1. Overview of Problem

The primary objective of the retailer game is to develop a strategic markdown pricing plan to maximize total revenue over a 15-week selling season. The sales data includes weekly information about the price charged, the number of units sold, and the remaining inventory at the end of each week. There are four possible price points: an initial price of $60, and three discounted prices of 10% off ($54), 20% off ($48), and 40% off ($36).

In the game, each week presents an opportunity to either maintain the current price or decrease it. However, once the price is decreased, it cannot be increased again. If the price is reduced to 40% off, the game concludes. The initial inventory is set at 2000 units and the goal is to maximize revenue from this stock over the 15 weeks.

The game does not account for any costs, such as production or salvage, as the focus is purely on revenue maximization. Any leftover stock at the end of the 15 weeks is considered lost. The demand for each item varies with each run of the game, represented by a unique demand curve with an associated mean and standard deviation for each price level.

The challenge of the game lies in strategically adjusting the price levels and timing to maximize the revenue from the available stock within the stipulated time frame.

2. Data Extraction and Preparation

Introduction

The code provided is a Python script that programmatically explores and simulates all possible combinations of markdown pricing strategies (680 combinations in total) over a 15-week period in a rule-based retail game .These combinations are validated to ensure that they adhere to the game's rules, i.e., the price starts at $60 and does not increase in any subsequent week. The strategies are based on four different price points: $60, $54, $48, and $36, corresponding to discounts of 0%, 10%, 20%, and 40% respectively. It uses Selenium WebDriver for web scraping and automation to interact with the game's website interface and collect data.

Combo_ID	Week_1	Week_2	Week_3	Week_4	Week_5	Week_6	Week_7	Week_8	Week_9	Week_10	Week_11	Week_12	Week_13	Week_14	Week_15
0	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
1	60	60	60	60	60	60	60	60	60	60	60	60	60	60	54
2	60	60	60	60	60	60	60	60	60	60	60	60	60	60	48
3	60	60	60	60	60	60	60	60	60	60	60	60	60	60	36
4	60	60	60	60	60	60	60	60	60	60	60	60	60	54	54
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

(Snapshot of all possible combination of markdown strategy)

Web Scraping and Interaction

The second part of the script initiates a Selenium WebDriver session and navigates to the retailer game's webpage (http://www.randhawa.us/games/retailer/nyu.html). It locates the buttons corresponding to each pricing decision using their HTML ID attributes.

Strategy Simulation

In the third part, the script simulates the game for each valid pricing strategy. It iterates over the strategy combinations and clicks the appropriate button on the webpage depending on the pricing decision for each week.

Data Extraction

During the simulation, the script scrapes each game's results from the webpage for each week. This includes details like the **price, sales**, and **remaining inventory**, column **rev0** represents the actual revenue each game got, **perfect0** represents the Perfect foresight strategy, **difference0** represents the difference between perfect and actual revenue. This data is appended to the **`output`** list. Furthermore, it extracts the total revenue, perfect score, and the difference between the two, adding these performance metrics to the **`new_com_df`** DataFrame with the corresponding strategy ID.

Combo_ID	Week_1	Week_2	Week_3	Week_4	Week_5	Week_6	Week_7	Week_8	Week_9	Week_10	Week_11	Week_12	Week_13	Week_14	Week_15	Revenue	Perfect_Rev	Difference
0	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	120000	120000	0%
1	60	60	60	60	60	60	60	60	60	60	60	60	60	60	54	85398	103374	17%
2	60	60	60	60	60	60	60	60	60	60	60	60	60	60	48	64380	88884	28%
3	60	60	60	60	60	60	60	60	60	60	60	60	60	60	36	74268	96924	23%
4	60	60	60	60	60	60	60	60	60	60	60	60	60	54	54	100272	110334	9%
5	60	60	60	60	60	60	60	60	60	60	60	60	60	54	48	117360	118248	1%
6	60	60	60	60	60	60	60	60	60	60	60	60	60	54	36	23484	35652	34%
7	60	60	60	60	60	60	60	60	60	60	60	60	60	48	48	113076	114612	1%
8	60	60	60	60	60	60	60	60	60	60	60	60	60	48	36	55524	78276	29%
9	60	60	60	60	60	60	60	60	60	60	60	60	60	36	36	56592	79980	29%
10	60	60	60	60	60	60	60	60	60	60	60	60	54	54	54	117672	118368	1%

