Data, People, and Systems: Pricing & Recommendations Project Report

Khaled Abughoush, Hoang Dang, Vincent Ferraiuolo
thethreemusketeers
Cornell University
New York, NY, 10011
{KA363, HMD62, VPF6}@cornell.edu

Part 1.1: Demand Estimation

1. First give an overview of your approach -- what was your high level strategy?

In this project, our objective was to predict prices in order to maximize sales revenue based on previous customer demand data. While we are presented with the purchasing history of all customers, the challenge with this part lies in two components: **1)** we have data on past interactions on our website for only half of the customers, in the form of the "noisy" embedded vectors. **2)** The second challenge in this project was the time constraint where our model needs to predict the users demand in less than ½ second. In other words, there is a tradeoff between more complex models (which might be able to better predict users' demands) and time.

Our team focused on predicting missing user vectors based on data from users with known embedding vectors using kNN. Following this, by using matrix multiplication between the user vectors and the item vectors, we got the predicted ratings of each customer for each item. Finally, we combined predicted ratings with user covariates and historical prices from *train_prices_decisions.csv* to model the probability (demand) of each customer buying item 0 or item 1 at different pricing combinations.

2. How did you evaluate your approach? Include figures/tables as necessary

To evaluate our model, we performed cross-validation to find the best parameters for our kNN model. By using 10-fold cross-validation, we determined that the best number of nearest neighbors for kNN is 78 (Figure 2-Appendix). We believe our model was performing well since it ranked 2nd place in class (Figure 1-Appendix).

3. What other things did you try or consider, and why did you choose the approach you did?

In our analysis, we wanted to incorporate as much data as possible, so we did not want to just ignore the user embeddings because some customers don't have them. We considered using another model other than kNN to predict the unknown user vectors, however, we ultimately decided that our final model (which we used in the competition) will offset the prediction errors from the kNN model.

4. For demand estimation, how (if at all) did you overcome the cold start problem? How did you evaluate how well you did? If you chose KNN, what was the distance function you used, and why?

In this project we suffer from the cold start problem because we don't have any historical interaction data for the new customers (i.e, transaction history, browsing history, etc…). Since there are no embedding vectors, it's challenging to find recommendations on the user's product preferences. To overcome this, we fit a kNN regressor on user covariates and their known embedding vectors. Following that, we used user covariates as an input for our kNN regressor to predict the missing user vectors. In doing so, we assumed that users with similar covariates will tend to like the same products.

While implementing the kNN regressor, our team used cross validation again to test the impact of using Euclidean, Manhattan, Chebyshev, and Minkowski as distance functions. As a result of Euclidean distance having the highest R^2 (Figure 3-Appendix), we adopted it for our final model.

5. For demand estimation, what machine learning methods did you try? Do they differ in (time, programming) complexity? Was there gain in fancier models?

During our analysis, we experimented with a few machine learning methods. First, we experimented with logistic regression, which is a simple model, but retains low run times while

maintaining accurate results. Next, we experimented with random forests, which is a more complex and accurate, but slower method. To elaborate, when scoring the trained models, logistic regression is less accurate than random forests (0.88 vs 0.934), but the random forests model takes more time to output one prediction (1.5s vs 0.03s). As a result, we decided to use our logistic regression model given its drastic runtime advantage and minimal accuracy loss.

Part 1.2: Price Optimization

1. First give an overview of your approach -- what was your high level strategy?

Once we have estimated the demand for all users, we are able to set prices for the products and maximize revenue. In other words, the demand estimation model outputs the probability of a customer with certain covariates buying item 0 or item 1 at a price combination (which is demand). For a set of prices (p_0, p_1) we let $d_0(p_0, p_1)$ and $d_1(p_0, p_1)$ be the fraction of people who buy items 0 or 1. As such, the expected revenue for each customer from the both items can be summarized in the following formula:

$$\textbf{Expected [Revenue]} = p_0 \times d_0\ (p_0, p_1) + p_1 \times d_1\ (p_0, p_1)$$

To identify the optimal price combinations for items 0 or 1, we leveraged a grid-based search approach. We started with a coarse grid that goes from 0 to 6, in increments of 1. We chose this range based on the minimum and maximum prices of the training dataset, adding some leeway to account for values beyond what is in the training set. For example, if we found [a,b] from the coarse grid to be the revenue-maximizing pair of prices for a customer, then we search using a finer grid for each coarse price pair ([a ± 0.7, b ± 0.7] in increments of 0.1) to find the final revenue-maximizing price pair.
By using the grid-based search approach we were able to drastically reduce the computation time for each customer.

