Group Assignment 1 : Markdown Game Strategy Report

Presented to
Professor Maxime Cohen
T.A. Nymisha Bandi

By
Delisle, Audrey - 261142504
Barabasz, Michelle - 261152119
De Silva, Dhevin - 261177497
Belmadani, Samia - 260927564

MGSC 670 Revenue Management - Section 075

McGill University - Desautels Faculty of Management
Tuesday May 21st 2024

Table of Contents

Section I : Introduction

In the world of retail, setting the right prices can make or break a business. We're diving into the retail markdown game to figure out how to make the most money while selling a limited amount of inventory over 15 weeks. This analysis is based on the retail markdown game[1], where players aim to maximize total revenue by adjusting prices over a 15-week selling season with a fixed initial stock. The players must strategically decide when to lower prices (either by 10%, 20% or 40%) to increase sales without incurring significant losses from leftover stock at the end of the season.

We'll explain how we gathered and studied game data to understand what sells and when. Then, we'll talk about how we forecasted what people will buy at different prices. Based on all this, we'll share our markdown strategy to make sure we sell as much as possible without losing money. We'll also show how we evaluate our results, by looking at how much money we make, and how close we get to a revenue incurred through a scenario using "crystal-ball forecasting data" Our aim is to present these results to businesses in the future to convince them to use our markdown strategy in order to maximize profits while not compromising consumer surplus.

Section II : Data Analysis & Extraction

To begin, we wanted to gather historical data of playing the game to create our markdown strategy. We were given a couple data points for this project, but we deemed this was not enough to get an in depth analysis of an accurate markdown strategy. So, we came up with the idea to write a code that would play the game multiple times and record the results. To do this, we used an open source code as inspiration on how to automatically play a game and extract data. Specifically, we referred to a script from GitHub[2] which helped us understand the process of automating game interactions and data extraction.

We began by generating all possible combinations of prices. Since it is possible to have 60, 54, 48 and 36 in 15 different spots (representing weeks), we combined all possible combinations into one dataframe making sure the price never went up after being decreased. We also made sure each combo always started with 60, since this value was given to us. This gave us a total of 680 combinations. We then assigned a unique ID to each combination and converted the list into a DataFrame for easy manipulation. Next, we set up the Selenium WebDriver to automate interactions with the game's web interface. We navigated to the game's webpage and ensured the necessary buttons were clickable using explicit waits. We had to add wait times because without them, the code ran quicker than the website was responding.

[1] Ramandeep S. Randhawa. 2014. Retailer Game. https://randhawa.us/games/retailer/nyu.html
[2] Github. 2021. Scraper_bike.py. https://github.com/vsaahil/Revenue_Management_Retail_Markdown_Game_Strategy/blob/main/data/scraper_bike.py

To manage the large number of combinations, we divided them into six manageable sections. This allowed us to process the data in chunks, making the task more feasible. We then wrote the main scraping script to iterate through one section of the combinations. We replicated the game twice for each combination, simulating the button clicks according to the values in the combination sequence. Depending on the value, we clicked different buttons (e.g., "maintain", "10%", "20%", "40%") and extracted the results displayed in the game's result table. We also extracted additional data, such as revenue and perfect values, and appended these results to our data list. Once all data for a section was collected, we saved it to a CSV file. We repeated this process for each of the six sections. Finally, we combined all the individual CSV files into a single dataset.

Section III : Demand Prediction & Markdown Strategy

To predict demand, we first focused on identifying the most successful pricing combinations. The reason we wanted to do this is instead of going through over 10,000 data points (680 * 15) we only wanted to evaluate the iterations that were the most successful. We began by filtering our dataset to isolate the top 100 performing combinations, which were determined based on the smallest difference between the actual revenue and the perfect foresight strategy revenue. This filtering process enabled us to determine the pricing strategies that demonstrated the highest efficiency and profitability.

For each of these top 100 combinations, we constructed a new data frame that tracked the decisions made each week alongside the remaining inventory levels. We did this by establishing a set of rules to guide the weekly pricing decisions.

- If prices remained constant from one week to the next, the decision was marked as 'Maintain'.
- If prices decreased from \$60 to \$54, it indicated a '10%' discount.
- Other price drops were labeled as: from \$54 to \$48 as '20%' and from \$48 to \$36 as '40%'.