(Snapshot of new_com_df Data Frame after game simulation)

Strategy_ID	Week	Price	Sales	Remaining_Inventory
0	1	60	141	1859
0	2	60	156	1703
0	3	60	176	1527
0	4	60	163	1364
0	5	60	188	1176
0	6	60	128	1048
0	7	60	113	935
0	8	60	118	817
0	9	60	114	703
0	10	60	153	550
0	11	60	102	448
0	12	60	174	274
0	13	60	97	177
0	14	60	124	53
0	15	60	53	0
1	1	60	100	1900
1	2	60	107	1793
1	3	60	81	1712
1	4	60	60	1652
1	5	60	24	1628
1	6	60	108	1520
1	7	60	107	1413
1	8	60	114	1299

(Snapshot of all info columns in each game simulation)

The script can run multiple simulations for each strategy and includes a delay of one second between requests to avoid overwhelming the server. The results of these simulations provide insights into the effectiveness of each pricing strategy, offering a data-driven approach to maximize revenue over the 15-week selling season. However, due to the extremely large running time, we only ran simulation once for each combination and stored all combination and their results in (**all_combinations_results.csv**), and stored game details such as sales, prices, etc. for each game in (**simulate_with_all_comb.csv**) for next-step analysis.

3. Approaches

3a. Rule-based Step-by-step Approach

Step 1: Evaluate Initial Sales and identify the upper and lower bound

The first week's sales provide crucial information about the item's popularity. If the initial sales are high, it indicates a greater likelihood of selling the entire inventory within 15 weeks. To determine if the initial sales are sufficient, we calculate the expected sales per period by dividing the total number of units by the number of periods:

$$\frac{2000\ units}{15\ weeks} = 133.33\ units/week$$

If the first week's sales are equal to or greater than 133 units, the original price can be maintained and most of the inventory can still be sold by the end of the periods.

If the expected sales per period are lower than 45 units, based on the average demand lift shown in table 1 (60-36), applying a 40% discount from the second period may still result in unsold inventory. In this scenario, the highest promotion, 40%, must be applied immediately.

$$x + 3.09x \cdot 14 = 2000$$

$$x = 45\, units$$

where x is the expected sales of the first week, and 2.0883 is the demand lift that represents the expected sales change if a 40% discount is applied starting from the second week.

Table 1: Average demand lift calculated from 680 simulations

	60-54	**60-48**	**60-36**	**54-48**	**54-36**	**48-36**
Strategy	10%	20%	40%	11%	33%	25%
Average demand Lift	1.45	1.98	3.09	1.48	2.53	1.71

Once the upper and lower bounds are identified, the next step is to determine the strategy if the sales fall between 66 - 133 units.

Step 2: Identify the step-by-step strategy based on sales

The rule-based strategy aims to achieve optimal weekly sales and clear out the inventory close to 15 weeks. To accomplish this, the sales of each week must be kept as close as possible to the optimal level, which has been determined to be around 133 units. To achieve this optimal sales target, the pricing strategy needs to be adjusted based on the demand lift.

To calculate the desire demand lift (promotion), the formula used is:

$x \cdot demand\ lift \approx 133\ units$, where x is the sale of a week

This formula can be readjusted to get the desired demand lift (promotion):

$$\frac{133}{x} = demand\ lift$$

Due to the potential lagging effect of promotions, where the impact may not be immediate but rather take effect in the following week, a moving average of the previous two months' sales is used to establish a base x. This helps to ensure more accurate adjustments to the pricing strategy to reach optimal weekly sales.

The final rule-based model can be expressed as:

$$\frac{133}{\frac{x1+x2}{2}} = desired\ demand\ lift$$

where x1 and x2 represent the expected sales per period for the last two months.

