Pricing Optimization Strategy for PunchhMart

Problem Statement

We need to optimize the pricing of 250 product items (see product_sales.csv) and maximize total profits for online grocery store, PunchhMart . From our own analysis, we have found that out of 100 people who visit your website, 5 would
end up making a transaction. From those who made a purchase, we have also obtained the general buying patterns, e.g. their avg units purchased of item_id:1 are 0.6 (see the screenshot below).

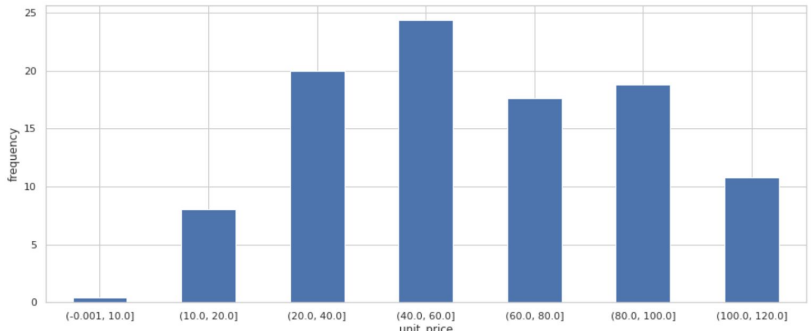
To understand the pricing impact, we have tested different price points for each product. The impact can be broken down into two aspects:
● A lower price increases the volume of the purchased products. For instance, Item 1 might be a frequently used product. If you decrease its price, the customer's conversion rate would increase. The "incr_cvr" column from the CSV file quantifies this price elasticity effect. More detailed definitions below together with the screenshot.
● A decrease in the unit price, however, would also reduce your profit margin on the product.

Therefore, our goal is to find the optimal price points for each item in order to maximize the total profits.

Exploratory Data Analysis

Let's analyze different segments (refer to **ItemSalesPriceEDA.ipynb**)

Price Segments

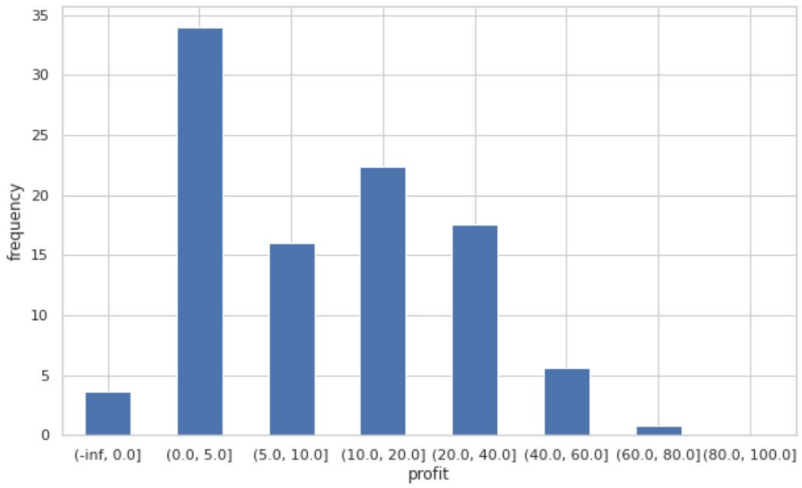


price_frqn_cat	unit_price_min	unit_price_max	unit_price_mean	units_sold_mean	units_sold_sum	incr_cvr_mean	incr_sales_mean	item_id_count
-inf_10	10.0	10.0	10.000000	2.600000	2.6	0.000000	0.049840	1
10-20	11.0	20.0	16.500000	1.430000	28.6	0.147650	0.044942	20
20-40	21.0	40.0	31.040000	1.520000	76.0	0.130060	0.049819	50
40-60	41.0	60.0	50.098361	1.600000	97.6	0.142148	0.056710	61
60-80	62.0	80.0	70.613636	1.584091	69.7	0.145818	0.052468	44
80-100	81.0	100.0	90.361702	1.242553	58.4	0.140830	0.047204	47
100-120	101.0	110.0	105.666667	1.562963	42.2	0.110741	0.048899	27
120-inf	NaN	NaN	NaN	NaN	0.0	NaN	NaN	0

price range (40 - 60)

