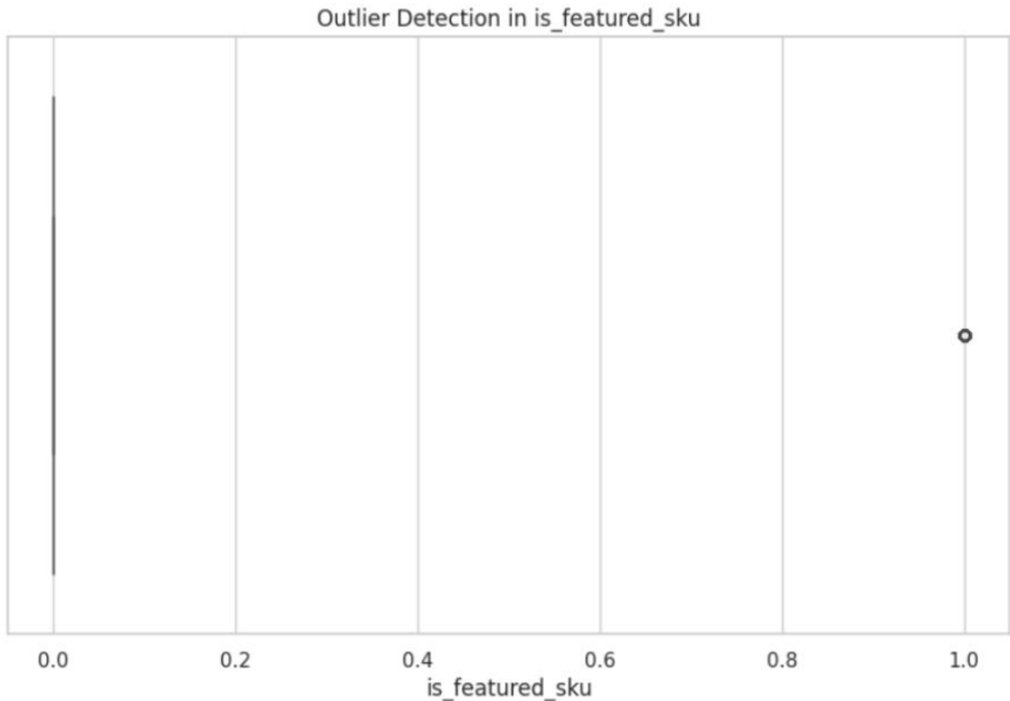


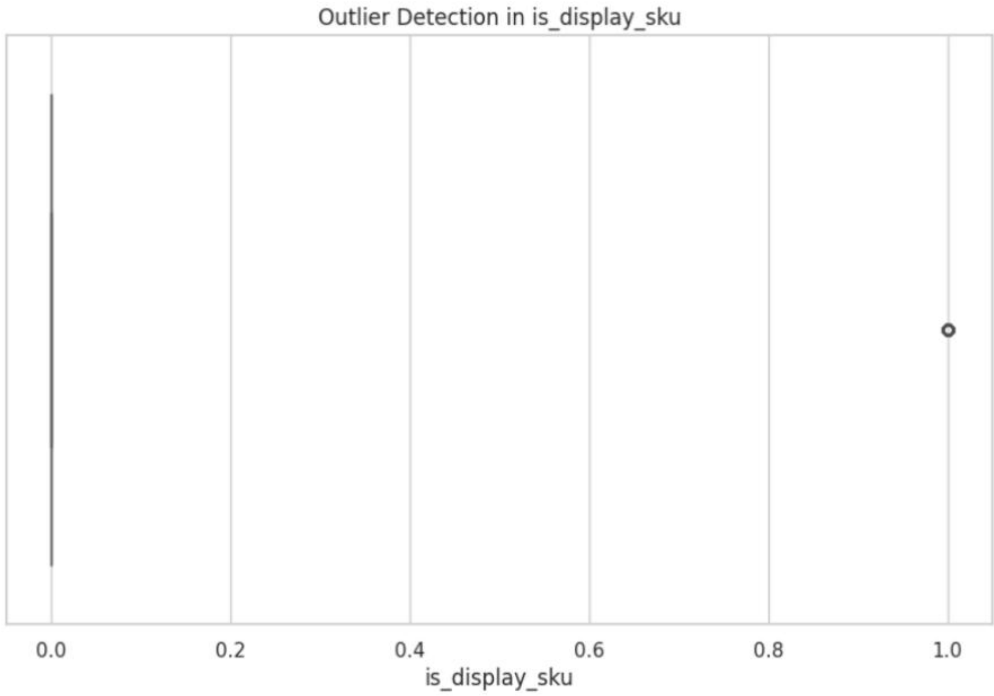
Outlier Detection in store_id
8000
8250
8500
8750
9000
9250
9500
9750
10000
store_id

