MGSC 670: Revenue Management

Group Assignment 1

Presented To

Prof. Maxime Cohen

Team

Arham Anwar (261137773)
Tashfeen Ahmed (26115667)
Yifan Lu (261144056)

Desautels Faculty of Management
McGill University

1 Introduction

The most important realization that arises in today's competitive retail environment is the implementation of the best pricing strategy for maximizing revenue and minimizing inventory-carrying costs. This report looks at three different methodologies that prescribe the best pricing strategies for a retail product over a 15-week selling period. Each of the methodologies is crafted to deliver optimal profit and inventory turnover, but underneath they are founded on distinctly different assumptions and considerations on customer behavior and marketplace operations.

- **Human Intuition Matrix Method**: Utilizes experienced managers' tacit knowledge and historical data to populate a dynamic decision matrix, which visualizes sales outcomes under different pricing scenarios. This method is quick to adapt but can suffer from subjectivity, leading to inconsistent pricing unless controlled.
- **Linear Optimization (Linear Demand Variation)**: Employs a linear programming model assuming a linear relationship between price and market demand. This method provides a systematic, unbiased approach to setting prices for maximum revenue, suited for markets with predictable consumer price reactions.
- **Non-linear Optimization (Non-linear Demand Variation)**: Advances beyond linear models by incorporating complex, non-linear consumer responses to pricing, considering factors like price sensitivity and psychological pricing, honoring individual weekly demand distributions for each price and price jump! This more realistic approach aims to optimize revenue while closely mirroring actual consumer behavior across various market segments.

In section 2 we talk about the simulation setup for testing the models. In section 3 we talk about our three approaches in detail, building from the most naive human intuition to advanced application of human intuition converted to an linearized optimization problem.

2 Data Collection

The data collection for this experiment was achieved through automation but by a unique way of automated web scraping designed specifically to collect data from a simulated retail environment as provided by retailer markdown game. This was done through deployment of the Selenium WebDriver which is a robust browser controlling tool running off a program. By automating the interactions in the web interface of the retail game, we have collected sales, pricing, and inventory-level data under different pricing strategies over a 15-week period.

2.1 Set Up and Configuration

The setup began with having the Selenium WebDriver configured to run in a headless mode. This means it runs without the graphical user interface. This way, it consumes the least resources hence saving time in collecting data. The following are the configurations set for the WebDriver:

- Initial Chrome WebDriver with options to disable the GPU and set up the Window size to be 1920 x 1200. This is to aid data collection uniformity.
- The browser session to be initialized by navigating to the URL hosting the game where the pricing simulation is done.

2.2 Data Extraction Methodology

The primary functionality of our data extraction script was to simulate different pricing decisions in the game and to capture the outcomes. We interacted with the web page elements to maintain the prices or raise them using discounts of 10%, 20%, and 40%. In this regard, we had to:

- Locate the page and its elements and interact with it through HTML IDs which in this case are the price maintaining buttons and the relaunch button that clears the previous simulation run and does a new run.
- A pre-defined strategy was applied where the price was kept constant for the first three weeks, followed by arbitrary application of discounts, as may be the case in practice should retailers want to change these prices given changing market conditions.

2.3 Automated Game Play and Data Extraction

The Selenium script automated several runs of the game (200 runs in all), each time with a randomly generated pricing strategy. For each week of the 15-week period being simulated, it extracted the following data:

- The Week Number: The week number of the simulation at the current time.
- Price: the price point set for that week
- Sales: How many units were sold at the price?
- Remaining Inventory: How much inventory is left at the end of the week?
- Total Revenue and Optimal Revenue: Computed to compare how well the applied pricing strategy does compared to a theoretically optimal pricing strategy

This data was logged in a structured form and compiled in a dataset, which recorded the consumer response to the varied pricing strategies over time. Data from each of the 200 runs were aggregated to a master list, converted into a DataFrame, and written out to a CSV file.

3 Approach

3.1 Approach - 1: Human Intuition via Matrix Method

The first approach in the study is the use of an algorithm availed through fine-tuned data that were gathered very carefully by simulating through some very many trials. It majorly

focuses on the situations that installed the least amount of variance between expected and actual outcomes—under 5%. The target here is to make the decision-making process far more superior and granular. We developed the same using a matrix method. In this approach we find average weakly uplifts for each price as shown in table 2, and price jumps as shown in table 4. Using these matrices, and guardrail lower and upper limits, we can play with the website to achieve 4-11% difference from perfect forecast.