- contains highest# of items (61)
- highest# of avg units sold (1.6)
- pretty good avg price-elasticity (14.2 % incr_cvr, 5.2 % incr_sales)

Profit Segments



profit_cat	unit_price_mean	units_sold_mean	units_sold_sum	incr_cvr_mean	incr_sales_mean	profit_mean	profit_sum	item_id_count
-inf_0	39.111111	2.300000	20.7	0.088222	0.052581	0.000000	0.0	9
0-10	50.440000	1.069600	133.7	0.136000	0.052868	3.912800	489.1	125
10-20	64.535714	1.775000	99.4	0.143304	0.051083	14.080357	788.5	56
20-40	77.659091	1.929545	84.9	0.144500	0.044404	27.127273	1193.6	44
40-60	90.000000	2.228571	31.2	0.117571	0.056269	48.150000	674.1	14
60-80	103.500000	2.600000	5.2	0.164500	0.031255	70.300000	140.6	2
80-100	NaN	NaN	0.0	NaN	NaN	NaN	0.0	0

- 9 items don’t generate any profit with lowest incr cvr mean value (0.09)

- 125 items have very low avg profit margin ($3.9) and low average units sold (1.07)
- Further analysis needed to understand
    - the root cause of the problem
        - Is there higher procurement / manufacturing cost for these items
        - Is there higher propensity to churn for customers for these items
        - Do these items sell well only in a specific season ?
        - Were there any promotions for these items ? Check the Item age
        - Since these are online grocery items, what are the search demands for these items i.e. are these item_keywords for the brands ranking in Google/Bing ?
        - Are consumers able to find these in ADs or Paid Social posts (in case already being promoted) ? In that case check the bidding strategy and keyword relevancy.
    - and then adopt some strategies to solve the problem
        - Can lowering price yield more profit as this group has moderate incr_cvr ? Or maybe we can afford to increase the price a bit as the mean price already in the lower bracket , units_sold is low and avg incr_sales value looks good.
        - Since these 125 items have significant incr_cvr and incr_sales, they can be promoted (if not done already) in paid social and search engines to boost the cvr
    - Definitely additional marketing signals and user engagement metrics are needed to answer the above questions.

Incremental CVR Segments

incr_cvr_cat	item_idcount	incr_cvr_percmin	incr_cvr_percmax	incr_cvr_percmean
-inf_0	99	0.0	0.0	0.000000
0-5	17	0.2	4.9	2.629412
5-10	18	5.3	9.6	7.188889
10-20	39	10.5	20.0	15.256410
20-40	57	20.6	40.0	29.528070
40-60	19	40.1	57.5	47.447368
60-80	1	61.5	61.5	61.500000

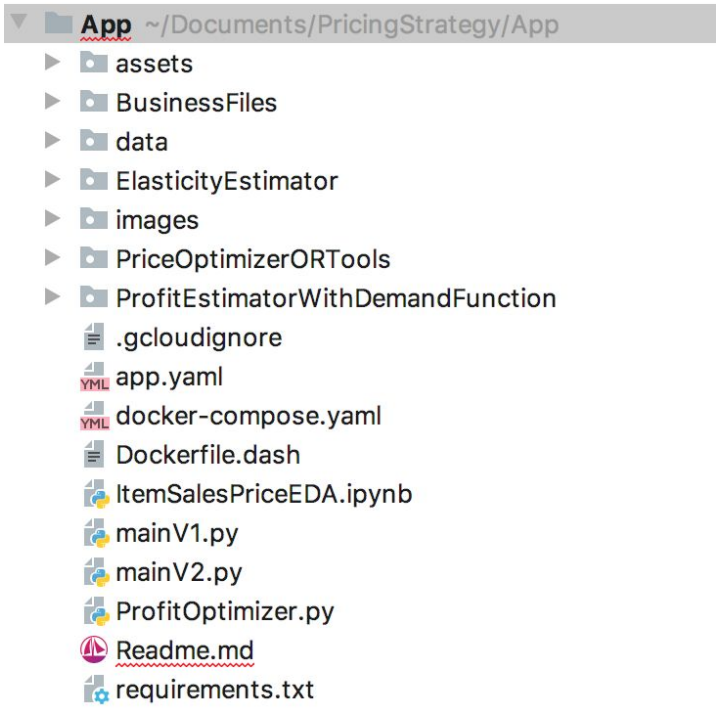
- 99 items do not have any incr_cvr value either due to the fact that these are cold-start items without enough prior history of customer transactions or may be they are not getting enough customer visibility and can be candidates for Customer Acquisition Campaign
- We can take similar measures as mentioned in the strategies for boosting profit for low-profit items