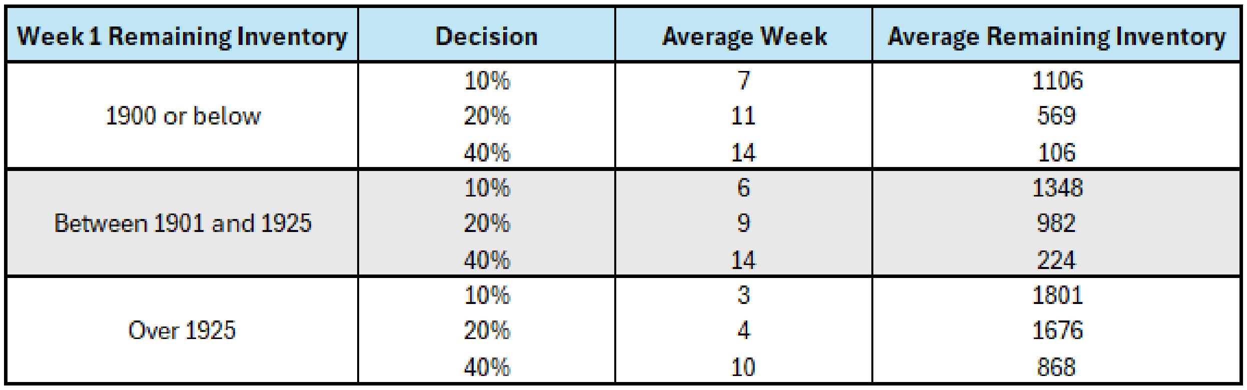
Using these rules, we were able to translate the price drops into the decisions that were made for each of the top 100 combinations over the 15-week period.

Next, we wanted to write a code that would tell us the average remaining inventory and the average week at which each type of price change (10%, 20%, 40%) was implemented. This analysis was important for deriving a generalizable markdown strategy. By understanding these averages, we could identify patterns and optimal timing for markdowns. However, one of the thoughts that came to mind was that the pricing strategy should be dynamic. When we played the game as humans, our pricing strategy depended on what the remaining inventory was at week 1 (this was given to us in the game). Logically, for instances where initial sales were high (inventory less than 1900), you would not want to markdown the price as early because you can tell the product is selling well. This was based on the assumption that higher initial sales indicated strong demand (steeper slope), allowing us to maintain higher prices for longer. On the other hand, if initial sales were lower (inventory greater than 1925), you would want to markdown the price quicker because the item did not have as high demand.

Hence, we split our top 100 data points into 3 categories: remaining inventory under 1900, between 1900 and 1925 and over 1925. These numbers were determined by starting the game multiple times and writing down what the remaining inventory at week 1 was over and over. Then the points were split into 3 equal categories.

Once we had these three categories, we ran the code to give us at what week and at what remaining inventory on average the price markdowns were made. This dynamic approach enabled us to adapt our markdown strategy to varying initial conditions, ensuring optimal performance across different scenarios. The main takeaways and our final markdown strategy are in the chart found below.

Table 1: Final Markdown Strategy



Week 1 Remaining Inventory	Decision	Average Week	Average Remaining Inventory
1900 or below	10%	7	1106
	20%	11	569
	40%	14	106
Between 1901 and 1925	10%	6	1348
	20%	9	982
	40%	14	224
Over 1925	10%	3	1801
	20%	4	1676
	40%	10	868