The selection of a suitable pricing strategy can be based on the proximity of the calculated demand lift to the available base prices (60, 54, or 48). Depending on this proximity, the options are either to promote the product or maintain the current pricing. Table 2 indicates the range of demand lift for each strategy and the corresponding demand range.

Table 2: Demand range and the corresponding rule-based strategy

	Current price is $60			Current price is $54		Current Price $48
	60-54	**60-48**	**60-36**	**54-48**	**54-36**	**48-36**
Pricing Strategy	10%	20%	40%	20%	40%	40%
Demand Lift	1.45	1.98	3.09	1.48	2.53	1.71
Considered Demand Lift Range	1.23 to 1.72	1.73 to 2.54	2.55 and up	1.24 to 2	2 and up	1.35 and up
2-week Average Demand	77-108	52-76	52 or lower	66-107	66 or lower	98 or lower

By considering the lagging effect of promotions, this model allows for precise pricing adjustments to optimize weekly sales and clear out inventory within the 15-week timeframe.

3b. Linear Programming using Historical Data

The objective of maximizing revenue can be achieved by framing the problem as a Linear Optimization problem. In this approach, the objective function is defined as the sum of the product of the sale price, average weekly demand, and the number of weeks an item is sold at a particular price. The decision variables in this case are the number of weeks the item is sold at each price point. To create this model, the historical sales data was utilized to determine the weekly average demand for each item at each price point. However, in cases where inventory sold out before the 15-week period, the average was calculated by excluding the weeks where the item was out of stock. The demand jump, or lift, when the price dropped from $60 to either $54, $48, or $36 was determined using the weekly average sales values, which showed that the demand lift was the same for all items. The average demand jump for each price drop and the incremental jump in demand lift were then calculated. The result of this model is a pricing strategy that maximizes revenue while minimizing the risk of stockouts or excess inventory.

Item	Price 1	Price 2	Price 3	Price 4	Demand Increase	Average Demand Increase
	$60	$54	$48	$36		
	Average Sales					
1	58	76			1.30	
2	108	144			1.34	
3	59	82			1.39	**1.31**
4	61	78			1.27	
5	93	114			1.23	
6	114		209		1.83	
7	67		120		1.77	
8	53		97		1.83	**1.73**
9	74		132		1.79	
10	67		97		1.44	
11	100			264	2.63	
12	64			189	2.94	
13	66			197	3.00	**2.81**
14	61			164	2.67	
15	62			175	2.81	

Table 3: Each Item and the Average Sales Corresponding to Each Price

Demand Estimation

The Demand Increase or Lift was calculated by:

$$Demand\ Lift\ = |(\ New\ Avg\ Sales\ -\ Previous\ Avg\ Sales\ /\ Previous\ Avg\ Sales\ )|\ +\ 1$$

For the first cell,

$$Demand\ Lift\ =\ |(\ 76\ -\ 58/\ 58\ )|\ +\ 1\ =\ 1.30$$

Then the Average Demand Increase was calculated for each price drop:

$$\frac{\sum_{i=1}^{5} Demand\ Lift_i}{5}$$

For the first cell,

$$\frac{1.30\ +\ 1.34\ +\ 1.39\ +\ 1.27\ +\ 1.23}{5}\ =\ \mathbf{1.31}$$

Since the Demand lift with respect to price is found to be homogenous across items, the demand for each price can be estimated using the lift factor as a constant through each price drop and utilizing the initial number of sales observed during week 1 when game starts.

Thus, we can estimate the weekly demand and revenues per price using the table below. Assume initial sales was found to be 65 units in week 1.

Prices	**$60**	**$54**	**$48**	**$36**
Demand Lifts	1.00	1.31	1.73	2.81
Weekly Demand (Initial Sales x Lift)	65.00	84.89	112.67	182.62

Table 4: Weekly Demand per price

Where the Weekly Demand for each price is equal to the sales realized in week 1 multiplied by its lift:

For instance, the Weekly Demand for price 60 is found using:

$$initial\ sales\ *\ lift\ =\ 65\ *\ 1\ =\ 65\ units\ every\ week\ with\ price\ set\ to\ \$60$$

Consequently, the Weekly Demand for price 48 is found using:

$$initial\ sales\ *\ lift\ =\ 65\ *\ 1.73\ =\ 112.67\ units\ every\ week\ with\ price\ set\ to\ \$48$$

Revenue Estimation

To get the estimated revenues or total quantities sold for each price, the weekly demand mentioned earlier should be multiplied by the number of weeks the item is sold. That is:

$$Total\ Revenue =\ Weekly\ Demand\ *\ number\ of\ weeks$$

Where,

Number of weeks is unknown and will be used as the decision variable in the LP Formulation.

