EMORY | GOIZUETA BUSINESS SCHOOL

ISOM 680 - Marketing Analytics

Project Report 3: Pricing Models

Team 6

Shuwen Huang, Firat Geyik, Gaows Mohammad, Cecilia Wang

1. Introduction

1.1 Background

ACSE Supermarket, a company that sells everything, has over 40 stores in Lunitunia and sells over 100 thousand products in over 100 categories. ACSE customers can opt to join the Lunie Rewards program to avail of weekly sales and promotions. ACSE regularly partners with suppliers to fund promotions and derives a significant portion of its sales on promotions. While a majority of its promotion activities are in-store promotions, it recently started partnering with select suppliers to experiment on personalized promotions. In theory, personalized promotions are more efficient as offers are only made to targeted individuals who require an offer to purchase a product. In contrast, most in-store promotions make temporary price reductions on a product available to all customers whether or not a customer needs the incentive to purchase the product. The efficiency of personalized promotion comes from an additional analysis required on customer transaction data to determine which customers are most likely to purchase a product to be offered in order to maximize the opportunity for incremental sales and profits.

1.2 Business Problem

Our analytics consulting firm was selected by ACSE (the client) to adjust list (shelf) prices to improve revenue. Although ACSE is in the process of implementing personalized promotions, it understands that certain products are more price sensitive than others with respect to the majority of its customers. With this understanding, ACSE is investigating an everyday-low-price (EDLP) strategy on up to 50 products across all ACSE stores. ACSE also understands that an EDLP strategy will likely require investments in price reductions across the 50 products and would like to fund the investment with price increases on 100 less price sensitive products. Our consulting firm is tasked to investigate the feasibility of such a strategy.

We are to develop pricing models to investigate ACSE's proposed strategy, and our pricing models consider the following factors:

- list/regular price
- promoted price
- product affinity: substitutes and complements
- sales seasonality

Our price change recommendations should be realistic, reasonable and justified with price elasticity measures. Assuming that proposed prices will be in effect for one calendar year, we need to report to ACSE the following:

- a list of 150 products with recommended price changes and justifications
- expected changes in sales quantity, revenue and profitability for each store and overall across all stores

.

2. Data Understanding

2.1 Data Preparation

There are two datasets, transactions dataset and products dataset:

1.transactions.csv contains transaction history from June, 2017 to December, 2020 for over 9 million customers

2. products.csv contains the product to subcategory and category mapping and descriptions for over 100,000 products

For efficiency purposes, we decided to use a sampled transaction dataset with 12,061,594 transaction records. Removing negative sales amount and negative quantity results in a dataset containing 11,907,126 transaction records and 8 columns.

Given the scope of our business question, merging the transactions dataset and the products dataset is needed for further analysis. Our merged dataset contains 11,710,259 rows and 17 columns.

The column “trans_dt” contains timing information. For our seasonality analysis purposes, we decide to take out the year as a single column called “year”.

Next, given that the purpose of our final recommendation is to have a list of 150 products with recommended price changes and justifications, therefore, we removed those products that remained the same price for all of the four year, where we set, if the product maximum price equals the minimum price and equals to the mean price over the 4 years, then this product should not be taken into consideration.

Our final dataset put into use for analysis contains 11,582,680 columns and 19 rows.

The table below shows the result of the number of product IDs at each level, and as a result, there would be 93,074 product IDs that fall into our later analysis.

Total count of unique product IDs in ACSE	152,578
Total count of unique product IDs we take into account	93,074
Count of prod_id with no price change	28,375

3. Product-level Analysis

3.1 Choosing Product Categories

Given that ACSE is investigating an everyday-low-price (EDLP) strategy on up to 50 products across all ACSE stores, therefore, store-level segmentations/analysis is not considered at this time. Instead, we would conduct a product category analysis, as well as calculating elasticity, seasonality, and identifying complements and substitutes, our final recommended price and the 150 products should come from the product category that we choose in this step.

As figure 1 displays, we explored the distribution of sales for all product categories. We could see in our resulting data that 28 of the product categories make up about 80% of sales. These product categories account for a significant share of the sales, which would have the biggest reaction to price changes.

