Pricing Precision

Maximizing Revenue Through Price Elasticity

Amit Chandhok

June 5, 2025

Agenda

1. Business Context

2. Exploratory Insights

3. Modelling Framework

4. Demand Forecasting

5. Implementation Roadmap

6. Conclusion

7. Q&A

Appendix: Technical backup if required

Business Context

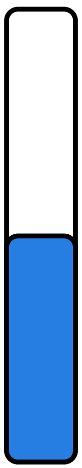
Problem Statement

Online retail’s inaugural pricing strategy operated in the dark with no historical data to guide decisions, resulting in revenue leakage and missed strategic opportunities

Objective

Increase overall revenue with a new pricing plan backed by predictive models

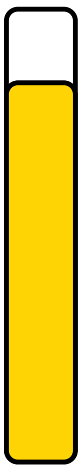
Prospective Growth



$57,468

PRIOR REVENUE

Measured impact of 3 key SKUs from Q1 2010

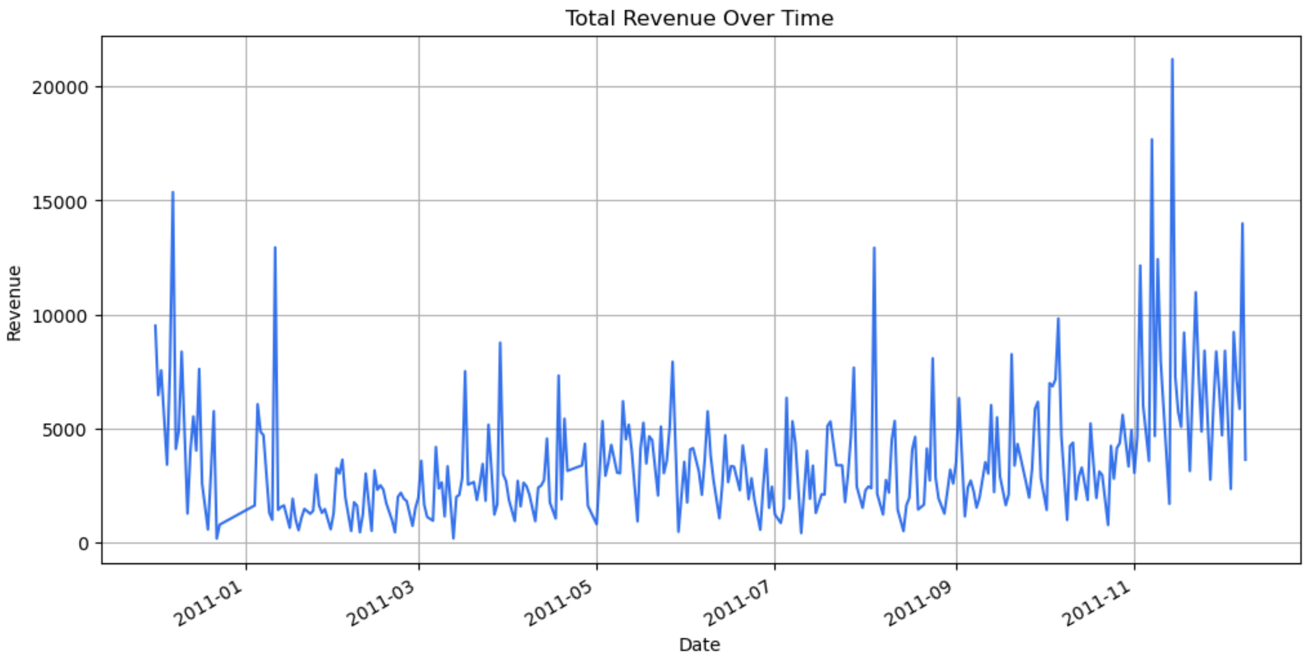


$103,271

FORECASTED REVENUE

Potential 79% lift in Q1 revenue for the same SKUs

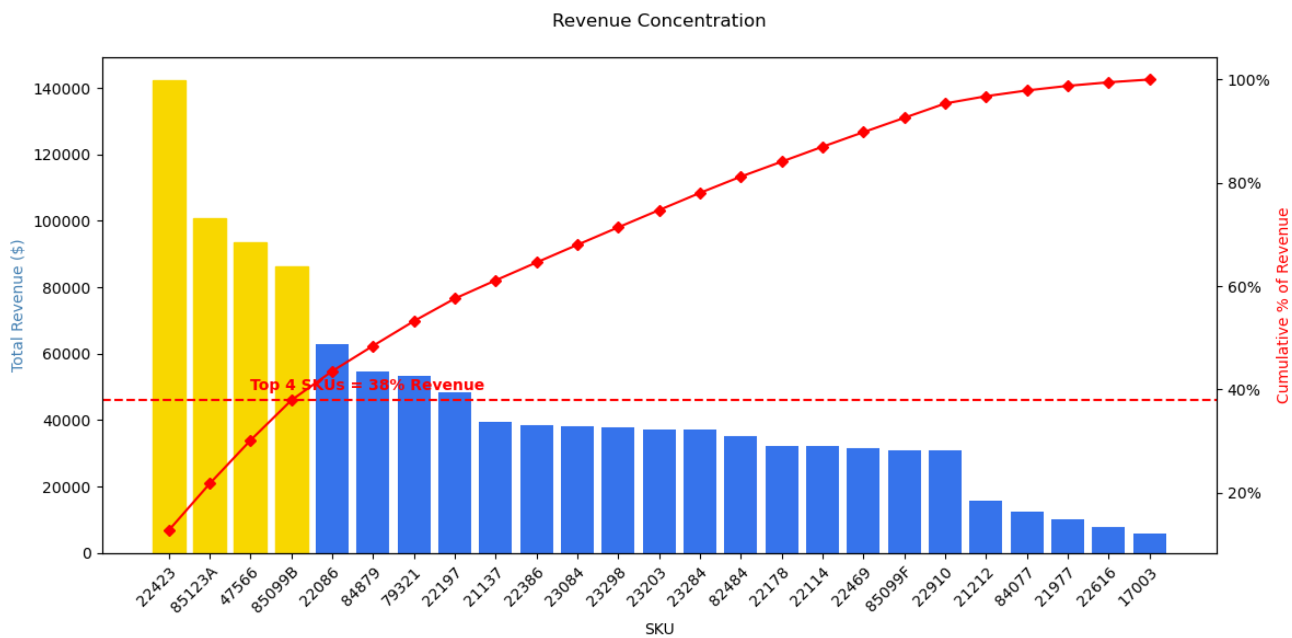
Exploratory Insights



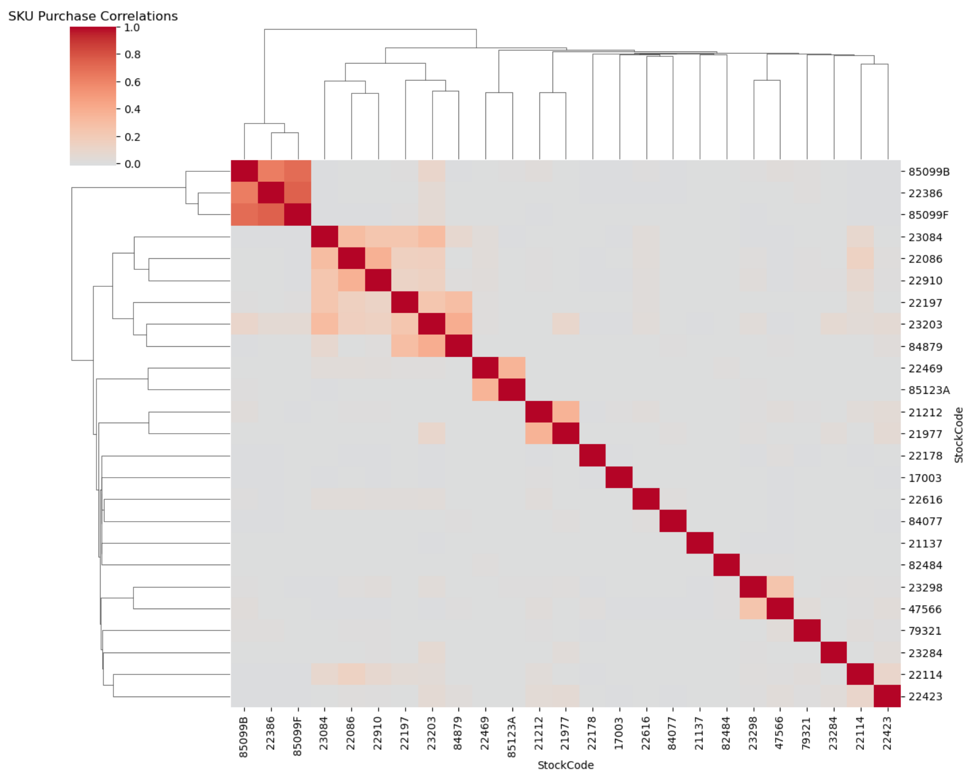
Seasonality is pronounced, with spikes in sales clearly visible during the holiday season, indicating a critical period for promotional efforts