3.1.1 Algorithm Overview

The algorithm analyses the sales data of the weeks leading up to a possible price change. This analysis is run under different price scenarios for each week of median sales figures, and a set of "trigger points" for each of the six possible price changes in a given week was set up. Each pathway had a predetermined median sales figure that would trigger the activation of a price change. If sales went below this median threshold for that pathway, a price change was triggered for the next week in the direction and degree of the respective pathway; otherwise, no price change was triggered.

3.1.2 Handling Sparse Data

For a couple of the price changes—especially \$60 to \$36 and \$48 to \$36—there was very sparse data for the pathway that would have adequately established trigger points. Since there were very few data points to make the trigger points more certain, the thresholds for these pathways were set extremely low, as not to trigger an early or otherwise unjustified price change. This was important, as another analysis under the earlier trigger points for this pathway showed that this led to suboptimal or non-profitable outcomes.

3.2 Approach 2: Linear Optimization (Assuming Linear Weekly Demand Variation for Each Price)

The second approach employed linear optimization methods to devise an optimal pricing strategy over a 15-week period, assuming that the weekly demand variance at each price level is linear. Advanced linear programming models provided through the Gurobi Optimizer platform were implemented. Regression analysis was subsequently utilized to set decision variables and constraints.

3.2.1 Model Development

First and foremost, the model integrated key parameters obtained from a linear regression analysis. This analysis determined the relationship between sales and variables such as price, week number, and the interaction between price and week. According to the regression model, sales declined with an increase in price and as the selling season progressed, albeit at a diminishing rate due to the interaction term.

3.2.2 Setting Up the Optimization Model

The Gurobi model was set up with the objective of maximizing total revenue. This objective was captured into a set of decision variables determining the price level for each week. The variables were bound by available inventory and by the requirement that prices must be lowered or held constant, aligning with common retail markdown strategies:

- Decision Variables: A binary decision variable for each price and week, indicating if a given price was active during that week.
- Objective Function: The objective was to maximize the sum of weekly revenues, which are the products of price, projected sales, and a cumulative modifier over price and week.
- Constraints:
  - Exactly one price must be selected each week.
  - The total sales in a week must not exceed the initial inventory.
  - Prices can only decrease or remain constant as weeks progress, starting at the highest price in the first week.

3.2.3 Mathematical Formulation

Decision Variables:

- Let $X_{i,j}$ be a binary variable indicating whether price p_i is chosen in week j.
- $X_{i,j} \in \{0, 1\} \quad \forall i \in \{1, 2, 3, 4\}, j \in \{1, 2, \ldots, 15\}$
- Let S be the total sales.

Objective Function:

Maximize total revenue: $$\max \left(\sum_{i=1}^{4} \sum_{j=1}^{15} p_i \cdot X_{i,j} \cdot (\beta_0 + \beta_1 p_i + (\beta_2 + \beta_3 p_i) \cdot (j + 1)) \right)$$

Where:

- p_i is the price for index i
- β_0 is the intercept
- β_1 is the coefficient for price
- β_2 is the coefficient for week
- β_3 is the coefficient for the interaction between price and week

Constraints:

1. Each week must have exactly one price selected:

$$\sum_{i=1}^{4} X_{i,j} = 1 \quad \forall j \in \{1, 2, \ldots, 15\}$$

2. Total sales should not exceed the initial inventory:

$$S = \sum_{i=1}^{4} \sum_{j=1}^{15} X_{i,j} \cdot (\beta_0 + \beta_1 p_i + (\beta_2 + \beta_3 p_i) \cdot (j+1)) \leq 2000$$

3.2.4 OLS Regression Results

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.439
Model:	OLS	Adj. R-squared:	0.438
Method:	Least Squares	F-statistic:	781.4
Date:	Sun, 19 May 2024	Prob (F-statistic):	0.00
Time:	18:13:55	Log-Likelihood:	-17408.
No. Observations:	3000	AIC:	3.482e+04
Df Residuals:	2996	BIC:	3.485e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	995.5491	18.744	53.114	0.000	958.797	1032.301
Price	-14.9418	0.352	-42.487	0.000	-15.631	-14.252
Week	-67.1839	2.567	-26.169	0.000	-72.218	-62.150
Price:Week	1.0545	0.064	16.539	0.000	0.929	1.180