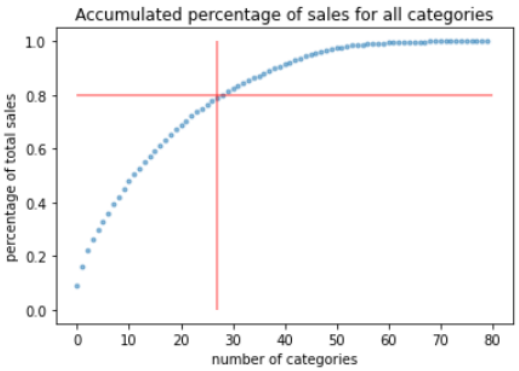

FIGURE 1

The dataset that contains the top 80% sales (28 product categories) contain 6,560,974 rows and 19 columns, which would be the dataset that we calculate for elasticity in the next step.

3.2 Elasticity and Seasonality

It was important that the products we selected to target would be able to adapt its demand well to changes in price. Therefore, for the EDLP(every-day-low-price) products, we want to choose highly elastic products on a weekly level, as these should be the products that are most responsive to changes. If we change the price of these products, demand should also change significantly. These are products that consumers are more price-sensitive towards.

$$\frac{pd'(p)}{d(p)} = \epsilon$$

Given the formula of price-demand elasticity above, we used the average list price (p), average demand (d(p)), and performed *log transformation*[1] for each

[1] Reference: Using log transformation to calculate price elasticity of demand: Calculating Price Elasticity of Demand (sas.com)

product to calculate its average price elasticity, then followed by an elasticity constant linear regression to calculate price elasticity for each product.

Calculating the elasticity would result in two lists of products for our analysis: 50 EDLP (everyday-low-price) products and 100 products that we would increase their prices.

We define an elasticity of smaller than -1 as high elasticity products and would therefore become our list of EDLP products, since a decrease in price would lead to a high increase in demand and would therefore result in higher sales. We stored a list of 50 EDLP products with unique product IDs into a dataset called ela_list2 by having an ascending order of their elasticity.

On the other hand, we defined an elasticity of larger than -0.0001 and smaller than -0.1 as low elasticity products and would therefore become our list of products that we would increase their prices, since these products are not price sensitive, therefore increasing their prices would not lead to a sharp decrease in demand, and if the influence of the increase in price offsets the impact of decrease in demand, the overall sales should actually increase and lead to more profits for ACSE. We stored a list of 100 products with unique product IDs into a dataset called ela_list1 by having a descending order of their elasticity.

To account for seasonality in the data, we recorded the percentage that the sales are deviating from average sales. Seasonality would then become one of the features for our later modeling purpose.

3.4 Substitutes and Complements

Customers' decision on whether to purchase a product is also impacted by the price of its substitutes and complements.

Substitutes are defined as two products that are rarely bought together (eg, Coca-Cola and Pepsi Cola). Therefore, an increase in price should lead to an increase in demand for its substitutes. Complements are defined as two

products that are often bought together. Therefore, there should be a positive relationship towards their demand and sales.

We decide to apply the association rule and apriori algorithm[2] to explore the effects of our selected products' complements and substitutes. For these, we would take the following definitions into account:

Support: The number of transactions that include items in the {X} and {Y} parts of the rule as a percentage of the total number of transaction.It is a measure of how frequently the collection of items occur together as a percentage of all transactions.

Confidence: It is the ratio of the number of transactions that includes all items in {B} as well as the number of transactions that includes all items in {A} to the number of transactions that includes all items in {A}.

Lift: Lift value near 1 indicates X and Y almost often appear together as expected, greater than 1 means they appear together more than expected and less than 1 means they appear less than expected.Greater lift values indicate stronger association. The following table has shown the products with lift higher than 1, which indicates that the antecedents and consequent (both product id) are all complementary products.