Formulation

Decision Variables:

$Let\ x_i\ be\ the\ number\ of\ weeks\ the\ item\ is\ sold\ at\ prices\ \$60, \$54, \$48, and\ \$36$

x_1: $number\ of\ weeks\ the\ item\ is\ sold\ at\ price\ \$60$

x_2: $number\ of\ weeks\ the\ item\ is\ sold\ at\ price\ \$54$

x_3: $number\ of\ weeks\ the\ item\ is\ sold\ at\ price\ \$48$

x_4: $number\ of\ weeks\ the\ item\ is\ sold\ at\ price\ \$36$

x_i variables are formulated to be integer variables since the assumption is that the choice of price is locked for the entire week. Therefore, the user can't modify the price throughout the week.

Objective Function:

To Maximize Revenue,

$$\sum_{i=1}^{4} Price_i * S_{60} * lift_i * x_i$$

Where,

$Price_i$: $price\ given\ at\ \$60, \$54, \$48, and\ \$36$

S_{60}: $Initial\ sales\ realized\ for\ every\ week\ at\ price\ 60\ as\ the\ game\ starts\ (real\ time\ data)$

$lift_i$: $Demand\ lift\ for\ each\ price\ which\ is\ found\ to\ be\ 1, 1.31, 1.73, and\ 2.81\ (Table\ 3)$

x_i: $number\ of\ weeks\ the\ items\ is\ sold\ at\ prices\ \$60, \$54, \$48, and\ \$36$

Constraints:

Time Constraint: Inventory must be emptied in the next 14 weeks

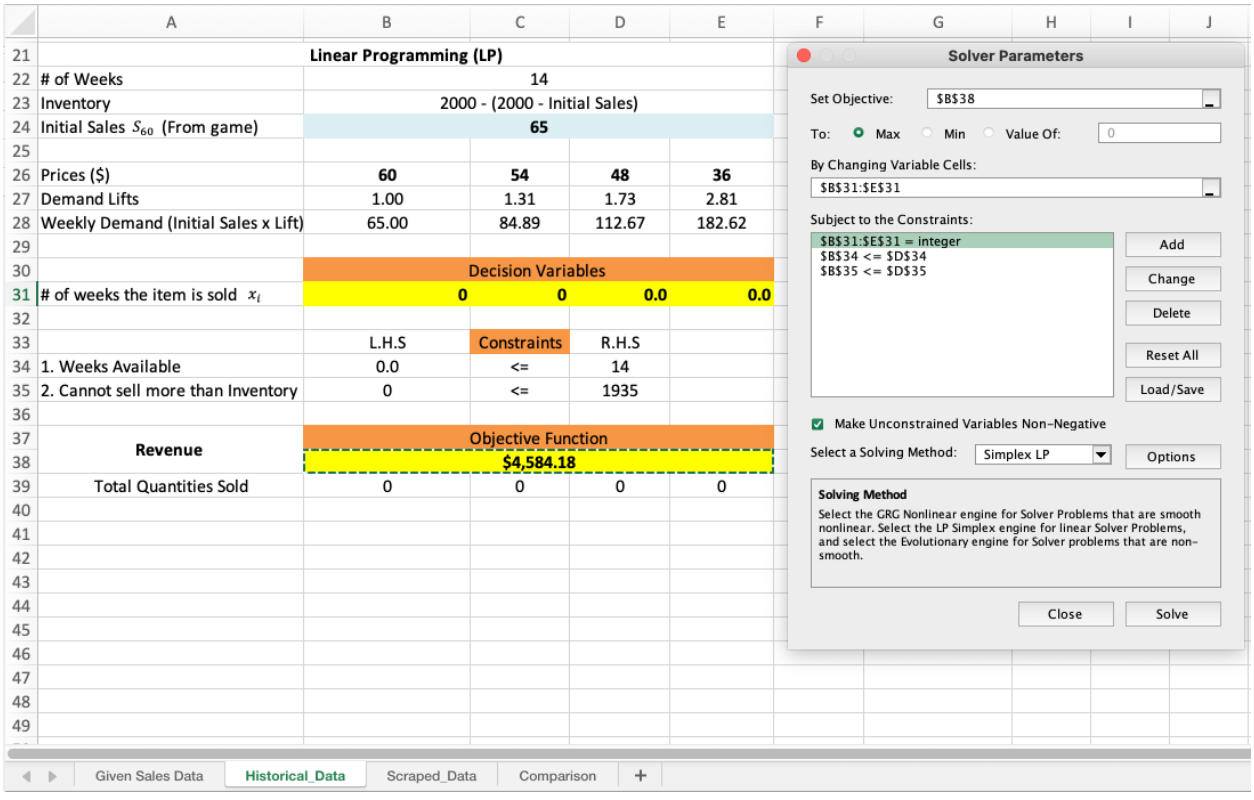
$$x_1 + x_2 + x_3 + x_4 \leq 14$$

Inventory Constraint: Cannot sell more than the inventory available after initial sales realized

$$\sum_{i=1}^{4} S_{60} * lift_i * x_i \leq 2000 - S_{60}$$

Excel Output (When S_{60} is realized as 65 units per week)

Updating the initial sales cell highlighted in blue as 65 and using Excel solver as shown on the right hand side below:



	A	B	C	D	E
21		**Linear Programming (LP)**			
22	# of Weeks		14		
23	Inventory		2000 - (2000 - Initial Sales)		
24	Initial Sales S_{60} (From game)		**65**		
25					
26	Prices ($)	**60**	**54**	**48**	**36**
27	Demand Lifts	1.00	1.31	1.73	2.81
28	Weekly Demand (Initial Sales x Lift)	65.00	84.89	112.67	182.62
29					
30			Decision Variables		
31	# of weeks the item is sold x_i		**0** **0**	**0.0**	**0.0**
32					
33		L.H.S	Constraints	R.H.S	
34	1. Weeks Available	0.0	<=	14	
35	2. Cannot sell more than Inventory	0	<=	1935	
36					
37	**Revenue**		Objective Function		
38			**$4,584.18**		
39	Total Quantities Sold	0	0	0	0

Solver Parameters

Set Objective: $B$38

To: Max Min Value Of: 0

By Changing Variable Cells: $B$31:$E$31

Subject to the Constraints:

$B$31:$E$31 = integer
$B$34 <= $D$34
$B$35 <= $D$35

Add | Change | Delete | Reset All | Load/Save

Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP | Options

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Close | Solve

Given Sales Data | Historical_Data | Scraped_Data | Comparison | +

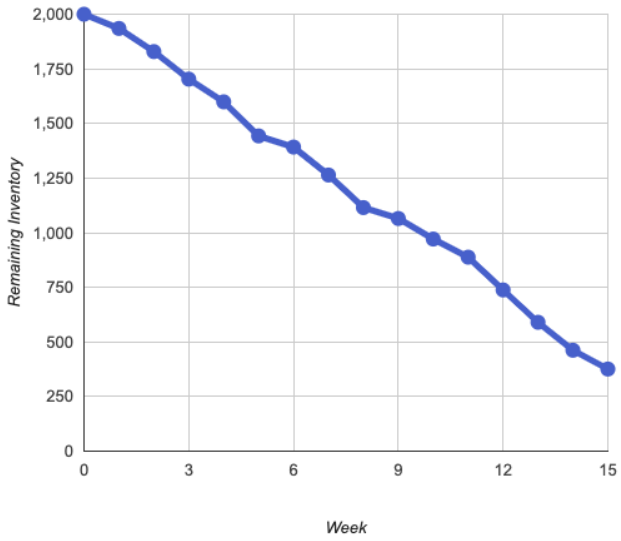
Results are shown below:

Linear Programming (LP)				
# of Weeks	14			
Inventory	2000 - (2000 - Initial Sales)			
Initial Sales S_{60} (From game)	**65**			
Prices ($)	**60**	**54**	**48**	**36**
Demand Lifts	1.00	1.31	1.73	2.81
Weekly Demand (Initial Sales x Lift)	65.00	84.89	112.67	182.62
	Decision Variables			
# of weeks the item is sold x_i	**0**	**0**	**9.0**	**5.0**
	L.H.S	Constraints	R.H.S	
1. Weeks Available	14.0	<=	14	
2. Cannot sell more than Inventory	1927.083457	<=	1935	
Revenue	Objective Function			
	$86,127.08			
Total Quantities Sold	0	0	1014	913

As seen above, the Maximum total Revenue is estimated as $86,127 upon dropping the price to $48 for the first 9 weeks and $36 for the last 5 weeks.

Plugging the solved factors into the game shown below, the revenue calculated is found to be $71,592 whereas, the optimal demand is $77,592 with a revenue difference of **7.7%.**

Week	Price	Sales	Remaining Inventory
1	60	65	1935
2	48	106	1829
3	48	126	1703
4	48	104	1599
5	48	156	1443
6	48	51	1392
7	48	128	1264
8	48	149	1115
9	48	50	1065
10	48	94	971
11	36	83	888
12	36	150	738
13	36	148	590
14	36	128	462
15	36	86	376



Maintain Price | 10% | 20% | 40%

Your revenue: $71,592, Perfect foresight strategy: $77,592, Difference: 7.7%

Enter the numeric seed (key) for this item:

1234567890

The result is decent since the historical data is only analyzing 15 items. Next strategy will be using the demand lift factors from the *Scraped Data* which analyzes all possible combinations.

3c. Linear Programming using Scraped Data

Similar approach was followed except that the demand lifts for each price are analyzed using all 680 possible combinations. Demand lifts are found to be:

Price Drop	Average Demand Increase for Each Price
60 - 54	1.45
60 - 48	1.98
60 - 36	3.09

Table 5: refer to table 1 for more details

Formulation is exactly the same but with modified demand lift factors as highlighted below

Objective Function:

To Maximize the revenue,

$$\sum_{i=1}^{4} Price_i * S_{60} * lift_i * x_i$$

Where,

$Price_i$: $price\ given\ at\ \$60, \$54, \$48, and\ \$36$

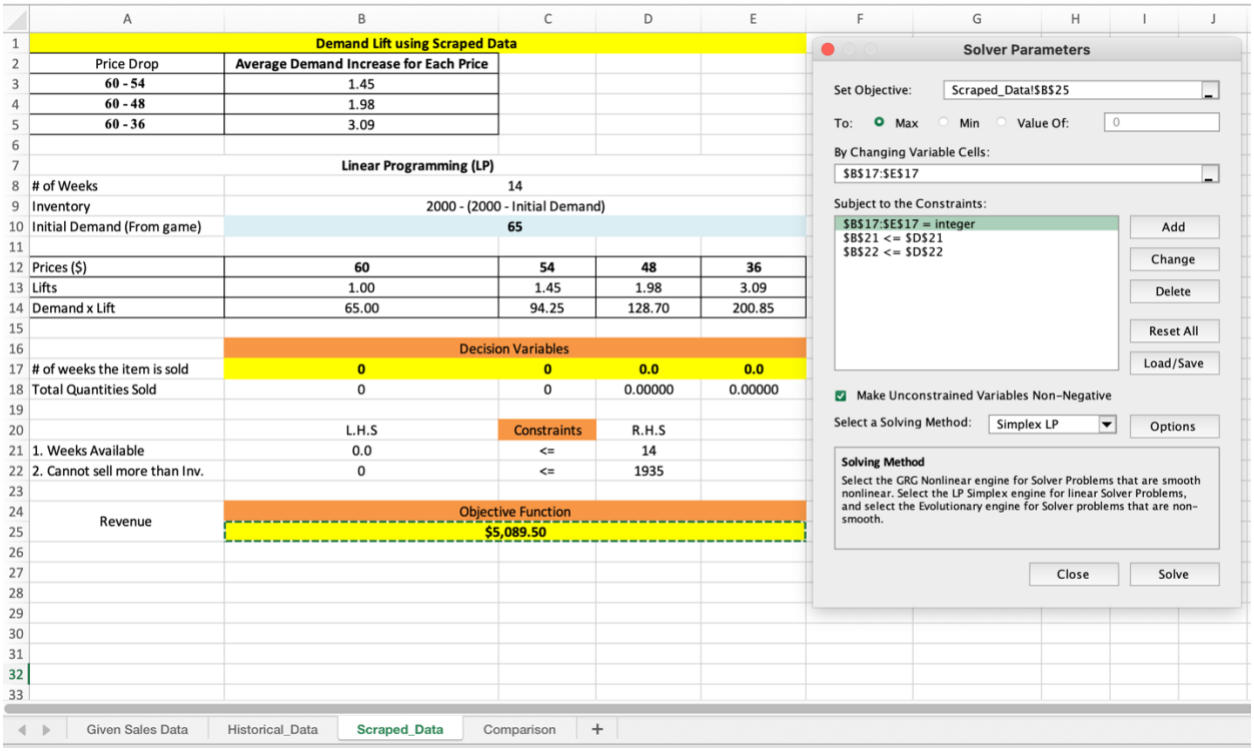
S_{60}: $Initial\ sales\ realized\ for\ every\ week\ at\ price\ 60\ as\ the\ game\ starts\ (real\ time\ data)$