Incremental Sales Segments

incr_sales_cat	incr_sales_perc_min	incr_sales_perc_max	incr_sales_perc_mean	incr_sales_perc_count
0_1	0.037	0.977	0.494192	26
1_2	1.033	1.991	1.499917	24
2_4	2.035	3.916	2.961100	40
4_6	4.033	5.936	4.969017	58
6_8	6.016	7.996	6.980837	49
8_10	8.009	9.943	9.001660	53

- 26 items have very low incr_sales

POC for Profit Optimization



(A) Solve a standard Price Optimization problem

Expression of the optimization problem:

$$
\begin{aligned}
\max \quad & \sum_t \sum_j p_j \cdot d(t,j) \cdot x_{tj} \\
\text{subject to} \quad & \sum_j x_{tj} = 1, \quad \text{for all } t \\
& \sum_t \sum_j d(t,j) \cdot x_{tj} = c \\
& x_{tj} \in 0, 1
\end{aligned}
$$

where t iterates over time intervals, j is an index that iterates over the valid price levels, p_j is the price with index j, $d(t,j)$ is the demand at time t given price level j, c is the inventory level at the beginning of the season, and x_{tj} is a binary dummy variable that is equal to one if price j is assigned to time interval t, and zero otherwise. The first constraint ensures that each time interval has only one price, and the second constraint ensures that all demands sum up to the available stock level. This is an integer programming problem that can be solved using conventional optimization libraries.

© https://blog.griddynamics.com/deep-reinforcement-learning-for-supply-chain-and-price-optimization/

Process Flow:

#Assumptions

Since the optimizer can either maximize or minimize the price values, use maximization technique

```
price_change_multipler = np.array(data.shape[0] * [0.9])
price_lb = [float(i) for i in initial_price*0.9]
price_ub = [float(i) for i in initial_price*1.2]
```

Define the solver

```
solver = pywraplp.Solver('SolveStigler', pywraplp.Solver.GLOP_LINEAR_PROGRAMMING)
```

Set the objective function

```
for i in range(0, len(data)):
  target_price[i] = solver.NumVar(price_lb[i], price_ub[i], str(data_arr[i][0]))
  objective.SetCoefficient(target_price[i], float(price_change_multipler[i]))

objective.SetMaximization()
```

Set the constraints: Sum (price*quantity) >= Initial Profit ($3285.90)

```
constraints.append(solver.Constraint(float(np.sum(initial_profit_vals)),solver.infinity()))
for i in range(0, len(data)):
  constraints[0].SetCoefficient(target_price[i], float(adj_quantity[i]))
```

Demo:

Run: **python PriceOptimizer.py**

Initial total price: $15155.00

Optimal total price: $18186.00
Initial total profit: $3285.90
Optimal total profit: $10929.95

Limitations:

Not sure if custom rules can be specified in OR Tools library
e.g.

- Decrease price for high-profit , high-volume , high incr_cvr items
- Increase price for a low-profit , low volume items
- Increase price for very low incr_cvr value etc.

We can apply custom constraint rules using scikit optimizer which offers some flexibility.

```
opt_var = sco.minimize(min_profit_stat, initial_weights, method='SLSQP', bounds=bnds, constraints=cons)
```

As a next step developed the following simple program with custom rules.