2. How did you evaluate your approach? Include figures/tables as necessary

We calculated the score for the logistic regression model (0.88), which proved to be a good fit, based on the outcome of part 1 (Figure 1-Appendix).

3. What other things did you try or consider, and why did you choose the approach you did?

We started out doing a brute-force search combining every single price combination from 0 to 10 in increments of 0.1, and used the demand estimation model from the previous part to predict the probability of each customer buying each item at each price combination. In return, we'd get the expected revenue at every price point and find the revenue-maximizing one this way. Eventually, we implemented the grid-based search approach because the brute-force approach would take too long and not satisfy the ½ second constraint.

4. Optional: For pricing, how did you choose your optimization method, stopping criterion, etc.? How did the time constraint play a role in your choices? What hardship (theoretical, computational, etc.) did two-dimensional pricing bring, what did you do to overcome them? How did you evaluate whether your price optimization found good prices? Did you consider "robustness"?

Our main strategy focused on meeting the ½ second computation per customer constraint. Testing every single price combination would be computationally expensive. Therefore, we stopped at 0.1 increment as going in finer increments (0.01, 0.001, etc.) would have too expensive of a trade-off between

time and accuracy. We don't believe that going into the order of 0.01 would make a significant impact on our model. In addition, we later in part 2 addressed robustness by outputting the top 20 revenue-maximizing prices instead of just the best one. This helps account for the probability that the best few prices will not result in a sale.

5. Optional: Suppose the first covariate was a sensitive feature, and you do not want to discriminate based on it. How do your prices vary by this first covariate? Suppose now you simply removed the first covariate from your model...does that help? How would you go about removing the effect of things that correlate with the first covariate?

Simply removing the first covariate would not help, because of the potential correlation between it and the second and/or third covariate, or even the embedding vectors. If the data is biased, the model can still encode that information using other features. A possible approach is to actively model this bias, to measure how much it impacts our predictions, and then subtract the effect of this bias from our outcome.

Part 2: Competition

1. First give an overview of your approach -- what was your high level strategy?

We used the same models from part 1 to output the top 20 revenue-maximizing prices for each customer, but also took into account our ongoing competition performance to adjust these prices. Using our prices and the competition results (from whom did the customers buy from), we fitted a classification model to predict the probability of the next sale being ours (for each of these top 20 price pairs) and selected the one with the highest probability.

2. How did you evaluate your approach? Include figures/tables as necessary

We benchmarked against the dummy_fixed_prices_adaptive many times and we also benchmarked against different variations/parameters of our own model. As time progressed, our model continued to outperform the dummy_fixed_prices_adaptive until the profit gap was at ~30% (Figure 4-Appendix). In addition, we compared different machine learning models to use during the competition. For instance, we fitted a logistic regression vs SVC where the logistic outperformed the SVC (Figure 5-Appendix).

3. What other things did you try or consider, and why did you choose the approach you did?

We also considered predicting the opponent's prices during the competition as additional features for the above model, but decided against it as making predictions from predictions could have significant variance.

Some specific things that you may want to discuss include, for example:

1. How did you deal with the tradeoff of exploration-exploitation? i.e, you want to win as many customers as possible right from the start, but you would also want to gain more knowledge about your opponent by strategically designing your output, so that you stand a higher chance of winning in later rounds.

During this stage we faced a second "cold start" problem where we wanted to predict the probability of winning the sale, but we started with no historical data. As a result, our team "sacrificed" the first 200 rounds, where we only returned random prices to gauge how our opponent behaves, as well as to build our initial training dataset. After that, we begin to train the newly collected data using logistic regression to output actual competitive prices.

2. How did your agent take into account the feedback provided after each customer?

After every round, we stored the results (our prices and who did the last customer buy from) and the covariates of the last customer in one dataset and retrained our logistic regression model. Once the size of this dataset went past 1000, we randomly sampled only 1000 entries from it to retrain our model, thereby keeping the size of the training data consistently at 1000, because we did not want to overfit the model.