Exploratory Insights

The top 4 SKUs drive approximately 40% of total revenue within the top 25 SKUs, highlighting significant revenue concentration



Exploratory Insights



Significant purchase correlations among SKUs suggest potential for optimized bundling strategies to leverage cross-SKU demand and manage price elasticity effectively

Modelling Framework

Model	Pros	Cons	Log-Log Comparison
Linear Regression	Simple, interpretable	Assumes linearity in levels, poor fit for % based effects	Log-log better captures relative changes
XGBoost	Captures non-linear pricing thresholds and complex interactions	Black box, lacks interpretability of elasticity	Less suited when transparency is required
Panel Regression	Controls for SKU fixed effects and cross-SKU price impacts	May overcomplicate SKU level tactical decisions	More strategic level insights, less actionable for SKU pricing
Log-Log Regression	Models % change in sales vs. % change in price directly, handles heteroskedasticity	Requires log transformation which may be less intuitive at first	Preferred when elasticity is core metric of interest

Modelling Framework

Categorical features

Log log regression applies a higher weight for larger numbers which need to be encoded

Temporal features

Extracted month, day of week and quarter to capture seasonality patterns and shopping behaviour trends

Lag features

Included prior period sales as predictors to capture temporal autocorrelation.

Model Evaluation Summary

How well it fits:

On average, the models explain about **65% of the variation** in sales across SKUs (R^2 = 0.65)

Forecast Accuracy:

On average, predictions are within **47 units** of actual sales (MAE = 47)

Price Impact:

18 out of 25 products show a meaningful link between price and demand (p-value < 0.05)

Results for end users to make decisions

17003

Highlights

- Elasticity: -2.54
- Elasticity p-value < 0.05
- Consistent R^2 scores (~0.65)
- Low MAE scores

Key Takeaways

- Highly price sensitive, a 1% decrease in price results in a 2.54% increase in sales. Consider price-based promotions to lift volume.

22178

Highlights

- Elasticity: 0.01
- Elasticity p-value > 0.05
- High and consistent R^2 (~0.8)
- Elasticity CI Upper: 0.98

Key Takeaways

- Price has no measurable impact. Focus on seasonality or bundling strategies instead.

21977

Highlights

- Elasticity: -1.06
- Elasticity p-value < 0.05
- MAE ~20 units
- Test R^2: 0.17

Key Takeaways

- Price moderately affects demand, but the model generalizability is weak. Use targeted, short term pricing experiments to validate before scaling.

Opportunities to improve our model

Missing features:

1. Competitor pricing changes
2. Inventory Effects
3. Promotion data
4. Improve product categories

Isolated modelling:

1. No cross SKU (compliments/substitutes) effects
2. Integrate a dashboard for interpretability

Limitations & Next Steps

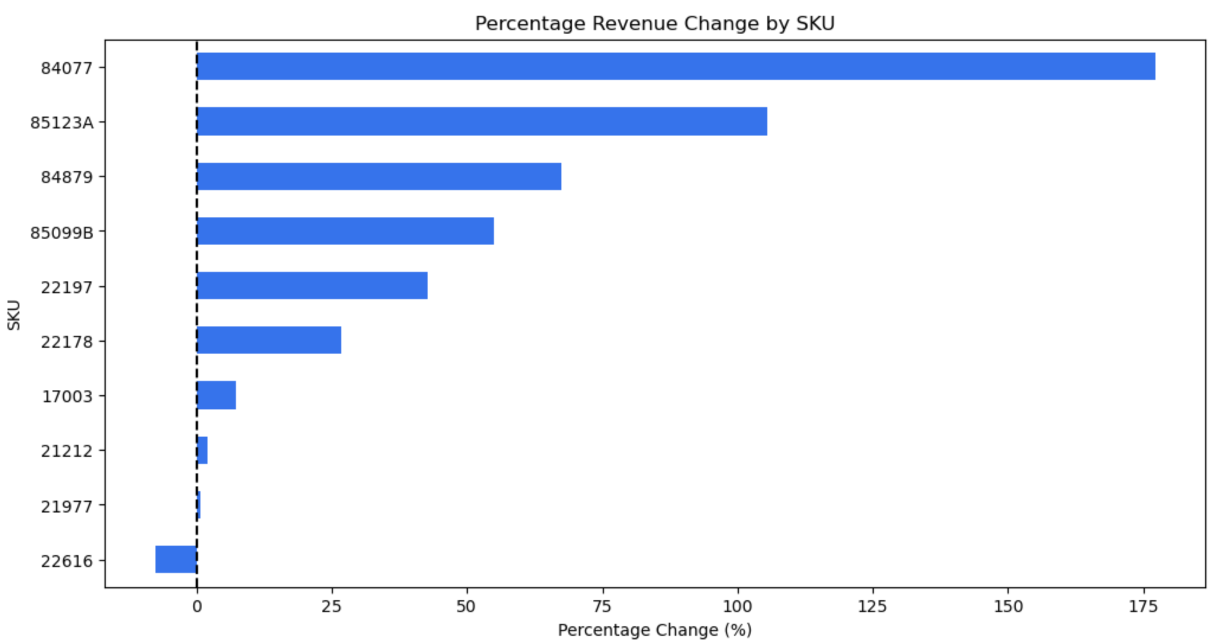
Simplified time effects:

1. Elasticity assumed static across time

Short time horizon:

1. Limited test periods reduce generalizability. Incorporate rolling time-based cross validation

Demand Forecasting



- Forecasted demand using prior year sales + SKU level elasticity
- Projected +65% revenue increase, or $76,645 lift
- Largest gains concentrated in high elastic SKUs with price reductions

Recommendation

Follow Pricing Plan

SKUs: 17003, 22197, 85099B, 85123A

Statistically significant elasticity estimates. Clear, consistent demand response to price. Forecasted 79% revenue lift in Q1 under proposed plan.

Monitor & Test

SKUs: 21977, 22616, 84077

Run A/B tests to understand price changes. Lower statistical confidence with varying model performance.

Hold & Reassess

SKUs: 21212, 22178, 84879

Maintain current pricing while collecting more data. Price elasticity not statistically significant and revenue impact is uncertain or negative. Current model lacks predictive power for these SKUs.

Conclusion

Key Findings:

- ~65% of SKUs show statistically significant price sensitivity
- Top SKUs drive majority of projected lift
- Predictive model can explain ~65% of variation in sales

Impact:

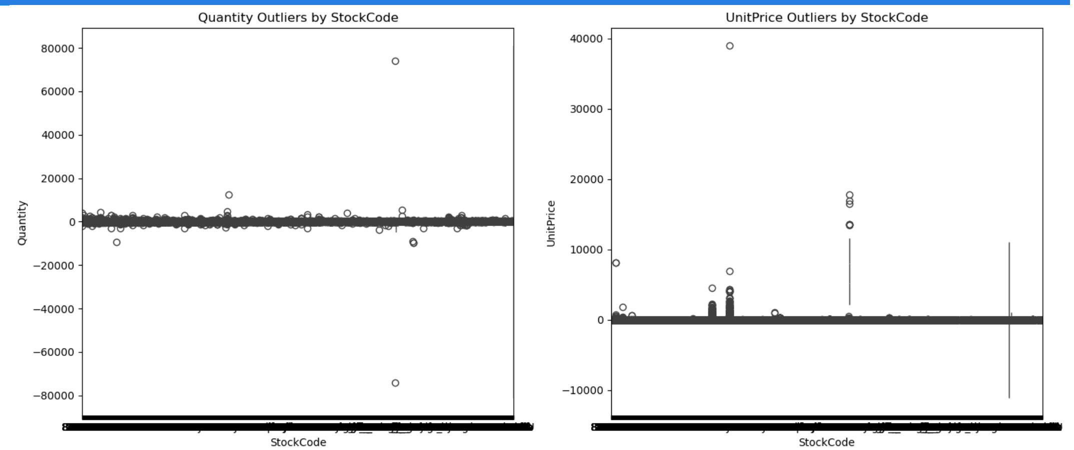
- Proposed pricing plan could yield Q1 lift of 79%

Next Steps:

- Expand model to include promotions, competitor pricing and seasonality

Technical Appendix: Outlier Detection

- Created boxplots for quantity and unit prices to check for outliers.
- SKU 22502 had 2 entries that represented 60 units rather than 1. I divided the unit price by 60 to get the actual unit price.
- This analysis removed SKU 22502 from the top 20 SKUs by revenue since it was reliant on an outlier.
- Outlier shown for quantity represents a cancellation, not a true outlier.




```python
# Outlier 22502 detected where one line item had 60 units, skewing total revenue results
per_unit_price = (df[(df['StockCode'] == 22502) &
                  (df['Description'] == 'PICNIC BASKET WICKER 60 PIECES')]['UnitPrice']/60).values[0]

# Update the UnitPrice for all matching rows
df.loc[(df['StockCode'] == 22502) &
       (df['Description'] == 'PICNIC BASKET WICKER 60 PIECES'),
       'UnitPrice'] = per_unit_price
```