* Please view Appendix A to see the code results to form this table

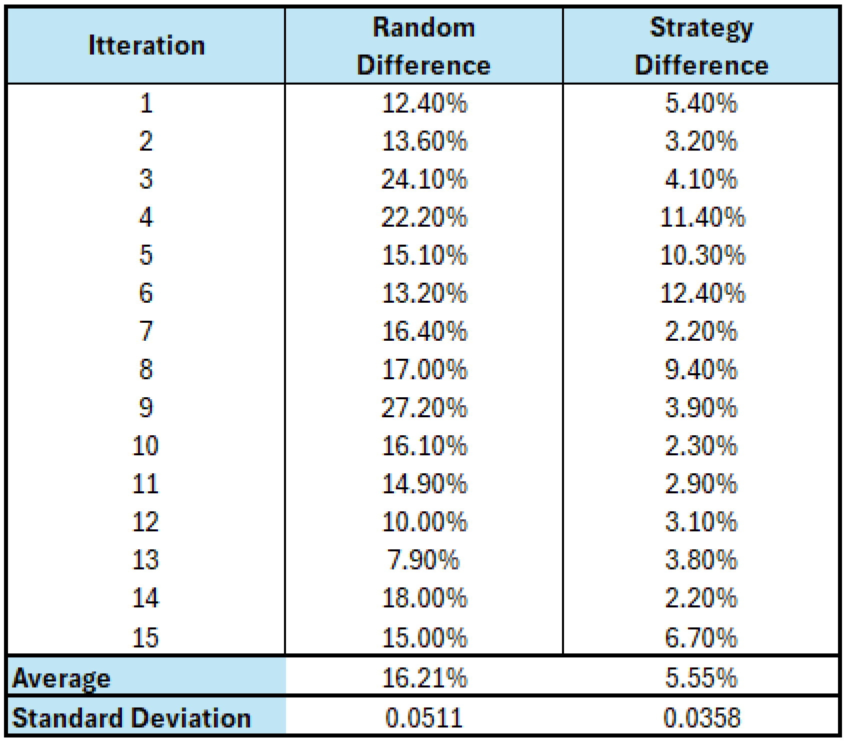
Section IV : Results Evaluation

In order to validate our markdown strategy, we wanted to compare our markdown strategy to random game outputs. We did this part manually. Initially, we played the retail markdown game 15 times, making random decisions to serve as a control group. This involved selecting price adjustments randomly without any strategic framework. For each gameplay, we recorded the resulting difference percentage in an excel sheet.

Next, we played the game 15 times but following our markdown strategy. In this phase, we used the initial week's remaining inventory to decide which of the three markdown strategies to follow. For instance, if the initial inventory was high, we clicked on the markdown percentages at the week given. This could also be done based on the average remaining inventory.

We then calculated the average revenue and the standard deviation for both the manual and enhanced markdown strategy gameplays. Please view the results below.

Table 2: Random vs Strategy Gameplay results



Itteration	Random Difference	Strategy Difference
1	12.40%	5.40%
2	13.60%	3.20%
3	24.10%	4.10%
4	22.20%	11.40%
5	15.10%	10.30%
6	13.20%	12.40%
7	16.40%	2.20%
8	17.00%	9.40%
9	27.20%	3.90%
10	16.10%	2.30%
11	14.90%	2.90%
12	10.00%	3.10%
13	7.90%	3.80%
14	18.00%	2.20%
15	15.00%	6.70%
Average	16.21%	5.55%
Standard Deviation	0.0511	0.0358

The average difference in revenue for the markdown strategy gameplay was significantly lower than that of the random gameplay. Specifically, our strategy yielded an average difference (5.55%) three times lower than the random approach (16.21%). This difference highlights the effectiveness of our strategic approach in optimizing revenue. We also observed a lower standard deviation in the enhanced markdown strategy gameplay (0.0358) compared to the manual gameplay (0.0511). Following our structured markdown strategy indicated more consistent performance and less outcome variability. A lower standard deviation is preferable as it implies greater predictability and reliability in the results.