Omnibus:	153.506	Durbin-Watson:	1.138
Prob(Omnibus):	0.000	Jarque-Bera (JB):	211.497
Skew:	0.477	Prob(JB):	1.19e-46
Kurtosis:	3.884	Cond. No.	4.46e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.46e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Figure 1: OLS Regression Results

3.3 Approach - 3: Linear Optimization (Assuming Non-Linear Weekly Demand Variation for Each Price)

This approach combines nonlinear decision boundaries of the "matrix method" with the robustness of linear optimization (with linear weekly demand variation). Each price's weekly changes are honored individually, as can be seen from table 2, where each week, and each price has its own mean and standard deviation calculated from given excel samples. The code for this approach is available in file "Assignment-1-Linear-Programming-Complex-Model.ipynb"

3.3.1 Decision Boundaries

First, we computed historical weekly increment averages individually for all prices. We limit this calculation only to the dataset directly given to us to ensure a fair analysis. Out-of-stock cases, where demand overshoots available stock, were adjusted by assuming demand to be equal to that of the penultimate week if the price is the same.

3.3.2 Optimization Problem Setup

We treat each week as a separate optimization problem, with a total of 14 problems, and address the pricing strategy through a series of binary decision variables.

Decision Variables

- $X_{i,k}$: Binary decision variable indicating whether price i is selected in week k, where $i = 0, 1, 2, 3$ (representing prices 60, 54, 48, 36, respectively) and $k = 0, \ldots, 13$.

Supporting Variables

- **Penultimate_Sales**: Captures the actual sales data from the previous week $(k-1)$.
- **Uplift_predictions**: Contains the uplift percentages for each price, adjusted based on historical data.

Objective Function

$$\text{Maximize} \sum_{i=0}^{3} X_i \cdot \text{Price\_List}[i] \cdot \text{Sales\_Predictions}[i] \tag{1}$$

Constraints

- **Single Price Constraint**: Ensures that exactly one price is chosen each week, i.e., $\sum_{i=0}^{3} X_i = 1$ for each week k.
- **Definition of Sales Prediction**: Mathematical definition as Penultimate week's sales multiplied by uplift predictions for each price.

$$\text{Sales\_Predictions}_i \leq \text{Penultimate\_Sales} \times \text{Uplift\_predictions}_i \quad \forall i \in \{0, 1, 2, 3\}$$

- **Sales Prediction Constraints**: Sales predictions for each price are capped by product availability and adjusted for historical uplift. This is mathemtically done by taking minimum of predicted demand and available inventory.

$$\sum_{i=0}^{3} \text{Sales\_Predictions}_i \leq \text{Weekly\_Inventory[last]}$$

- **Price Gap Threshold Constraint**: If penultimate week's actual sales was less than 70, then we should markdown. This is mathematically done by setting uplift prediction variable to zero for the same price forcing gurobi to markdown.
- **Good Market Constraint**: If penultimate week's actual sales was more than 100, we stay at the same price, even if there is a significant decline in sales. This is mathematically done by setting prediction uplift value for markdowns to zero.

Please see adaptation of Uplift Predictions in 41. The logic describes how the uplift predictions are dynamically adjusted based on the last price decision and market conditions reflected in penultimate sales.

Optimization Execution The model uses the `m.optimize()` function to find the optimal set of decisions that maximize the objective function while satisfying all constraints. At the end, the forecast is compared with actual sales to see the difference. The actual forecast then becomes the input for next week, and inputs such as uplift values, penultimate values are readjusted for next iteration.