(B) Find the best Profit considering different combinations of price variations

Process Flow:

Profit calculation logic:

- Calculate estimated_price = original_unit_price * (1+change_in_price)
- If there is a price reduction,
  Calculate current incr_sales per tx = incr_sales*abs(change_in_price)*10
  and current incr_cvr = incr_cvr*abs(change_in_price)*10
- If there is no price reduction, set current incr_cvr = 0.05 (default val)
- Calculate estimated_volume = original_units_sold * (1+current incr_sales per tx) * (1 + current incr_cvr)
- Calculate profit_with_price_change_effect = np.sum(estimated_price * estimated_volume) - np.sum(original_cost * estimated_volume)

Assumptions:

- Compute the next best price and volume against base values (original unit_price & units_sold)
- When price is not reduced, consider the default value of incr_cvr i.e. assume there is a cvr increment by 5%

Price Change calculation logic:

- Iterate through all Items
  - Apply maximum reduction for products with higher propensity to boost sales and conversion i.e. for different buckets of higher incr_sales and incr_cvr values
  - Increase price slightly if there is no incr_cvr / incr_sales and units_sold is high and unit_price is high
  - Decrease price if there is no incr_cvr / incr_sales and units_sold is comparatively lower and unit_price is low as well

- **Note:** many different combinations of price reduction rules can be created by blending high-price, high-volume, low-priced, low-volume items with the conversion rate factor and sales demand values.

Assumptions:

- **Price change boundary {-10%,20%}**

Main Workflow:

- Invoke Price Change calculation logic
- Check if new profit is better than old profit
- If not, keep iterating
    - Find new price_change and new profit
- If new profit and old profit are same then the process has converged

Assumptions:

- retain the price changes even if it doesn't yield better profit

Demo

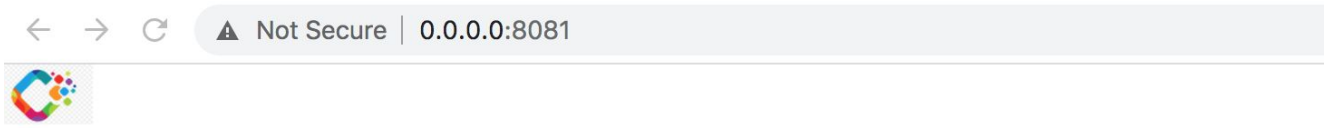
Prerequisites: Build Docker Image docker-compose build

```
kaniska_mac$ docker-compose up -d
Starting Pricing-Strategy ... done

kaniska_mac$ docker-compose ps
     Name            Command        State          Ports
-------------------------------------------------------------------------
Pricing-Strategy   python mainV2.py   Up      0.0.0.0:8081->8088/tcp
```

Profit without Optimization: 3285.90
Net Estimated Profit: 4772.95 , LIFT by [**45.26 %**]
Increase in Total Pricing: 879.47, [**5.80 %**]
Increase in Total Units Sold: 18.76 %, [**5 %**]

Web App



Upload Retail Data

Drag and drop or click to select a file to upload.

Optimal Price & Profit

› Original Retail data: product_salesV2.csv

Profit without Optimization: 3285.90

Net Estimated Profit: 4772.95 , LIFT by [45.26 %]

ncrease in Total Pricing: 879.47, [5.80 %]

ncrease in Total Units Sold: 18.76, [5.00 %]

item_id	unit_price	est_price	units_sold	est_volume	original_profit	est_profit
1	95	94.525	0.6	0.63	1.8000000000000043	1.5907499999999999
2	37	44.4	0.8	0.8400000000000001	8	14.615999999999996
3	34	33.83	0.6	0.63	4.799999999999999	4.9329
4	32	31.84	0.7	0.735	1.3999999999999986	1.3523999999999994
5	62	74.39999999999999	1	1.05	17	30.86999999999999
6	86	103.2	0.9	0.9450000000000001	7.200000000000003	23.814000000000007
7	105	105	1.9	1.9949999999999999	53.20000000000002	55.860000000000014
8	84	83.58	0.9	0.9450000000000001	15.300000000000004	15.668100000000003
9	69	67.965	1.8	1.8900000000000001	28.799999999999997	28.283850000000015

The above algorithm is a very rudimentary approach towards understanding the effect on sales by changing the price of a SKU and computing an optimal profit.