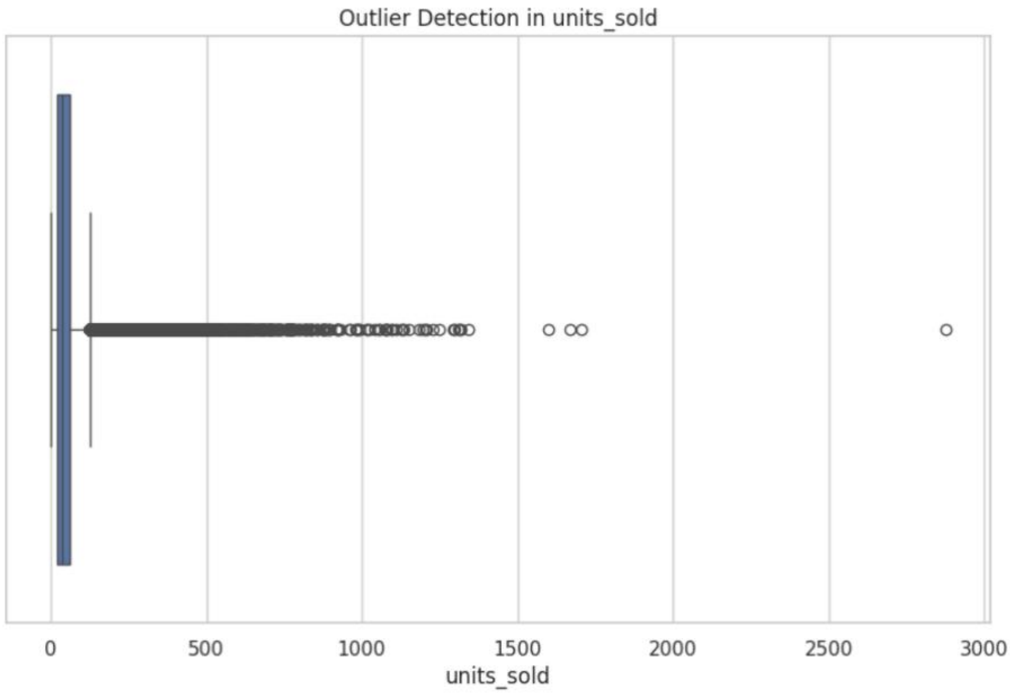

Outlier Detection in sku_id
200000
300000
400000
500000
600000
700000
sku_id

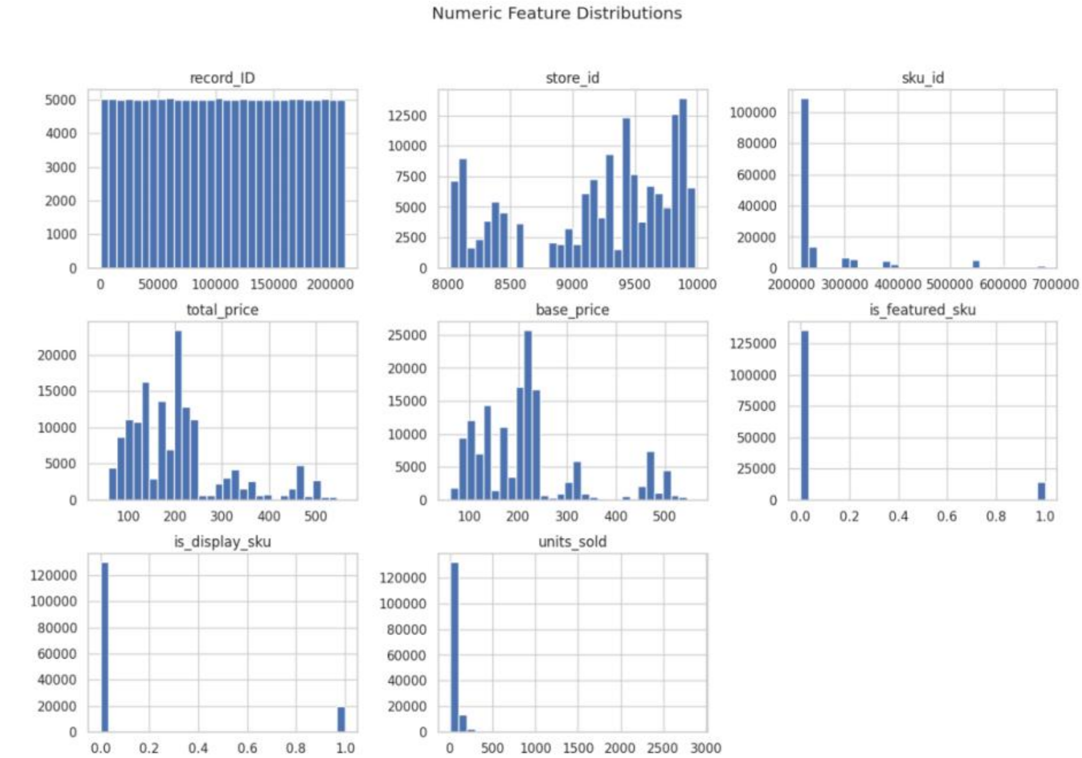

Outlier Detection in total_price
100
200
300
400
500
total_price

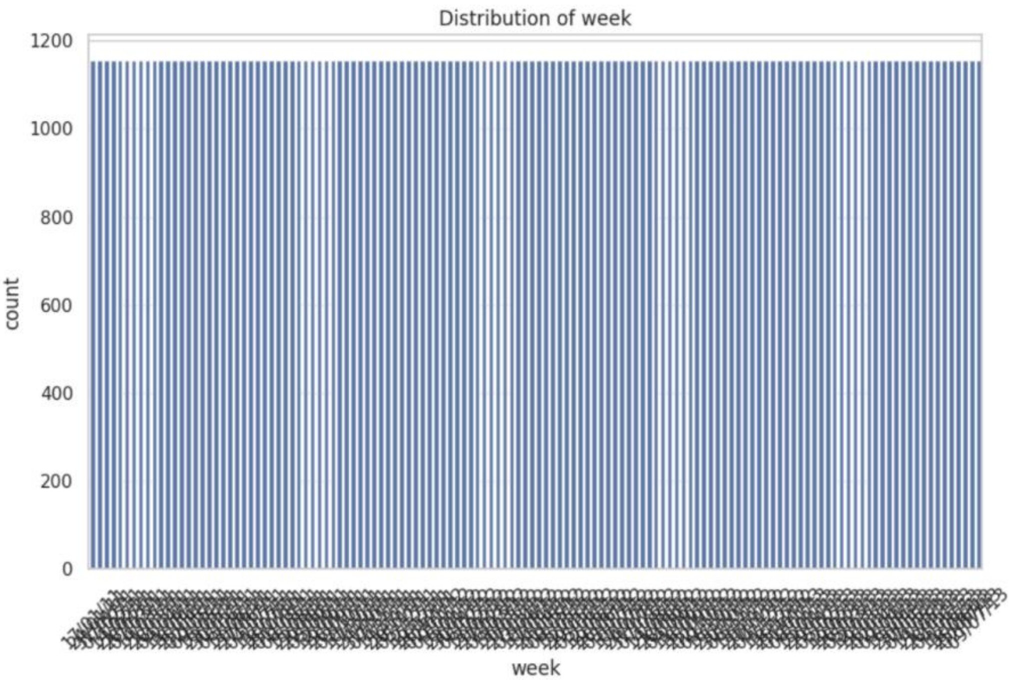

Outlier Detection in base_price
100
200
300
400
500
base_price

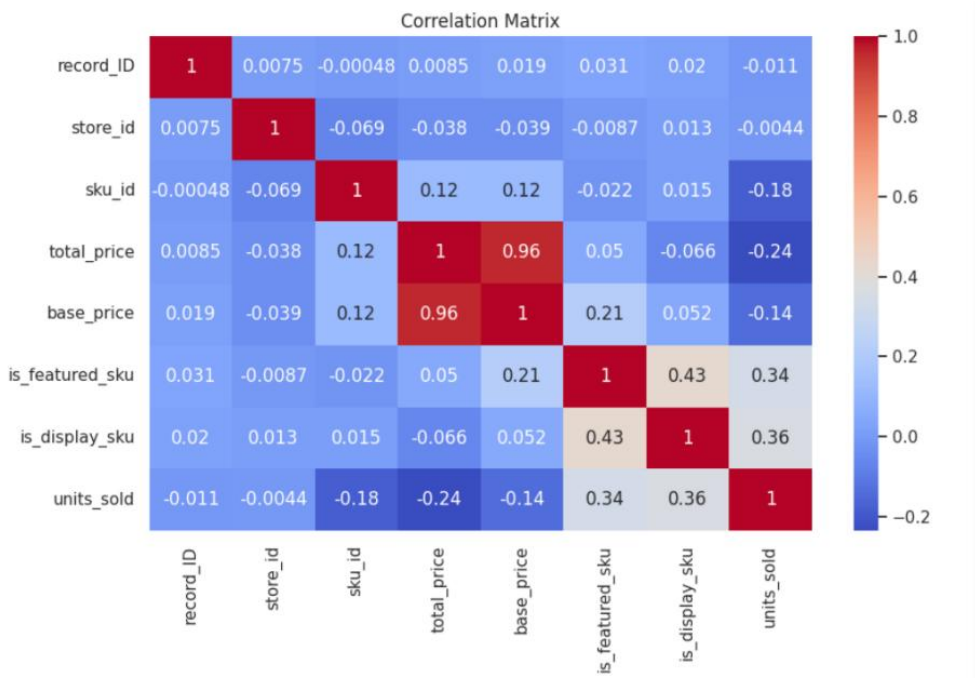

Outlier Detection in is_featured_sku
0.0
0.2
0.4
0.6
0.8
1.0
is_featured_sku


Outlier Detection in is_display_sku
0.0
0.2
0.4
0.6
0.8
1.0
is_display_sku


Outlier Detection in units_sold
0
500
1000
1500
2000
2500
3000
units_sold


Numeric Feature Distributions
record_ID
store_id
sku_id
total_price
base_price