Algorithm 1 Adjusting Uplift Predictions Based on Previous Price Decisions

$Uplift\_predictions \leftarrow Sales\_Deltas.iloc[k-1].values.tolist()$
if $Price\_decisions[-1]/6 == 6$ **then**
    $temporary \leftarrow 0$
else if $Price\_decisions[-1]/6 == 8$ **then**
    $Uplift\_predictions[3] \leftarrow jumps$
else if $Price\_decisions[-1]/6 == 9$ **then**
    $Uplift\_predictions[3] \leftarrow jumps$
    $Uplift\_predictions[2] \leftarrow jumps$
else if $Price\_decisions[-1]/6 == 10$ **then**
    $Uplift\_predictions[3] \leftarrow jumps$
    $Uplift\_predictions[2] \leftarrow jumps$
    $Uplift\_predictions[1] \leftarrow jumps$
end if

procedure Price Gap Threshold Constraint
    if $Penultimate\_Sales \leq lower\_threshold$ **then**
        if $Price\_decisions[-1]/6 == 6$ **then**
            $temporary \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 8$ **then**
            $Uplift\_predictions[2] \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 9$ **then**
            $Uplift\_predictions[1] \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 10$ **then**
            $Uplift\_predictions[0] \leftarrow 0$
        end if
    end if
end procedure

procedure Good Market Constraint
    if $Penultimate\_Sales \geq upper\_threshold$ **then**
        if $Price\_decisions[-1]/6 == 6$ **then**
            $temporary \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 8$ **then**
            $Uplift\_predictions[3] \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 9$ **then**
            $Uplift\_predictions[3] \leftarrow 0$
            $Uplift\_predictions[2] \leftarrow 0$
        else if $Price\_decisions[-1]/6 == 10$ **then**
            $Uplift\_predictions[3] \leftarrow 0$
            $Uplift\_predictions[2] \leftarrow 0$
            $Uplift\_predictions[1] \leftarrow 0$
        end if
    end if
end procedure

Results

This table summarizes the performance of different strategies in terms of their percentage difference from a perfect foresight strategy:

Table 1: Comparison of Different Pricing Approaches

Approach	**Mean (%)**	**Std (%)**	**Confidence Interval LB (%)**	**Confidence Interval UB (%)**
Random (200 runs)	16.27	6.52	15.36	17.18
Simple Linear Optimization (50 runs)	8.05	5.75	6.42	9.68
Human (3 Runs)	4.28	1.45	1.96	6.59
Complex Linear Optimization (5 Runs)	2.27	-	-	-

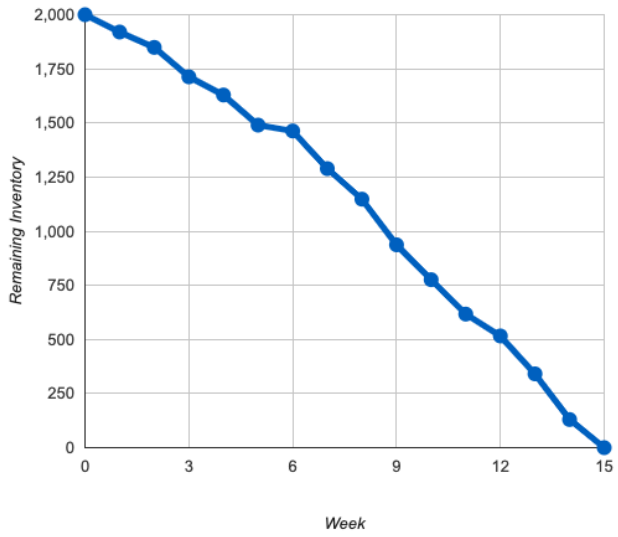
Simple Linear optimization is better than random with a lower mean. Human intuition if performed through extensive grids is just as powerful. The most powerful and robust is the complex linear optimization (Approach - 3) which is consistently able to capture even difficult markets with sudden market collapses, outlier behaviors and more! Let's go through two sample cases to see the power of our complex linear optimization model.

Case 1 as shown in figure 2 shows a market starting with demand 80. Our model motivates the first prediction to be 60 to test waters so it maintains the decision of 60. The actual realised demand of week 2 was 71 which forces our 'low market' constraint to markdown to price of 54. In third week, the demand is 136. The model maintains price of 54 until week 6 where demand hits a crash of 27, where the model keenly picks up and marks down the price to 48 stablizing the demand. The model maintains the price of 48 until the last week, where it forces the price to be 36 to clear the inventory. This results in 1.1% difference from perfect foresight strategy! Please see code file with this run out put saved in addition to other runs.