Now let's switch gears and focus on how we can create a Demand Function

(C) Create the demand function an item and find the maximum profit from predicted values

Expression of a generic demand function:
price decrease can create a temporary demand splash, while price increase can result in a temporary demand drop.

$$d(p_t, p_{t-1}) = d_0 - k \cdot p_t - a \cdot s((p_t - p_{t-1})^+) + b \cdot s((p_t - p_{t-1})^-)$$

where

$$x^+ = x \text{ if } x > 0, \text{ and } 0 \text{ otherwise}$$
$$x^- = x \text{ if } x < 0, \text{ and } 0 \text{ otherwise}$$

and p_t is the price for the current time interval and p_{t-1} is the price for the previous time interval. The first two terms correspond to a linear demand model with intercept d_0 and slope k. The second two terms model the response on a price change between two intervals. Coefficients a and b define the sensitivity to positive and negative price changes, respectively, and s is a shock function that can be used to specify a non-linear dependency between the price change and demand. For the sake of illustration, we assume that $s(x) = \sqrt{x}$.

© https://blog.griddynamics.com/deep-reinforcement-learning-for-supply-chain-and-price-optimization/

Process Flow

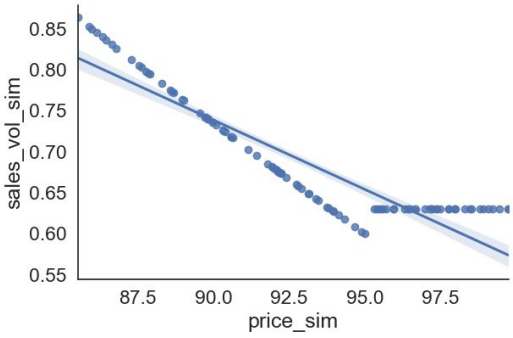
Code: **DemandFunctionProfitEstimator.py**

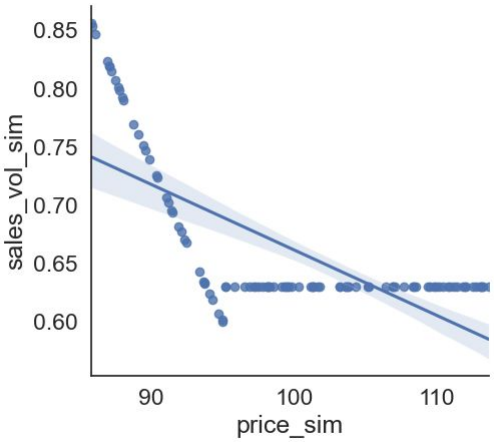
```
itemA_price_lb = itemA_base_price*0.90
itemA_price_ub = itemA_base_price*1.2
itemA_df = pd.DataFrame.empty
N = 100
#  Generate 100 random price for the item => random.uniform(itemA_price_lb, itemA_price_ub)
price_sim = [random.uniform(itemA_price_lb, itemA_price_ub) for i in range(0,N)]
```

we calculate simulated profit for every time step for Item by applying the same logic as mentioned in Scenario (B)

```
# plot demand curve
```

These curves shows the relationship between simulated price and simulated sales volume






```
# fit OLS model
model = ols("sales_vol_sim ~ price_sim", data = itemA_df).fit()
# print model summary
print(model.summary())
```

R-squared: 0.558 indicating it explains 55.8% of model's variability
p-value is 0 indicating the the Null Hypothesis that increase in price doesn't reduce sales volume can be rejected

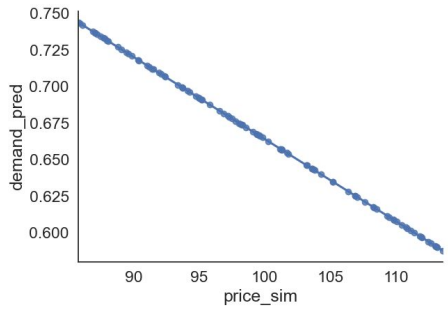
Intercept 1.3110 , price_sim coefficient -0.0065