is_featured_sku
is_display_sku
units_sold


Distribution of week
count
week



Preprocessing Steps:

1. **Missing Value Handling:** One missing value in `total_price` was removed.
2. **Data Type Conversion:** `week` converted to datetime for time-based grouping.
3. **Sorting:** Sorted by `store_id`, `sku_id`, and `week` to prepare for time-series operations.
4. **Lag Features:** Created 7 lag features — `day_1` to `day_7` — using `.shift()`.
5. **Rolling/Expanding Features:**
   a) `rolling_mean_3`: average of the last 3 days
   b) `expanding_mean`: cumulative historical mean
6. **Interaction Features:**
   a) `lag1_lag2_interaction`: product of day_1 and day_2
   b) `lag1_plus_lag2`: sum of day_1 and day_2
7. **Target Encoding:** Applied to `store_id` and `sku_id` based on mean `units_sold`
8. **Dropping Columns:** Dropped ID columns and `week` after encoding and transformation.
9. **Null Cleanup:** Rows with NaN values (introduced during shifting) were removed.
10. **Feature Matrix:** Final feature matrix had 17 columns used for modeling.

Train-Test Split:

Used 80% for training and 20% for testing using `train_test_split()`

Final Data Shapes:

X_train: (113,651 rows, 17 features)

X_test: (28,413 rows, 17 features)

This preprocessing pipeline enabled the model to learn from recent trends, historical averages, and encoded relationships within the data.