Analytical Decision Making

Final Report

Team dunnhumby

$$Final = \sum_{TeamMembers=1}^{4} TeamMembers$$

With TeamMembers in (Dane Hamlett,
Leo Pei,
Rae Huang,
Scott Virshup)

Table of Contents

Introduction to the Business Problem

As part of the UC Davis MSBA program, our team has been partnered up with dunnhumby, a UK-based customer science company leveraging data to optimize customer loyalty and pricing strategies for global retailers. This partnership is grounded on the execution of a 10-month analytics-based practicum project, aimed at tackling an industry knowledge gap and improving the consumer price perception of one their key clients, Raley's Supermarkets.

Raley's Supermarkets is a family-owned grocery chain that currently operates 128 stores in Northern California and Nevada. According to our key stakeholders at dunnhumby, Raley's has traditionally scored very high with their customers on specific elements of their shopping experience - service, product quality, etc., yet they have also scored very low on price perception.

While our practicum team has been hard at work identifying the link between consumer price perception, competitor prices, and price elasticities, we have also identified another problem that dunnhumby and Raley's face: determining an optimal price to charge. There are many factors that come into play when setting prices at a grocery store. We hypothesis that if Raley's can identify a way to regularly update prices based on a long-term strategy to maximize revenue, then they will be able to improve their business's price perception.

In this jupyter notebook, we explore how to use a relatively simple dynamic programming model to determine what prices should be set for a certain product (we've chosen the product "Orange Milanos" as an example) given only a demand function and some initial conditions.

This notebook walks through the logic behind defining the correct objective function, decision variables, and constraints. It explores the data behind the model, creates our own demand function, and then plugs it all into a SciPy minimization solver to determine the optimal prices that will achieve the best revenue.

Defining the Problem

Objective Function

Raley's has control over their price, but how do prices influence their unit sales? How can Raley's maximize the revenue by choosing different price levels of their product?

$$Objective: Maximize \sum_{t=1}^{52} (P_t \times S_{Pt})$$

Decision Variables

For each time period, t, there is a price, P. t is an index between 1 and 52 for each week of the year. S(P,t) represents unit sales as a function of price and week. Similarly, R(P,S,t) represents revenue as a function of price and week.

Constraints

We have to make some assumptions about our constraints to simplify our model. They are:

- Due to restocking limitations, no single week can sell more than 200 boxes of orange milanos

$$S_{Pt} <= 200$$

- Due to a marketing contract with Pepperidge Farms, Raley's has promised to purchase at least 1500 boxes of orange milanos. If less than this number is sold, Raley's will have to pay for them anyway.

$$\sum_{t=1}^{52} S_{Pt} >= 1500$$

- Raley's contract with Pepperidge Farms also limits the total unit sales for the entire year to be 7,500 boxes due to their manufacturing and supply chain limitations.

$$\sum_{t=1}^{52} S_{Pt} <= 7,500$$

- Non-negativity
    - Price and Demand cannot be negative:

$$P >= 0$$

$$S_{Pt} >= 0$$

Demand Function

Having a demand function is an important component of this dynamic optimization problem. This is necessary in order to map the demand at any given price. This is critical to know because, from the available data in the Raley's POS system, we only have one price point at any given time. Creating a demand function allows us to anticipate how that demand would

have changed if we had a slightly different price. An example of such a demand function is below (we do not end up using this function in our model, it is just illustrative):

$$S(P,t) = \frac{1}{B}\left(A - P\frac{D}{D+t}\right)$$

We can also adjust this equation slightly to determine price given price, week, and constants of A, B, and D through some simple algebra:

$$P(S,t) = (A - (B \times S))\frac{D+t}{D}$$

Explore the Data

As part of our practicum project, we have access to the Point Of Sale (POS) transactional data from Raley's supermarkets. To better understand the size and composition of the data that is required for our model, we explore it below. There are three key columns:

- **total_units_sold** This column is intuitively the unit sales for that product during a given time period (in this case, weeks)
- **avg_active_price** This column was created throughout the work on our practicum and is actually a combination of a number of other columns. This column takes the lowest active price for a given week, and averages out all price points within the system. By "lowest active price," it is understood that Raley's sometimes offers promotional pricing and/or loyalty pricing for certain products; this column will choose whichever one of those prices is lowest (the best deal) and display it.
- **fiscal_year_week** This column is also broken into two other columns **fiscal_year** and **fiscal_week** which simply adds a time-series component to this data.