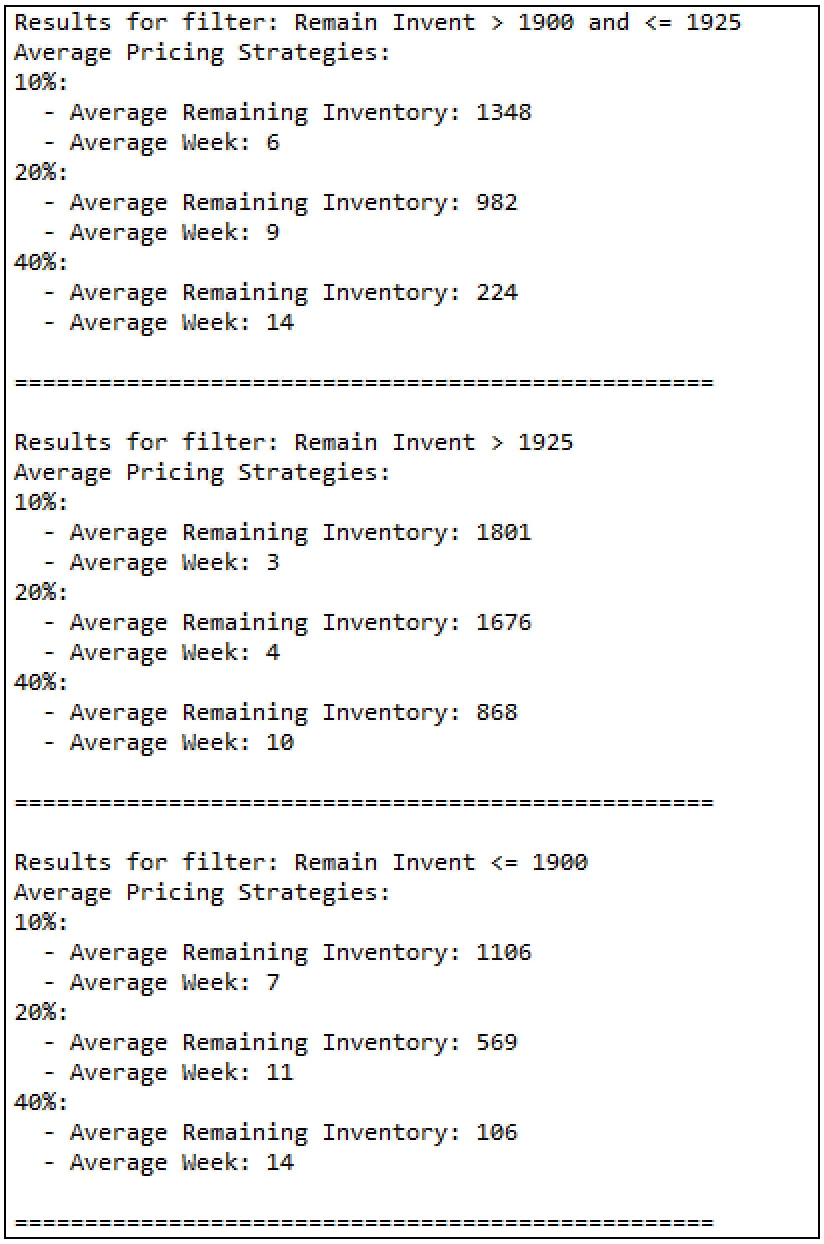
Section V: Conclusion

In conclusion, our analysis of the retail markdown game demonstrated the significant impact of strategic price adjustments on revenue optimization. By automating data extraction and analyzing historical game data, we identified the top-performing pricing combinations and derived a dynamic markdown strategy. This strategy, tailored to varying initial inventory levels, showed a marked improvement in revenue and consistency compared to random price adjustments. Specifically, our approach reduced the average revenue difference to 5.55%, a substantial improvement over the 16.21% difference seen with random strategies, and resulted in a lower standard deviation, indicating more predictable outcomes.

Our study involved a comprehensive process of data collection, demand prediction, and strategic markdown implementation. However, there is potential for further enhancement. If we were to redo this project with more time, incorporating neural networks could refine demand predictions by learning from larger datasets and complex patterns, potentially leading to even more effective markdown strategies. This advanced approach could further optimize pricing decisions, ensuring maximum profitability while minimizing leftover inventory.

Section VI: Appendix

Appendix A: Original Code Markdown Strategy



```
Results for filter: Remain Invent > 1900 and <= 1925
Average Pricing Strategies:
10%:
  - Average Remaining Inventory: 1348
  - Average Week: 6
20%:
  - Average Remaining Inventory: 982
  - Average Week: 9
40%:
  - Average Remaining Inventory: 224
  - Average Week: 14

==================================================

Results for filter: Remain Invent > 1925
Average Pricing Strategies:
10%:
  - Average Remaining Inventory: 1801
  - Average Week: 3
20%:
  - Average Remaining Inventory: 1676
  - Average Week: 4
40%:
  - Average Remaining Inventory: 868
  - Average Week: 10

==================================================

Results for filter: Remain Invent <= 1900
Average Pricing Strategies:
10%:
  - Average Remaining Inventory: 1106
  - Average Week: 7
20%:
  - Average Remaining Inventory: 569
  - Average Week: 11
40%:
  - Average Remaining Inventory: 106
  - Average Week: 14

==================================================
```