Technical Appendix: Data Cleaning

- Normalized StockCodes to all be in upper case and group codes where suffix was “c” instead of “C.”
- Filtered out all StockCodes that were not SKUs, not from United Kingdom and not in the top 20 by revenue. Included top 10 by quantity since promo plan had StockCodes ranked by quantity.
- Mapped descriptions to match their most common description to help with forming product categories.
- Aggregated cancellations into original invoice and filtered quantities and prices below 0 to help with log transformations.

```python
def data_cleaning(df):

    # Filter to see product SKU only
    df['StockCode'] = df['StockCode'].astype(str).str.upper() # Some SKUs have mixed case for the same product
    df = df[df['StockCode'].astype(str).str.match(r'^([0-9]|D)[0-9C]')] # Only looking at product SKUs, excluding DOT, Amazon, etc

    # Calculate total revenue and filter top 20 SKUs in UK
    df['TotalRevenue'] = df['Quantity'] * df['UnitPrice']
    top20 = df[df['Country'] == 'United Kingdom'].groupby('StockCode')['TotalRevenue'].sum().nlargest(20).index
    top20_quantity = df[df['Country'] == 'United Kingdom'].groupby('StockCode')['Quantity'].sum().nlargest(10).index

    combined_skus = list(top20) + list(top20_quantity)

    selected_skus = list(set(combined_skus))

    df = df[(df['Country'] == 'United Kingdom') & (df['StockCode'].isin(selected_skus))]

    # Create a mapping dictionary for Descriptions that have multiple formats
    desc_mapping = df.groupby('StockCode')['Description'].apply(lambda x: x.mode()[0]).to_dict()

    # Apply the mapping to standardize descriptions
    df['Description'] = df['StockCode'].map(desc_mapping)

    # Clean InvoiceNo and aggregate data
    df['InvoiceNo'] = df['InvoiceNo'].astype(str).str.lstrip('C').astype(int)
    df = df.groupby(['InvoiceNo','StockCode','Description']).agg(
        {'Quantity': 'sum',
        'InvoiceDate': 'min',
        'UnitPrice': 'max',
        'TotalRevenue': 'sum'}
    ).sort_values(by='InvoiceDate').reset_index()

    # Convert InvoiceDate feature and filter positive values
    df['InvoiceDate'] = df['InvoiceDate'].dt.to_period('D')
    df = df[(df['Quantity'] > 0) & (df['UnitPrice'] > 0)]

    return df
```

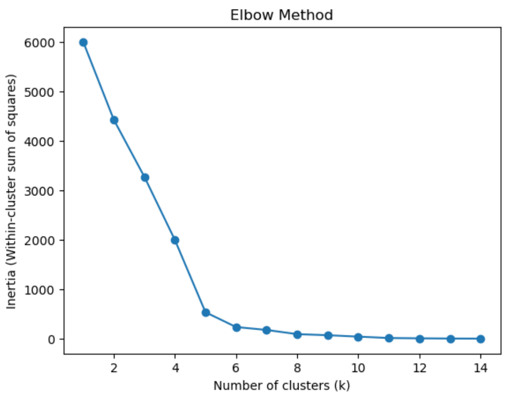
Technical Appendix: Product Categories

- Utilized TF IDF to convert words to numbers based on frequency to highlight “important” words like “Small.”
- Truncated the number of important words to 4 to help summarize the description.
- Utilized the elbow method that helps understand the optimal number of groups where groups are not overfitting and yet have similarities.
- Built 5 distinct product categories using K Means clustering.

```
# Product clustering based on descriptions
tfidf = TfidfVectorizer(stop_words='english')
desc_tfidf = tfidf.fit_transform(train_df['Description'].astype(str))

svd = TruncatedSVD(n_components=4, random_state=42)
desc_svd = svd.fit_transform(desc_tfidf)
```

```
kmeans = KMeans(n_clusters=5, random_state=42)
train_df['ProductCategory'] = kmeans.fit_predict(desc_svd)
```



Technical Appendix: EDA

- No clear linear pattern
- Long tail stretching right and up.
- Setup for a log transformation