Data Improvements

For the model we use in this problem, the above is all the data that is necessary. However, as is described later on when talking about the limitations of this model, more data points will be necessary when improving the accuracy of the demand function. Some other parameters/model features that are being explored as part of my practicum project, and should be incorporated into future versions of this model are:

1. **Holiday Flag -** Useful in addressing seasonality that might influence sales.
2. **Lagged Average Price -** Helpful in accounting for previous prices strongly influencing their current sales.

3. **Marketing Spend -** Amount spent on marketing for a certain product influences demand and revenue by incentivizing sales but also decreasing the amount of revenue per sale.
4. **Competitor Pricing -** Competitor prices are a cornerstone of our practicum project. We use a log of the difference in prices instead of the actual price of the competition.

In addition to our recommendations around which columns should be included, further expanding on the number of data points will also be helpful. This example only uses 52 rows from three-year-old data, which has clear implications on how relevant any insights are.

Dynamic Optimization

The logic to find a solution follows that, for a given time period, we will use the demand function to calculate the anticipated demand units sold. Because we are using dynamic programming to solve this problem, the demand will be calculated for each possible price point, and the optimal price level will be determined. This will then be repeated for each time, t, such that we have a matrix of unit sales as a function of price (x-axis) and time (y-axis). These matrices are shown in the appendix under “heatmap of demand” and “heatmap of revenue.” Their interpretations can be found in the jupyter notebook that comes with this report.

Defining the Problem Parameters

Because we want to maximize revenue, we set our objective function to minimize -1 * (price * unit sales). Defining the constraints means creating a set of inequalities that align with the constraints outlined earlier in the paper. There are 5 constraints total, but adding more would help contain this model within the bounds of reality and how Raley’s truly operates.

We also defined demand by plugging in average price and sales into a simple linear regression and turning the output into a demand function. We adjusted the function slightly to account for changes in time that are perhaps specific to our chosen product. The demand function is:

$$S(P,t) = \frac{PriceCoefficient \times 10 \times t}{P \times t^2}$$

Our team used the python library scipy to run the actual optimization. The method scipy.optimize takes our objective function, constraints, bounds, initial starting values, and uses the Sequential Least SQuares Programming (SLSQP) method to solve the model.

Interpretation of the Output

The last array of numbers (see appendix “model output”) were the suggested price points for the 10 weeks that the model interpreted. We could have had the model suggest 52 prices (1 per week), but having 10 price points is more realistic and comparable to the real world because Raley's cannot realistically change their price every week. 10 price points is something that our stakeholders at dunnhumby agree is in-line with how quickly a supermarket can change their display pricing and POS system relationships.

We find that these prices decrease over time. This outcome is expected due to our demand model decreasing demand as time marches steadily onward. This aspect of the demand model would also make sense if our product showed seasonal variations, or if the product was part of a one-time promotion (for instance, if it was a branded promotional product that sponsored a movie).

Prices start at approximately $7.45, and decline to about $0.75. This optimization model also predicts that we should expect to sell 1,500 units of these cookies over the course of these 10 weeks by using these price points. However, those 1,500 units are calculated as the total unit sales from just the 10 weeks that were calculated, in reality there will be 52 total weeks instead of 10, so we could expect this figure to be much larger.

Limitations of this Model

Such a dynamic model requires both the accurate definition of the constraints that our model will operate under as well as the accurate definition of demand. In our example, demand is defined very generally and so does not precisely represent how price will influence unit sales. Instead, we should improve the model in the future first by spending time to collect additional data on marketing efforts, seasonality, competitor prices, and other columns in order to refine our demand function. As part of our practicum team, we will further expand upon this by asking our dunnhumby stakeholders what their demand functions currently are, and see how closely they align to what we've created ourselves.

We made many assumptions that made this problem solvable with the limited data that we have, but that also means that it is less representative of the real world. As the famous saying goes, "all models are wrong, but some are useful." This wise saying is what should be considered if one uses this model to make pricing decisions in the future.