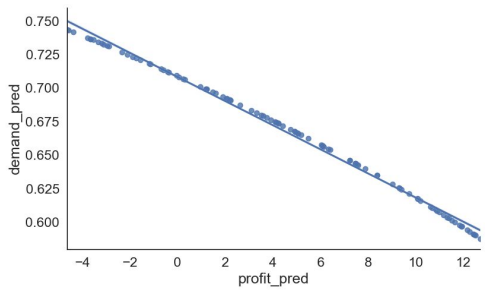
We can now compute the demand function by *plugging in regression coefficients: C= Intercept, S = Price Coeff* as follows *demand_pred = C - S * price*

We can now apply the demand function on a new set of price ranges for the Item.
For the sake of brevity, applying on the simulated price values

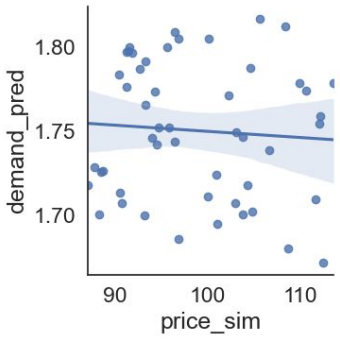
```
itemA_df['profit_pred'] = itemA_df.demand_pred * (itemA_df.price_sim - itemA_df.fixed_cost)
```

We now fit the Demand Function against the same training dataset just for checking





Next we fit the curve on Test data (50 samples). As expected we don't get a very good result as the Demand Function and OLS model is not optimized !




```
max_val = itemA_df2[itemA_df2['profit_pred'] == itemA_df2['profit_pred'].max()]
```

Limitations:

- This is just to explore the concept of Demand Function and show that if we can experiment with different price ranges for an item we can find a model with good coeff for determination and then use
- We haven't removed any outlier values as we observed in EDA phase and didn't change the price and quantity to log scale (log-log is recommended for Demand Function)
- we didn't run additional iterations with different sample size , hence we didn't get a better Coeff of Determination

Future Improvements

(A) Improve the custom price variation logic

The price variation logic as demonstrated in Approach-(B) in the POC section currently considers only one set of Business Rules with one specific range of threshold values for incr_cvr param.

It can be further improved by running the Optimizer with different threshold values for different groups (incr_cvr group, profit group, sales_volume group) and then compute the max profit from the output of different combinations after removing outliers.

(B) Improve Demand Forecast using Price Elasticity

Price elasticity of demand (PED) is a measure used in economics to show the responsiveness, or elasticity, of the quantity demanded of a good or service to a change in its price when nothing but the price changes. More precisely, it gives the percentage change in quantity demanded in response to a one percent change in price.

© https://towardsdatascience.com/price-elasticity-data-understanding-and-data-exploration-first-of-all-ae4661da2ecb

In marketing, it is how sensitive consumers are to a change in price of a product.
It gives answers to questions such as:

- "If I lower the price of a product, how much more will sell?"
- "If I raise the price of one product, how will that affect sales of the other products?"
- "If the market price of a product goes down, how much will that affect the amount that firms will be willing to supply to the market?"

Find the best Elasticity Coefficient based on simulated time series data.
Good amount of money are spent on promotions and campaigns

Estimating the demand against a price change can help us find the ROI of a promotion

```
'elasticity_l' = np.log(quantity_delta/quantity) / price_delta/price)
"units_sold_l' = np.log[units_sold]
"price_l' = np.log[unit_price]
```

We can run GLM to generate a Normal distribution

```
with Model() as model:
        GLM.from_formula('units_sold_l ~ 'elasticity_l', df_l, family='normal')
        trace = sample(4000, cores=4)
```

These observed data samples will now be our elasticity values which we obtained from GLM.
Note that we really need a considerable amount of observations!

Now we can configure and run Bayesian Model and create a distribution of the possible values of the elasticity conditions based on the observed and prior data.