Similarly, in case 2 figure 3 we encounter a low market where the model nicely picks up the rhythm and achieves a 2.8% difference from perfect forecast. Please see figures ahead.

To replicate the results for your chosen trial, simply launch the game website, and use our model. We will need to update the weekly 'realised sales' after every forecast by adding it to the 'Tempo' list available in the first cell and run code for each week. Limitations of the model currently includes no constraints set for highly complex multiple sudden patterns which the models. To fix this, we can devise additional constraints. As for Future Scope, we wish to encorporate deep learning forecasts including state of the art models to our gurobi models to elevate the robustness of the models.

Week	Price	Sales	Remaining Inventory
1	60	80	1920
2	60	71	1849
3	54	136	1713
4	54	84	1629
5	54	139	1490
6	54	27	1463
7	48	174	1289
8	48	141	1148
9	48	211	937
10	48	161	776
11	48	159	617
12	48	101	516
13	48	175	341
14	48	211	130
15	36	130	0

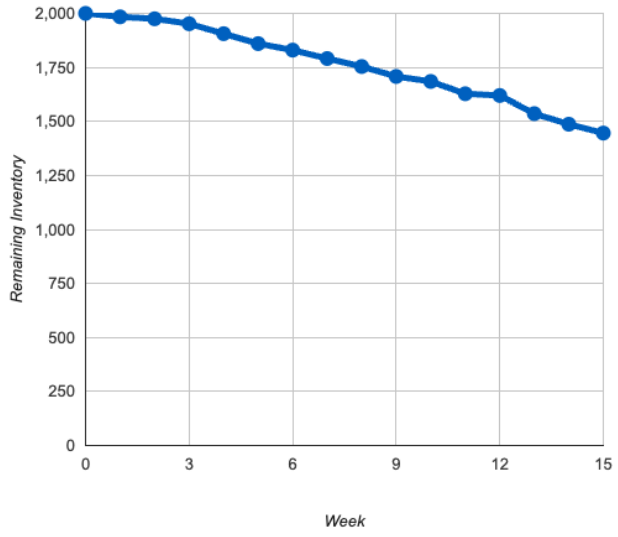


Maintain Price 10% 20% 40%

Your revenue: $98,568, Perfect foresight strategy: $99,649, Difference: 1.1%

Figure 2: Case 1 - Normal Market

Week	Price	Sales	Remaining Inventory
1	60	16	1984
2	60	9	1975
3	54	23	1952
4	48	46	1906
5	36	46	1860
6	36	30	1830
7	36	39	1791
8	36	37	1754
9	36	46	1708
10	36	23	1685
11	36	57	1628
12	36	8	1620
13	36	84	1536
14	36	49	1487
15	36	41	1446



Maintain Price 10% 20% 40%

Your revenue: $57,660, Perfect foresight strategy: $59,323, Difference: 2.8%

Figure 3: Case 2 - Outlier Market with Extremley Low Demand

Appendix

Table 2: Weekly Incremental Averages, Std Deviations based on given data

Week	**Avg of Inc60**	**Avg of Inc54**	**Avg of Inc48**	**Avg of Inc36**	**StdDev of Inc60**	**StdDev of Inc54**	**StdDev of Inc48**	**StdDev of Inc36**
1	-	-	-	-	-	-	-	-
2	13.5%	-	-	-	0.39	-	-	-
3	9.8%	-	-	-	0.52	-	-	-
4	21.3%	-	-	-	0.67	-	-	-
5	1.3%	-	-	-	0.27	-	-	-
6	-8.0%	-	-	-	0.37	-	-	-
7	24.5%	-	-	-	0.50	-	-	-
8	-11.3%	48.1%	-	-	0.51	1.15	-	-
9	-25.3%	11.3%	1.6%	-	0.21	0.25	0.31	-
10	-	-16.5%	-8.2%	-	-	0.20	0.42	-
11	-	68.7%	25.8%	-9.1%	-	1.02	0.33	0.60
12	-	-13.8%	-4.3%	56.8%	-	0.37	0.33	0.58
13	-	3.6%	24.7%	-13.3%	-	0.59	0.63	0.32
14	-	-17.6%	-26.5%	-27.2%	-	0.33	0.32	0.51
15	-	69.3%	46.3%	48.4%	-	0.58	1.16	0.74