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
frozenset({20189092})	frozenset({20009322})	0.002975153	0.067544024	0.000375245	0.126126126	1.867317317	0.000174291	1.067037251
frozenset({20175355001})	frozenset({20007312})	0.003243185	0.059288643	0.000402048	0.123966942	2.090905355	0.000209764	1.073830888
frozenset({20175355001})	frozenset({20016320})	0.003243185	0.048594173	0.000321638	0.099173554	2.040852794	0.000164038	1.056147753
frozenset({20007312})	frozenset({20175355001	0.059288643	0.003243185	0.000402048	0.006781193	2.090905355	0.000209764	1.003562164
frozenset({20016320})	frozenset({20175355001	0.048594173	0.003243185	0.000321638	0.006618864	2.040852794	0.000164038	1.00339817
frozenset({20009322})	frozenset({20189092})	0.067544024	0.002975153	0.000375245	0.005555556	1.867317317	0.000174291	1.002594818

50 EDLP Product Complements and Substitutes

antecedents	consequents	antecedent support	consequent suppc	support	confidence	lift	leverage	conviction
frozenset({20599862})	frozenset({20599700})	0.000264252	0.077329618	0.000216206	0.818181818	10.58044565	0.000196	5.074687102
frozenset({20189092})	frozenset({20312025})	0.003531362	0.039661758	0.000240229	0.068027211	1.715183954	0.0001	1.030435924
frozenset({20312025})	frozenset({20189092})	0.039661758	0.003531362	0.000240229	0.006056935	1.715183954	0.0001	1.002540964
frozenset({20599700})	frozenset({20599862})	0.077329618	0.000264252	0.000216206	0.002795899	10.58044565	0.000196	1.002538746

100 Low Elasticity Product Complements and Substitutes

[2] Reference for association rule and apriori algorithm: Implementing Apriori algorithm in Python - GeeksforGeeks

Using association rules, we have identified the compliments and substitutes for our target products.

4. **Modeling**

   For modeling we chose the linear regression as our predictive model because linear regression can give a number as output for our sales quantity forecast, moreover it is easy to train, implement and use. Furthermore we can use the linear regression to see the expected changes in sales quantity, revenue and profitability for each store and overall across all stores.

   Before moving on to constructing our linear regression model there are two things we need to do, one of them is to populate a column in our final data frame that is called discounted price and another column called cost. To calculate the discounted price we grouped our products by product id and aggregated the unit price for each week in our data set and were able to get mean unit price and max unit price for a given product across weeks and stores. If the given unit price was larger than the mean unit price that meant there was a discount for that given product and we took the mean unit price across weeks as the discounted price.

   To calculate the cost column we had an assumption. Our assumption was that the cost of a given product was the minimum sales price of that product across weeks in our data. The reason we need a cost column for our products is because later in the project we will use that cost to calculate our profit by subtracting the cost from the sales price, that way we can optimize the sales price for a give product by using our model later.

   For our model please see the screenshot below:

```
# we need to do a logit regression
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

```
#elasticity 3, which is our final df, is our train data for the model
train = elasticity4
train
```

	prod_id	week	sales_qty	unit_price	elasticity	seasonality	mean_unit_price	max_unit_price	discounted_price	cost
0	20000481.0	1	0.000000	1.501853	-4.268167	-1.000000	1.314448	1.501853	1.314448	1.095273
1	20000481.0	5	0.000000	1.501853	-4.268167	-1.000000	1.314448	1.501853	1.314448	1.095273
2	20000481.0	6	0.693147	1.383791	-4.268167	-0.434287	1.314448	1.501853	1.314448	1.095273
3	20000481.0	8	0.000000	1.501853	-4.268167	-1.000000	1.314448	1.501853	1.314448	1.095273
4	20000481.0	10	1.386294	1.464027	-4.268167	0.131426	1.314448	1.501853	1.314448	1.095273
...	...	...	...	...	...	...	...	...	...	...
251380	21215367001.0	42	2.302585	1.606233	4.123367	0.232377	1.509230	1.765587	1.50923	1.085189
251381	21215367001.0	43	2.833213	1.606728	4.123367	0.516377	1.509230	1.765587	1.50923	1.085189
251382	21215367001.0	44	2.079442	1.515127	4.123367	0.112947	1.509230	1.765587	1.50923	1.085189
251383	21215367001.0	46	2.772589	1.765587	4.123367	0.483929	1.509230	1.765587	1.50923	1.085189
251384	21215367001.0	47	2.397895	1.738710	4.123367	0.283388	1.509230	1.765587	1.50923	1.085189

251385 rows × 10 columns

```
#fit the model using the respective x values and y value
model = lr.fit(X = train.loc[:,['unit_price', 'elasticity', 'seasonality', 'discounted_price', 'cost']], y = train.sales_qty
```

```
#model coeficient
model.coef_
```

```
array([-8.10435960e-03,  1.31126992e-03,  1.52590836e+00, -7.01732274e-01,
        5.58276476e-01])
```