References

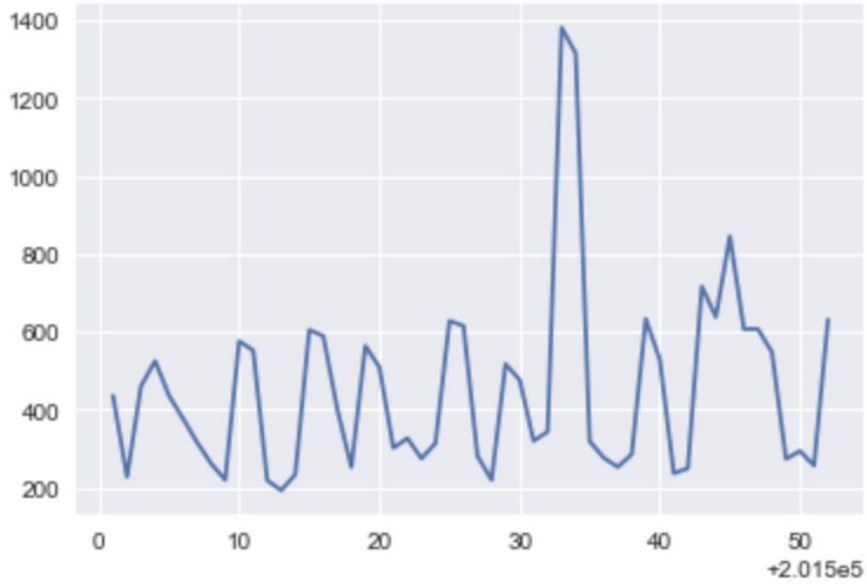
Note: most referenced material came from class notes, examples, and homework

http://isaacslavitt.com/2014/06/20/linear-optimization-in-python/

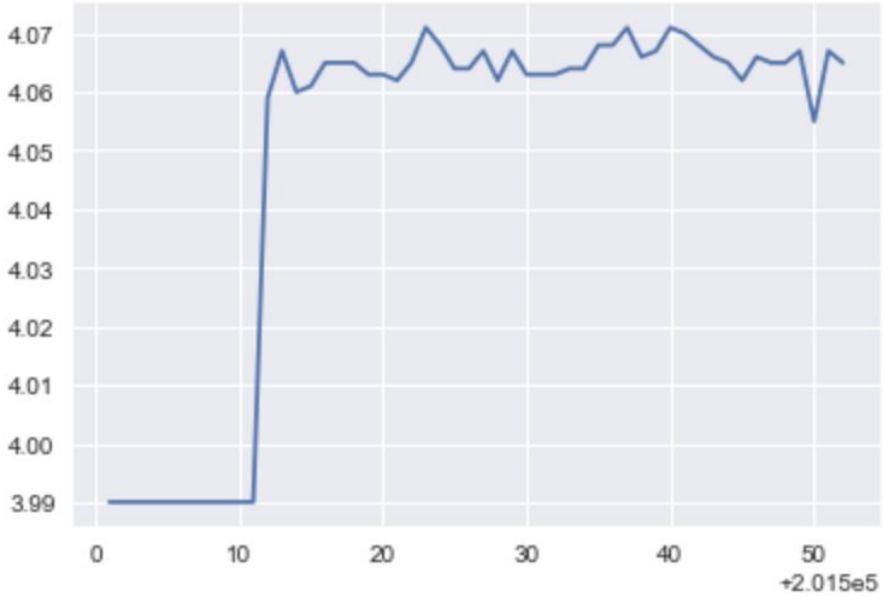
https://www.datascience.com/resources/notebooks/python-dynamic-pricing