Table 3: Extended Weekly Increment Percentages Using Weighted Moving Averages

Week	Increment (60)	Increment (54)	Increment (48)	Increment (36)
2	13.50	21.12	5.42	5.58
3	9.78	25.62	5.45	5.59
4	21.34	23.24	5.37	5.56
5	1.29	16.61	5.13	5.51
6	−7.95	25.29	5.49	5.38
7	24.52	14.86	5.92	6.11
8	−11.26	48.10	7.91	6.21
9	−25.29	11.32	1.59	4.55
10	−19.99	−16.50	−8.21	4.09
11	−20.62	68.66	25.80	−9.10
12	−20.72	−13.76	−4.34	56.67
13	−20.67	3.60	24.70	−13.31
14	−20.68	−17.64	−26.55	−27.21
15	−20.68	69.30	46.35	48.38

Table 4: Demand Jump Table

Jump From	Jump To	Demand Jump
60	54	31%
60	48	53%
60	36	79%
54	48	34%
54	36	53%
48	36	32%

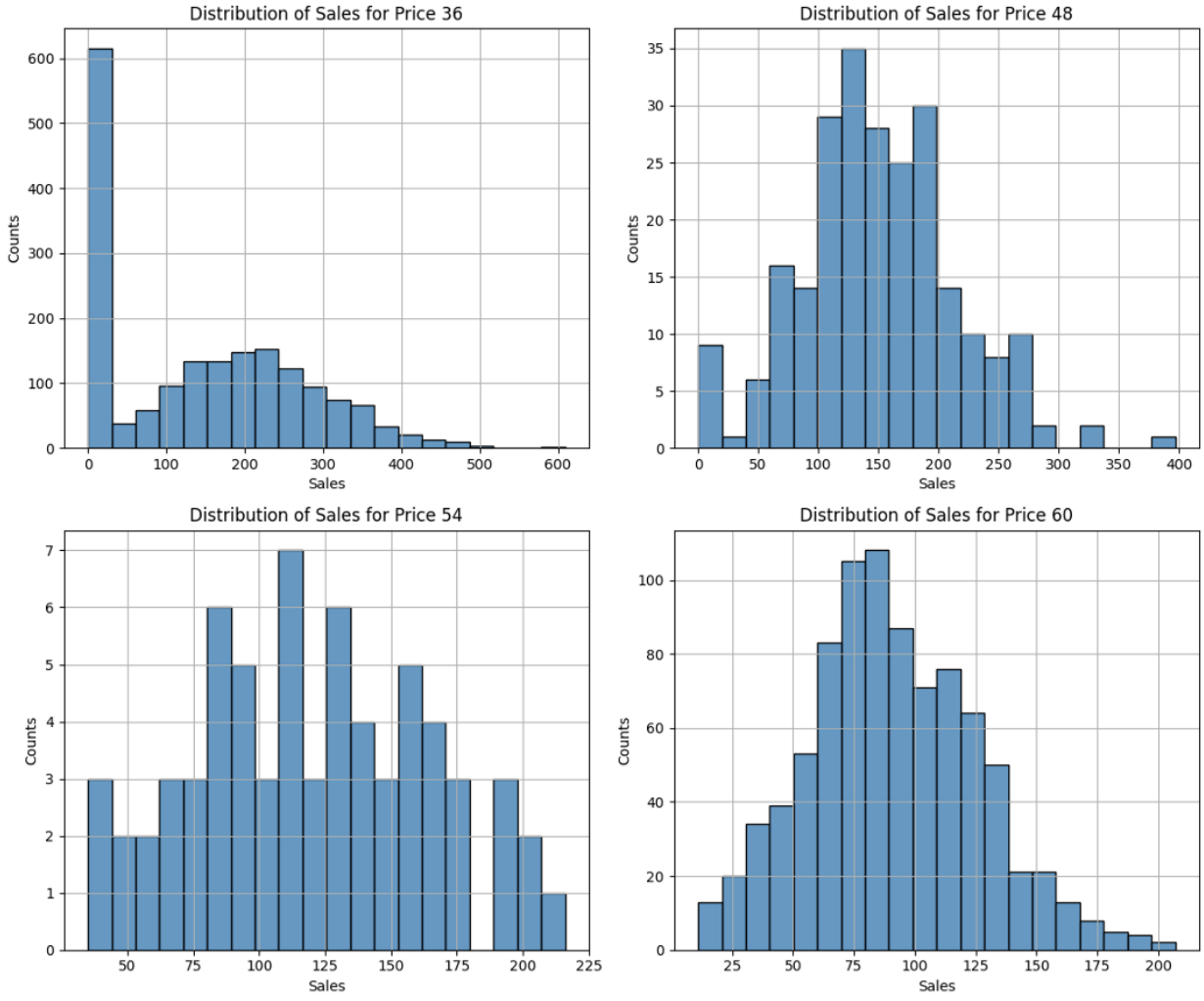


Figure 4: Distribution of Sales at Various Price Points

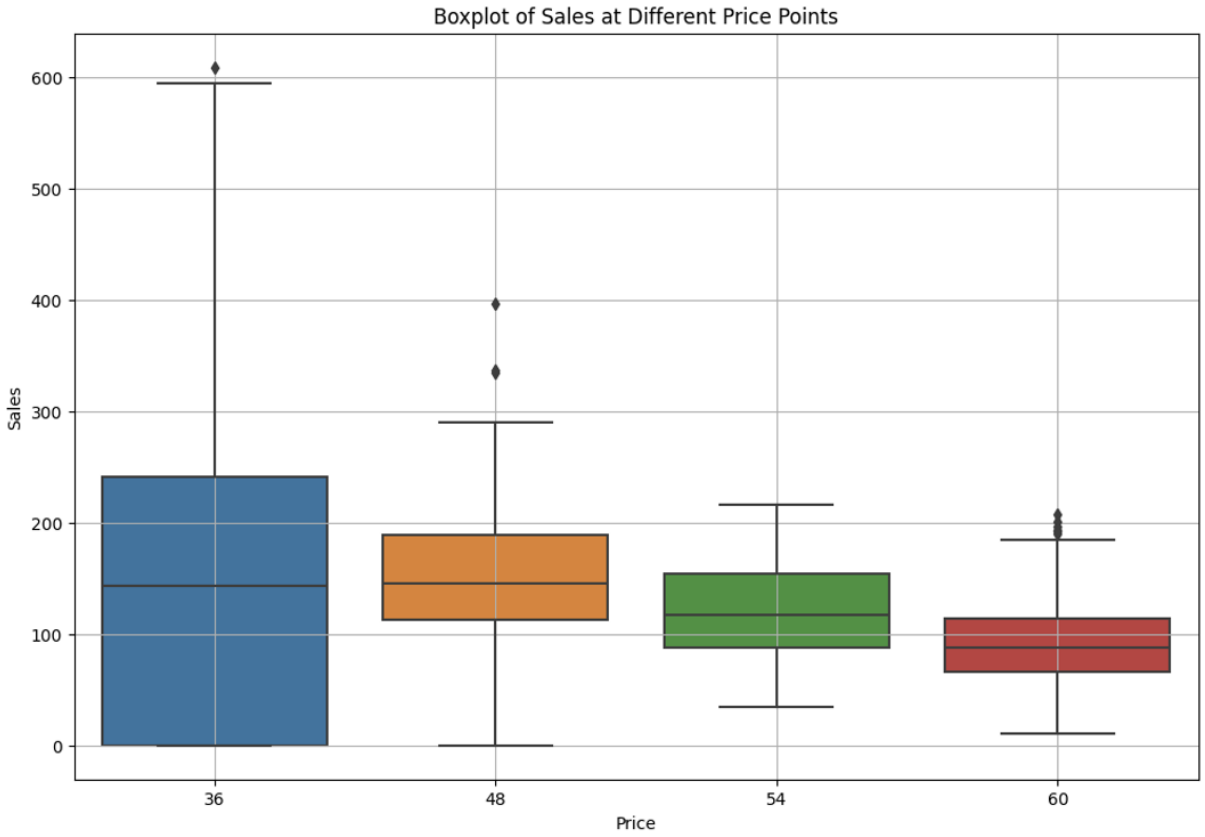


Figure 5: Boxplot of Sales at Different Price Points