```
#model intercept
model.intercept_
```

```
2.3587380922012984
```

Basically our target variable (y variable) is the sales quantity as we are trying to predict the amount of sales, and our independent variables are unit price, elasticity, seasonality, discounted price, and cost.

As you can see from the model coefficients, as the unit price for a product increases by one unit the sales quantity decreases by 0.81 unit, and the same logic applies for the remaining coefficients.

Furthermore, as you may find at the bottom part of our python code file we demonstrate a sample case for optimizing the profit by deploying our model. We describe the process we can take to optimize the price of a good that is in our low elasticity product list, the list that we created in the earlier phase of our project, and this list has the products that we hope to raise prices for. We created a temporary

data frame that is grouped by product id table and a mean aggregation of the coefficients of our linear regression model. This temporary table has products and the mean prod_id, sales_qty, seasonality, elasticity, discounted_price, cost across weeks and stores. We use those values and multiply them by the coefficients our model produced earlier above and this would result in a quantity. From there we can plot a graph that has the profit on the y axis and the price on the x axis, and the curve on the graph would help us detect the optimum price for that given product. There we can find the optimum price point for a product. This method would be deployed to detect the forecasted profit as we have the assumed cost for our products and we have a model that can forecast the sales quantity.

Due to some computational run time constraints we were not able to run this application for all of the products in our target price raise and everyday low price list, but this could be automated by the use of for loops and maybe more sophisticated python algorithms, and we could scale this application up.

5. Final Recommendations and Conclusion

```
ela_list2
```

```
array([20000481, 20000584, 20001100, 20001390, 20001792, 20001888,
       20001969, 20002342, 20002659, 20002786, 20003128, 20003315,
       20003749, 20003818, 20004164, 20004494, 20004726, 20005106,
       20005135, 20005571, 20006502, 20006692, 20007312, 20007659,
       20008721, 20008764, 20009282, 20009322, 20009380, 20009533,
       20009748, 20010274, 20010688, 20010752, 20010990, 20011473,
       20011872, 20013256, 20013718, 20013965, 20014406, 20014425,
       20014940, 20015041, 20015206, 20016104, 20016320, 20016394,
       20016661, 20016786], dtype=int64)
```

The list above (ela_list2) has the product IDs that we would want to decrease prices to gain a larger quantity (demand) and therefore increase our profit.

```
ela_list1
```

```
array([20006571, 20013696, 20041578, 20044812, 20048740, 20049268,
       20050766, 20051643, 20062715, 20065191, 20065217, 20077110,
       20078866, 20086819, 20087146, 20093660, 20095560, 20097635,
       20112752, 20113797, 20127030, 20130227, 20144220, 20146901,
       20152032, 20154966, 20155580, 20161456, 20164488, 20166588,
       20169592, 20217072, 20219195, 20219365, 20251602, 20298951,
       20300824, 20301966, 20303482, 20304191, 20305605, 20312025,
       20313480, 20314040, 20314272, 20315210, 20318243, 20319035,
       20322759, 20333488, 20345857, 20357915, 20374621, 20418528,
       20431280, 20437538, 20544858, 20556047, 20562752, 20566120,
       20571486, 20574966, 20585189, 20585545, 20593052, 20599700,
       20624869, 20629496, 20629671, 20629871, 20629885, 20647918,
       20660289, 20672225, 20675802, 20686034, 20686252, 20689950,
       20698091, 20713359, 20720189, 20729432, 20729778, 20732701,
       20745404, 20778516, 20779204, 20783821, 20784073, 20792384,
       20798010, 20810383, 20810413, 20812304, 20814921, 20816790,
       20839110, 20854843, 20856001, 20863284], dtype=int64)
```

The list above (ela_list1) has the product IDs that we would want to increase our prices, though it results in a decrease in quantity (demand), however, the increase in price offsets the decrease in demand and would eventually lead to a higher profit.

The approach we designed is able to select an optimal list price for each product in each store utilizing the price response function from our modeling part above.

In our case, we only used 50 EDLP products and 100 increase-price products across all store levels, however, we believe it is possible to expand this analysis to more product offerings across all stores.

Our recommendation is that: after applying price change on our recommended product, try to have more combinations of products + stores in the future if more datasets are available.