$lift_i$: $Demand\ lift\ for\ each\ price\ which\ is\ found\ to\ be\ 1, 1.45, 1.98, and\ 3.09\ (Table\ 5)$

x_i: $number\ of\ weeks\ the\ items\ is\ sold\ at\ prices\ \$60, \$54, \$48, and\ \$36$

Excel Output (When S_{60} is realized as 65 units per week)

Updating the initial sales cell highlighted in blue as 65 and using Excel solver as shown on the right-hand side below:



	A	B	C	D	E
1		Demand Lift using Scraped Data			
2	Price Drop	Average Demand Increase for Each Price			
3	60 - 54	1.45			
4	60 - 48	1.98			
5	60 - 36	3.09			
6					
7		Linear Programming (LP)			
8	# of Weeks	14			
9	Inventory	2000 - (2000 - Initial Demand)			
10	Initial Demand (From game)	65			
11					
12	Prices ($)	60	54	48	36
13	Lifts	1.00	1.45	1.98	3.09
14	Demand x Lift	65.00	94.25	128.70	200.85
15					
16		Decision Variables			
17	# of weeks the item is sold	0	0	0.0	0.0
18	Total Quantities Sold	0	0	0.00000	0.00000
19					
20		L.H.S	Constraints	R.H.S	
21	1. Weeks Available	0.0	<=	14	
22	2. Cannot sell more than Inv.	0	<=	1935	
23					
24	Revenue	Objective Function			
25		$5,089.50			

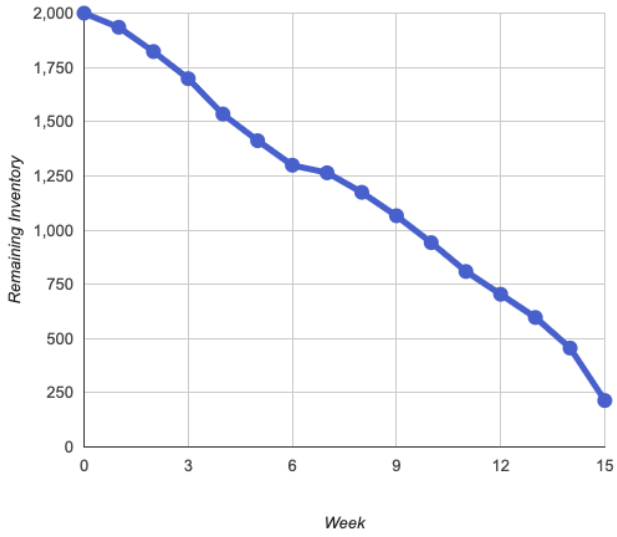
Results are shown below:

	Linear Programming (LP)			
# of Weeks	14			
Inventory	2000 - (2000 - Initial Demand)			
Initial Demand (From game)	65			
Prices ($)	60	54	48	36
Lifts	1.00	1.45	1.98	3.09
Demand x Lift	65.00	94.25	128.70	200.85
	Decision Variables			
# of weeks the item is sold	0	0	13.0	1.0
Total Quantities Sold	0	0	1673.10000	200.85000
	L.H.S	Constraints	R.H.S	
1. Weeks Available	14.0	<=	14	
2. Cannot sell more than Inv.	1873.95	<=	1935	
Revenue	Objective Function			
	$92,628.90			

As seen above, the Maximum total Revenue is estimated as $92,629 upon dropping the price to $48 for the first 13 weeks and $36 for the last week.

Plugging the solved factors into the game shown below, the revenue calculated is found to be $83,604 whereas, the optimal demand is $86,964 with a revenue difference of **3.9%**.

Week	Price	Sales	Remaining Inventory
1	60	65	1935
2	48	112	1823
3	48	125	1698
4	48	163	1535
5	48	123	1412
6	48	113	1299
7	48	35	1264
8	48	90	1174
9	48	108	1066
10	48	124	942
11	48	133	809
12	48	105	704
13	48	107	597
14	48	141	456
15	36	242	214



Maintain Price | 10% | 20% | 40%

Your revenue: $83,604, Perfect foresight strategy: $86,964, Difference: 3.9%

Enter the numeric seed (key) for this item:

1234567890

The result is < 5% difference which is satisfactory. This demonstrates the significant importance of training from larger portion of data. However, since the game strategy is random, both LP approaches, *Historical data* and *Scraped data* will be conducted on 10 random simulations and the difference of each simulation will be captured to verify and confirm the optimal approach.

4. Optimal Markdown Strategy

Three approaches were conducted, one using the *rule-based* model, second using LP with demand lifts observed from the *Historical Data* and the last approach was using LP with demand lifts utilized from the *Scraped Data*. Since the game initial sales varies and cannot be fixed for direct comparison. The game was played for 30 items in total, 10 items per model, and the difference between the calculated revenue and optimal revenue was captured for each round.

Followed by calculating the average difference based on the 10 rounds. As shown below, the results of the ***Scraped Data*** model offered a significant improvement in results of Average Error to be **1.63** as opposed to 3.3 for *rule-based* and 2.7 for *Historical Data*. More importantly, all 10 rounds of the *Scraped Data* model provided consistent results of less than 5% in revenue difference.

	Difference (%)			AVG Difference (%)		
Item	Rule-based	Historical Data	Scraped Data	Rule-based	Historical Data	Scraped Data
1	0.8	4.5	0.8	3.3	2.7	**1.63**
2	1.7	5.4	0			
3	1.5	0	4.4			
4	3.3	2.6	1.2			
5	0.9	0.6	0			
6	5.1	0	0			
7	3.1	4.7	4.3			
8	0.8	0	1.7			
9	8.8	1.5	0			
10	6.8	7.7	3.9			

Please note that all LP simulations are documented in screenshots for credibility.

5. Conclusion

Ensuring the successful implementation of markdowns is of utmost importance in the retail industry as it enables optimal profit and unit sales through effective revenue management. Managers employ diverse strategies and conduct experiments with various algorithms to accomplish this objective. The present study focuses on evaluating the effectiveness of different algorithms within The Retail Markdown Game and utilizes simulations to identify the most suitable algorithm that minimizes the disparity between revenue obtained with perfect information and the algorithm's revenue.

The findings highlight that the **Linear Programming using Scraped Data** algorithm outperforms both the Rule-based Approach and the Linear Programming using Historical Data algorithm. It consistently generates revenue with the smallest average difference when compared to the revenue obtained with perfect information. Although this standardized approach lacks the ability to consider diverse consumer trends and variations in unit stock sensitivity, its alignment with existing literature and suitability for implementation make it an ideal choice.