Appendix

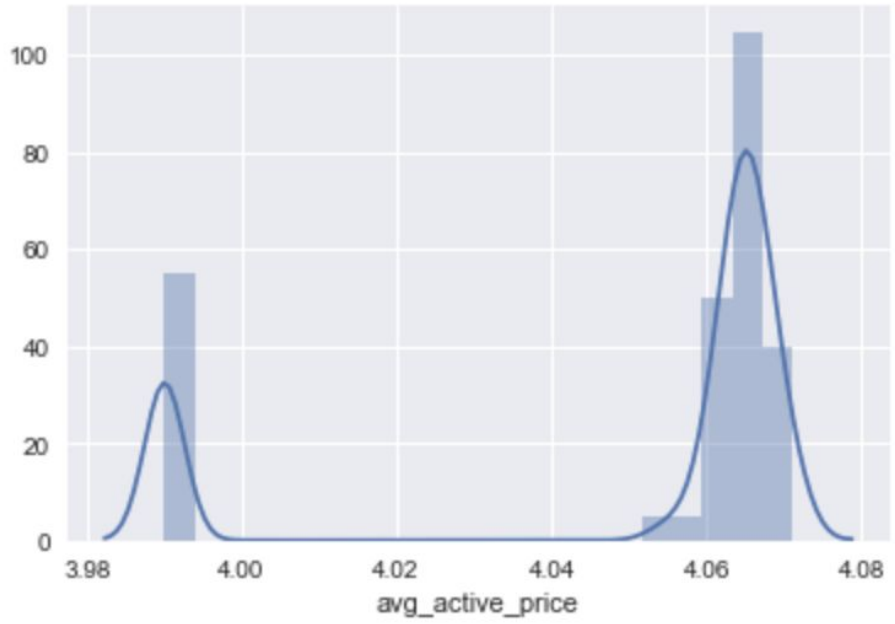
Demand over Time



Price over Time

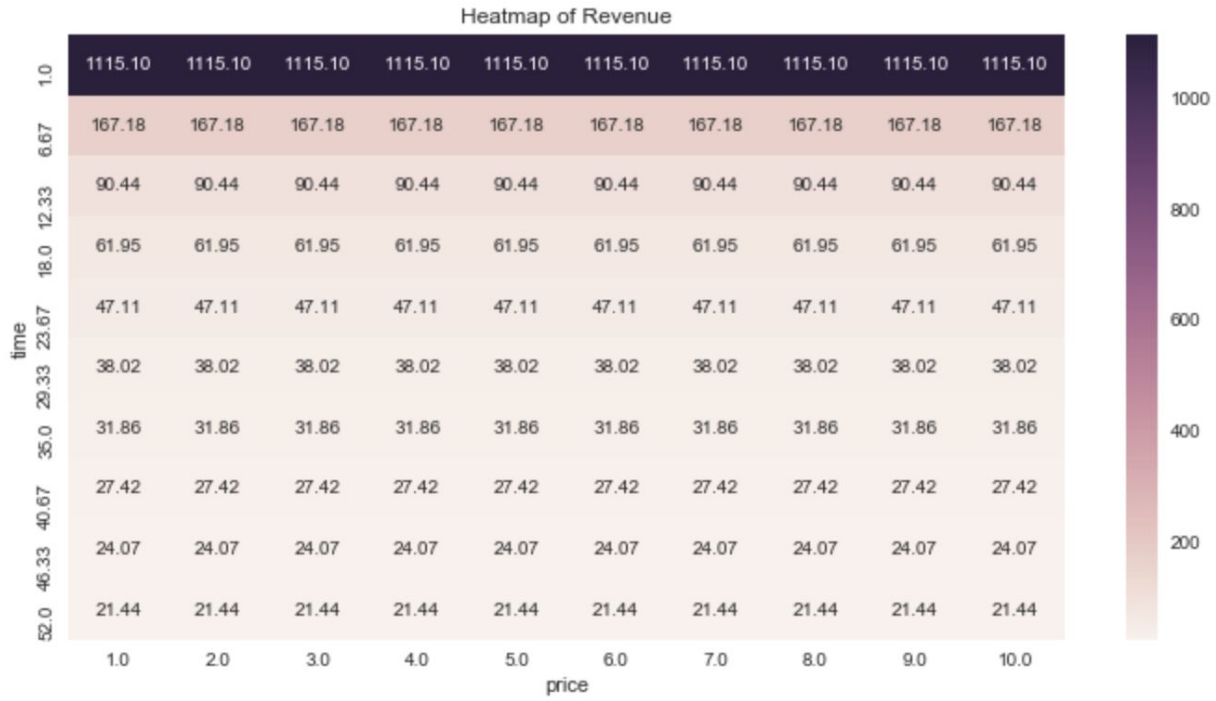


Average Prices Distribution



Heatmap of Demand

Heatmap of Revenue



Model Output

```
In [16]: print(opt_results)

     fun: -3266.0984149681794
     jac: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
 message: 'Optimization terminated successfully.'
    nfev: 13
     nit: 1
    njev: 1
  status: 0
 success: True
       x: array([150., 150., 150., 150., 150., 150., 150., 150., 150., 150.])

In [17]: np.sum(opt_results['x'])

Out[17]: 1499.9999999599413

In [18]: print( price(opt_results['x'], t=t) )

[7.43401346 3.71700673 2.47800449 1.85850337 1.48680269 1.23900224
 1.06200192 0.92925168 0.8260015  0.74340135]
```