Reference:
https://towardsdatascience.com/using-bayesian-modeling-to-improve-price-elasticity-accuracy-8748881d99ba

(C) Improve Demand Generation by considering additional signals like is_promotion, is_weekday , is_holiday, weekly change in demand, elasticity, log-price ratio

I have taken an excerpt from this notebook just to demonstrate how we can improve our prior Demand Function by considering various other features
© https://github.com/shumingpeh/price-optimization/blob/master/ea-price-optimization-model.ipynb

```
X = sm.add_constant(X)
y = self.itemA[['amount']]
model = sm.OLS(y, X).fit()

$\beta_{0} + \beta_{1}P_{i} + \beta_{2}PromotionPresent + \beta_{3}isWeekDay +
\beta_{4}isWeekEnd + \beta_{5}isHoliday + \beta_{6}PreviousPeriodDemand + \epsilon$

model.predict(X)

constant_A = ( model_param_A_df.const.values[0] + is_weekday *
model_param_A_df.is_weekday.values[0]
constant_numeric = np.array([[-1*constant_A],[-1*constant_B],[-1*constant_C]])

item_A_first = model_param_A_df.weighted_price_usd_A.values[0] * 2
values = np.linalg.solve(model_variables_matrix, constant_numeric)

original_demand_A = (
  model_param_A_df.const.values[0]
  + model_param_A_df.is_weekday.values[0]
  + model_param_A_df.previous_day_demand.values[0] * 16.44
  + model_param_A_df.weighted_price_usd_A.values[0] * original_price_A
  + model_param_A_df.weighted_price_usd_B.values[0] * original_price_B
  + model_param_A_df.weighted_price_usd_C.values[0] * original_price_C
)
new_demand_A = (
```

```
  model_param_A_df.const.values[0]
  + model_param_A_df.is_weekday.values[0]
  + model_param_A_df.previous_day_demand.values[0] * 16.44
  + model_param_A_df.weighted_price_usd_A.values[0] * 17.88810439
  + model_param_A_df.weighted_price_usd_B.values[0] * 41.13139183
  + model_param_A_df.weighted_price_usd_C.values[0] * 81.72909101
)
original_total_revenue = (
  original_demand_A * original_price_A
  + original_demand_B * original_price_B
  + original_demand_C * original_price_C
)
new_total_revenue = (
  new_demand_A * 17.88810439
  + new_demand_B * 41.13139183
  + new_demand_C * 81.72909101
)
```

(D) Long-term vision - leverage relevant Marketing Signals and adopt Experimentation Strategy

Maximizing profits through price variation requires understanding of how the Items behave in the Market and Customer purchase process.

In the grand scheme of things; Item price, Item inventory, Item Ad spend, Item score, Item keyword rank in search-engine , Customer search intent , User-Item engagement metrics are all related to each other !

- Can we apply different price variation logic to different segments of products ?
- Is there any specific order of priority to promote items based on discount, newness and rating ?
- Is there a specific Price Mark Down / Up Business requirement ?
- Is the ATC / GMV for an item low even when more people are searching for it ?
- Is the Item rank in Search Engine is low even when clicks / impressions are high ?
- What is the optimal discount value for a certain group of products ?
- Are the items being categorized in correct catalogs ?
- Do we have Item Basket data ? Then we can apply a similar discount to the related group of items and boost sales of a group of items together.
- Certain types of Items are good candidates for New Customer Sign Up campaign

- Is a certain category Items showing the signs of Propensity to churn ? then they may be candidates for Customer Retention Campaign.
- Are the items being catered to Local Demand by leveraging location-specific discounts and demographic properties
- What is the customer satisfaction rate for the items ?
- Does an item have any seasonal pattern ?
- What is the current age and life-cycle phase of the Item ? Is it a Cold-start or long-lived item ?
- Are these items being marketed as per latest search trends and demands ?
- Are the items tuned to all types of faceted properties e.g. brand, gender, color, size ?
- Are we keeping track of OOS to avoid bouncing and opportunity loss ?
- Are we boosting perishable with shorter shelf life ?

  There are various tools which provide access to different types of data based on a specific requirement !

  E.g. Adobe Omniture => User Engagement metrics, eMarketeer => Ad spends, B2B, DSP , Flight Deck => supplier data, IRI => top brands, top items , Nielsen => POS data, Braze => Marketing attribution data , Brightedge => competitive keywords , SEMRush => Keyword Search Ranks , Google Ad => Keyword Ad ranks

Presence of different categories of Items and possibilities to combine multiple signals offer a great opportunity for implementing A/B Tests (Market-driven or simulated) and select right set of combinations of parameters to optimize profit.