3. How did you deal with the tradeoff of higher potential revenue (high prices) versus higher chances of winning customers in each round (low prices)?

At first, we knew we did not want to just output lower prices to win as many games as possible since our objective was to maximize revenue. More importantly, we did not want to compromise the quality of the data that we use to train our competition model. If all of the data we have to train on are "wins" then we run into the issue of class imbalance, effectively making our model worthless. To prevent this, we output random prices first as an attempt to create a balanced training dataset, thus also relying on a bit of luck.

4. How did the competition objective (maximizing overall revenue, not winning # of games or median revenue) affect your strategy?

We used the demand estimation model to output the top 20 price pairs that would give the highest revenue, instead of just one. Using the competition model, we predicted the probability of the next sale being ours for each of these top 20 price pairs. Out of the top 20 price pairs, we select the one with the highest probability. If the model predicts that none of the top 20 pairs would be our sale, then the model outputs random prices as an attempt to further improve the training data for our model. This way, we maximize the probability of the next sale being ours, with minimal compromise in revenue, because outputting one of the top 20 revenue-maximizing prices guarantees a certain level of revenue.

5. What assumptions did you make about your opponents' agents? Did you have different strategies for different agent "types"? (that you could detect from their strategies?)

We assumed that the other teams would occasionally throw in random data points (i.e. too high or too low prices) in order to throw off others' algorithms. We accounted for this by randomly sampling our training data from the historical performance data that we continue to collect as the competition goes on. Ultimately, this should help us reduce the chances of our model being trained on high-variance, random data points.

Conclusion

- Reflect upon your performance. If you were satisfied with your results, discuss what you did well and what in particular likely helped the most. If you were not satisfied, what could have been done differently?

Overall, our team is satisfied with our algorithm's performance. Even though our more complex algorithm scored slightly higher than our simpler model, the processing time advantage of the simpler model was too good to pass over (considering the 50x improvement). With more computation power, we could have trained and fitted more complex and accurate models at each of the stages.

- What else would you like to do related to pricing or recommendation, but was not included in the project scope?

We would have loved to have learned about reactive pricing based on our competitors. In other words, when is it advantageous to match competitor prices? As a business, is it worth matching competitor prices if most of our customers are local and we don't have much competition? Will other competitors pop up locally as a result?

- Did this project spark your interest in anything beyond the class that you would like to learn more about afterwards?

We are very interested in learning about how large corporations manage prices based on their competitors. For example, is Amazon willing to lose some revenue and charge lower prices in order to gain more lifelong customers/Prime subscribers? What impact do these memberships have on item pricing, consumer retention, and competition?

In the end, given what we've learned in class, our group was happy with our results and our experience during the project.

Appendix

team_name	revenue	expected_revenue	number_sales_0	number_sales_1
teamsvm	4334.553	4289.279	1221	1470
thethreemusketeers	4309.4	4151.414	1287	1420
datapeople	4251.037	4119.667	1263	1331
tjmin	4236.616	4148.034	1296	1378
pricegouge	4204.874	4172.643	1368	1335
pgrk	4172.16	4076.71	1260	1385

