In [3]:
```python
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from xgboost import XGBRegressor
```

In [4]:
```python
external_df= pd.read_csv(r"C:\Users\ameys\Downloads\walmart_dynamic_pricing_data\external_factors.csv")
external_df
```

Out[4]:

	Date	Weather	Event	CompetitorAveragePrice
0	2024-07-01	Cloudy	NaN	1.86
1	2024-07-02	Cloudy	NaN	2.99
2	2024-07-03	Cloudy	NaN	1.50
3	2024-07-04	Cloudy	NaN	3.37
4	2024-07-05	Rain	NaN	3.03
...	...	...	...	...
360	2025-06-26	Cloudy	NaN	2.03
361	2025-06-27	Cloudy	Festival	2.37
362	2025-06-28	Snow	Festival	3.43
363	2025-06-29	Rain	NaN	1.88
364	2025-06-30	Cloudy	NaN	2.62

365 rows × 4 columns

In [9]:
```python
sales_df= pd.read_csv(r"C:\Users\ameys\Downloads\walmart_dynamic_pricing_data\historical_sales.csv")
sales_df
```

Out[9]:

	Date	ProductName	Category	SalesVolume	PricePerUnit
0	2024-07-01	Apple	Fruit	87	3.76
1	2024-07-01	Banana	Fruit	91	4.82
2	2024-07-01	Carrot	Vegetable	70	4.16
3	2024-07-02	Apple	Fruit	49	3.03
4	2024-07-02	Banana	Fruit	33	3.57
...	...	...	...	...	...
1090	2025-06-29	Banana	Fruit	33	3.73
1091	2025-06-29	Carrot	Vegetable	55	2.59
1092	2025-06-30	Apple	Fruit	51	2.20
1093	2025-06-30	Banana	Fruit	73	4.59
1094	2025-06-30	Carrot	Vegetable	65	4.95

1095 rows × 5 columns

In [11]:
```python
inventory_df= pd.read_csv(r"C:\Users\ameys\Downloads\walmart_dynamic_pricing_data\inventory_data.csv")
inventory_df
```

Out[11]:

	ProductName	Category	ExpiryDate	CurrentStockLevel	RestockLevel	ReorderQuantity
0	Apple	Fruit	2024-12-31	84	40	65
1	Banana	Fruit	2024-12-31	182	23	77
2	Carrot	Vegetable	2024-12-31	194	34	54

In [13]:
```python
external_df.isnull().sum()
```

Out[13]:
```
Date                      0
Weather                   0
Event                   257
CompetitorAveragePrice    0
dtype: int64
```

In [15]:
```python
# Sab object columns me NaN ko 'Unknown' se fill karna
external_df = external_df.fillna('Regular Day')
external_df
```

Out[15]:

	Date	Weather	Event	CompetitorAveragePrice
0	2024-07-01	Cloudy	Regular Day	1.86
1	2024-07-02	Cloudy	Regular Day	2.99
2	2024-07-03	Cloudy	Regular Day	1.50
3	2024-07-04	Cloudy	Regular Day	3.37
4	2024-07-05	Rain	Regular Day	3.03
...	...	...	...	...
360	2025-06-26	Cloudy	Regular Day	2.03
361	2025-06-27	Cloudy	Festival	2.37
362	2025-06-28	Snow	Festival	3.43
363	2025-06-29	Rain	Regular Day	1.88
364	2025-06-30	Cloudy	Regular Day	2.62

365 rows × 4 columns

In [17]:
```python
sales_df.isnull().sum()
```

Out[17]:
```
Date            0
ProductName     0
Category        0
SalesVolume     0
PricePerUnit    0
dtype: int64
```

In [19]:
```python
inventory_df.isnull().sum()
```

Out[19]:
```
ProductName          0
Category             0
ExpiryDate           0
CurrentStockLevel    0
RestockLevel         0
ReorderQuantity      0
dtype: int64
```

In [21]:
```python
external_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Date                    365 non-null    object
 1   Weather                 365 non-null    object
 2   Event                   365 non-null    object
 3   CompetitorAveragePrice  365 non-null    float64
dtypes: float64(1), object(3)
memory usage: 11.5+ KB
```