Figure 1: Top 6 teams performance based on revenue generated

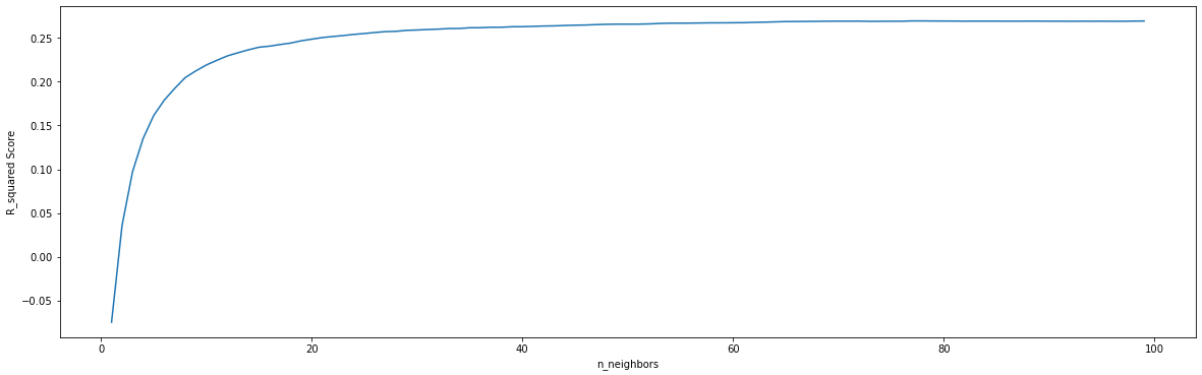

Figure 2: R^2 values to identify the optimal n neighbors using cross-validation

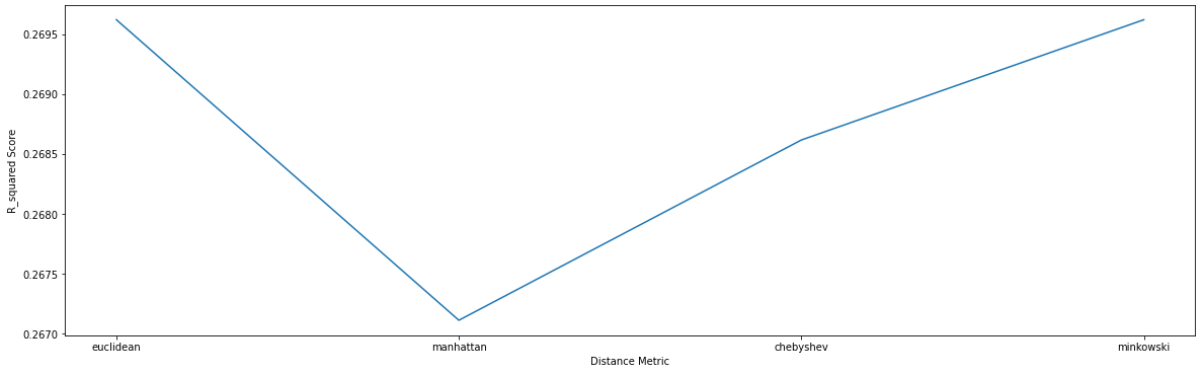

Figure 3: Comparing the R^2 value for Euclidean, Manhattan, Chebyshev, and Minkowski distance functions

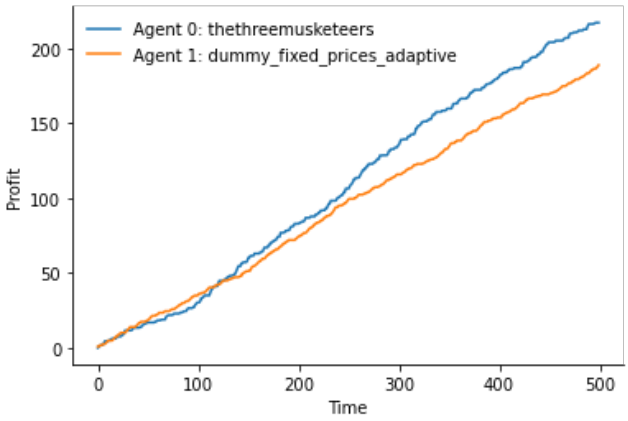

Figure 4: Profit generated for thethreemusketeers model and the benchmark model

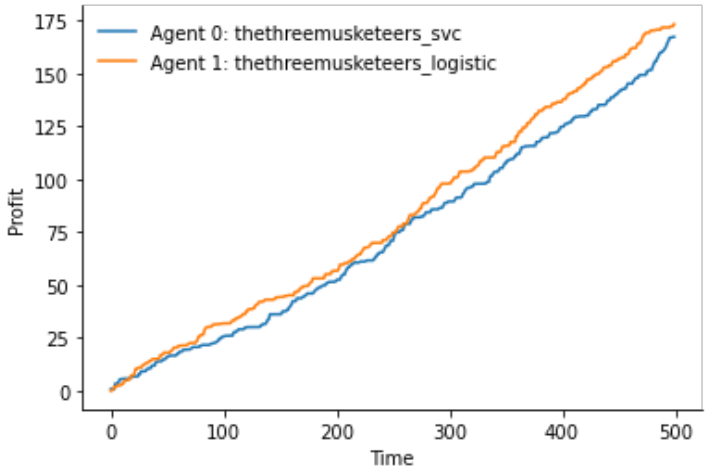

Figure 5: Profit generated for the logistic and SVC model