In [23]:
```python
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1095 entries, 0 to 1094
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          1095 non-null   object
 1   ProductName   1095 non-null   object
 2   Category      1095 non-null   object
 3   SalesVolume   1095 non-null   int64
 4   PricePerUnit  1095 non-null   float64
dtypes: float64(1), int64(1), object(3)
memory usage: 42.9+ KB
```

In [25]:
```python
inventory_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ProductName        3 non-null      object
 1   Category           3 non-null      object
 2   ExpiryDate         3 non-null      object
 3   CurrentStockLevel  3 non-null      int64
 4   RestockLevel       3 non-null      int64
 5   ReorderQuantity    3 non-null      int64
dtypes: int64(3), object(3)
memory usage: 276.0+ bytes
```

In [ ]:

In [28]:
```python
# Date columns ko datetime me convert karo
sales_df['Date'] = pd.to_datetime(sales_df['Date'])
inventory_df['ExpiryDate'] = pd.to_datetime(inventory_df['ExpiryDate'])
external_df['Date'] = pd.to_datetime(external_df['Date'])

# Check data types
sales_df.dtypes
```

Out[28]:
```
Date            datetime64[ns]
ProductName             object
Category                object
SalesVolume              int64
PricePerUnit           float64
dtype: object
```

In [ ]:

In [31]:
```python
# merging
# Merge sales + inventory on ProductName and Category
merged_df = pd.merge(
    sales_df,
    inventory_df,
    on=['ProductName', 'Category'],
    how='left'
)

# Merge with external factors on Date
merged_df = pd.merge(
    merged_df,
    external_df,
    on='Date',
    how='left'
)

# Preview final merged data
print(merged_df.head())
```

```
        Date ProductName   Category  SalesVolume  PricePerUnit ExpiryDate  \
0 2024-07-01       Apple      Fruit           87          3.76 2024-12-31
1 2024-07-01      Banana      Fruit           91          4.82 2024-12-31
2 2024-07-01      Carrot  Vegetable           70          4.16 2024-12-31
3 2024-07-02       Apple      Fruit           49          3.03 2024-12-31
4 2024-07-02      Banana      Fruit           33          3.57 2024-12-31

   CurrentStockLevel  RestockLevel  ReorderQuantity Weather        Event  \
0                 84            40               65  Cloudy  Regular Day
1                182            23               77  Cloudy  Regular Day
2                194            34               54  Cloudy  Regular Day
3                 84            40               65  Cloudy  Regular Day
4                182            23               77  Cloudy  Regular Day

   CompetitorAveragePrice
0                    1.86
1                    1.86
2                    1.86
3                    2.99
4                    2.99
```

In [ ]:

In [34]:
```python
# Feature Engineering
```

In [ ]:

In [37]:
```python
# Sort for lag & rolling
merged_df = merged_df.sort_values(by=['ProductName', 'Date'])

# Lag & Rolling
merged_df['SalesVolume_Lag1'] = merged_df.groupby('ProductName')['SalesVolume'].shift(1)
merged_df['SalesVolume_Lag7'] = merged_df.groupby('ProductName')['SalesVolume'].shift(7)
merged_df['SalesVolume_Rolling7'] = merged_df.groupby('ProductName')['SalesVolume'].shift(1).rolling(7).mean()

# Days to Expiry
merged_df['DaysToExpiry'] = (merged_df['ExpiryDate'] - merged_df['Date']).dt.days

# Low stock flag
merged_df['IsLowStock'] = (merged_df['CurrentStockLevel'] < 10).astype(int)

# Price diff with competitor
merged_df['PriceDiff_Competitor'] = merged_df['PricePerUnit'] - merged_df['CompetitorAveragePrice']

# Event & Weather
merged_df['HasEvent'] = merged_df['Event'].notnull().astype(int)
merged_df['IsBadWeather'] = merged_df['Weather'].apply(lambda x: 1 if x in ['Rain', 'Snow'] else 0)

# Date parts
merged_df['DayOfWeek'] = merged_df['Date'].dt.dayofweek
merged_df['IsWeekend'] = (merged_df['DayOfWeek'] >= 5).astype(int)
merged_df['Month'] = merged_df['Date'].dt.month

# One-hot encode Category
merged_df = pd.get_dummies(merged_df, columns=['Category'], drop_first=True)

# Fill missing
merged_df.fillna(0, inplace=True)

print(merged_df.head())
```

```
         Date ProductName  SalesVolume  PricePerUnit ExpiryDate  \
0  2024-07-01       Apple           87          3.76 2024-12-31
3  2024-07-02       Apple           49          3.03 2024-12-31
6  2024-07-03       Apple           52          2.61 2024-12-31
9  2024-07-04       Apple           84          3.99 2024-12-31
12 2024-07-05       Apple           79          3.71 2024-12-31

    CurrentStockLevel  RestockLevel  ReorderQuantity Weather        Event  \
0                  84            40               65  Cloudy  Regular Day
3                  84            40               65  Cloudy  Regular Day
6                  84            40               65  Cloudy  Regular Day
9                  84            40               65  Cloudy  Regular Day
12                 84            40               65    Rain  Regular Day

    ...  SalesVolume_Rolling7  DaysToExpiry  IsLowStock  PriceDiff_Competitor  \
0   ...                   0.0           183           0                  1.90
3   ...                   0.0           182           0                  0.04
6   ...                   0.0           181           0                  1.11
9   ...                   0.0           180           0                  0.62
12  ...                   0.0           179           0                  0.68

    HasEvent  IsBadWeather  DayOfWeek  IsWeekend  Month  Category_Vegetable
0          1             0          0          0      7               False
3          1             0          1          0      7               False
6          1             0          2          0      7               False
9          1             0          3          0      7               False
12         1             1          4          0      7               False

[5 rows x 23 columns]
```

In [ ]:

In [40]:
```python
# Spliting x and y
```

In [42]:
```python
# Final features
feature_cols = [
    'PricePerUnit',
    'SalesVolume_Lag1',
    'SalesVolume_Lag7',
    'SalesVolume_Rolling7',
    'CurrentStockLevel',
    'IsLowStock',
    'DaysToExpiry',
    'PriceDiff_Competitor',
    'HasEvent',
    'IsBadWeather',
    'DayOfWeek',
    'IsWeekend',
    'Month'
] + [col for col in merged_df.columns if col.startswith('Category_')]

# X and y
X = merged_df[feature_cols]
y = merged_df['SalesVolume']

# Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

print("Train shape:", X_train.shape)
```

```
Train shape: (876, 14)
```

In [ ]:

In [45]:
```python
# XGBoost
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1]
}
```

In [ ]:

In [ ]:

In [ ]:

In [317…
```python
# training Ml model
```

In [51]:
```python
from sklearn.model_selection import GridSearchCV


param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
}

grid = GridSearchCV(
    RandomForestRegressor(random_state=42),
    param_grid,
    cv=3,
    scoring='neg_root_mean_squared_error',
    n_jobs=-1
)

grid.fit(X_train, y_train)

print("Best params:", grid.best_params_)
print("Best CV RMSE:", -grid.best_score_)
```

```
Best params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 200}
Best CV RMSE: 23.584649891247917
```

In [ ]:

In [ ]:

In [54]:
```python
#model Evalution
from sklearn.model_selection import cross_val_score

best_rf = grid.best_estimator_

y_pred = best_rf.predict(X_test)

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print("Final Test RMSE:", rmse)
print("Final Test R²:", r2)

# Cross-validation check
scores = cross_val_score(best_rf, X, y, cv=5, scoring='neg_root_mean_squared_error')
print("5-Fold CV RMSE:", -scores.mean())
```

```
C:\Users\ameys\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the function'root_mean_squar
ed_error'.
  warnings.warn(
Final Test RMSE: 23.526284096957358
Final Test R²: -0.06260334035269355
5-Fold CV RMSE: 23.375989161768683
```

In [ ]:

In [332…
```python
# saving Predictions
```

In [59]:
```python
results_df = X_test.copy()
results_df['Actual_SalesVolume'] = y_test.values
results_df['Predicted_SalesVolume'] = y_pred

results_df.to_csv('xgboost_dynamic_pricing_predictions.csv', index=False)
print("Saved to 'xgboost_dynamic_pricing_predictions.csv'")
```

```
Saved to "xgboost_dynamic_pricing_predictions.csv"
```

In [ ]: