Recommender Systems for Instacart

CS 6220 Final Project
Team 7

Nakul Camasamudram, Rosy Parmar, Rahul Verma, Guiheng Zhou

December 13, 2017

1 Introduction

Instacart is an American company that operates as a same-day grocery delivery service. Customers select groceries through a web/mobile application from various retailers which are then delivered by a personal shopper. Clearly, with data on customers product purchases, a recommender system can be designed using unsupervised learning methods.

In this project, we would like to investigate if products suggested by a recommender system is more indicative of user purchases on Instacart as compared to recommending the 10 most popular products to each user. To address this question, we use a grocery shopping dataset released by Instacart in 2017. [1].

2 Experiment Design

In order to perform our analysis, we conduct the following steps:

(1) Collect Data: For our base data, we use 'The Instacart Online Grocery Shopping Dataset 2017' which is available for public download[1]. The dataset contains a sample of over 3 million grocery orders from more than 200,000 Instacart users.

(2) Missing Values and Outliers: There are no missing values in the dataset. Also, there are no obvious outliers as every user has placed at least 4 orders with multiple products in each order.

(3) Data Transformations and Partitions: We split the dataset, which is spread over 3 '.csv' files, into previous and current purchases for every user. The prior purchase data is used to build a utility matrix where rows are products, columns are users and each entry u_{ij} is the number of times product i appears in user u's purchase history. We use current purchases of 20% of the users in the dataset as the test set.

[1] https://www.instacart.com/datasets/grocery-shopping-2017

(4) Perform data mining analysis: To carry out our investigation, we use and compare the performance of three different collaborative filtering techniques: a neighborhood-based method that recommends products to a target-user based on the purchases of similar users, a latent factor method that uses Singular Value Decomposition(SVD) and a matrix factorization method that was designed specifically for implicit feedback datasets.

(5) Evaluate methods: Each of the above three methods are evaluated on an independent test dataset consisting of the current purchases of around 26,000 users, and compared to a baseline model that recommends the 10 most popular products to each user. We can now determine if the products suggested by our recommender systems result in more purchases on Instacart as compared to the baseline model.

3 Data Mining Analysis

3.1 Term Frequency-Inverse Document Frequency (TF-IDF) based neighborhood

We explore a neighborhood method on a TF-IDF weighted utility matrix that suggests recommendations to a target user based on products purchased by other users who share the same taste.

Why TF-IDF?
TF-IDF is often used to assess the importance of a word in a document. A word will gain its importance as its frequency in a document increases, while it will lose importance as the frequency of documents containing this word increases. Given a query, documents (a bag of words) related to the query (another bag of words) can be generated from a corpus, regardless of the order of words. Thus, we made an analogy. The purchase history of a user can be treated as a document and each product the user purchased is just like a word in the document. To produce recommendations for a target user (a query) who'll receive the recommendations, we can find similar users (documents), and recommend the products of similar users that the target user didn't purchase.

What is a TF-IDF matrix?
Each column of the matrix is a term, more specifically a product users purchased. Each row of the matrix is a document, representing for the purchase history of a user (hereinafter also denoted as **UPH**), comprised of varied products.

How does TF-IDF work?

1. *Term Frequency* matrix is generated based on the utility matrix by counting the frequencies of products for each row (user purchase history).

2. *Inverse Document Frequency* vector is generated for each product p by applying division between the number of users and the number of users purchasing p, as shown below

$$idf_p = \log(\frac{\#(UPH)}{\#(UPH\ containing\ p)}).$$

3. *TF-IDF* matrix then can be produced by multiplying *TF* matrix with *IDF* vector.

4. *Cosine Similarity* vector is calculated based on *TF-IDF* matrix for all the pairs of the target user and the other users, as shown below

$$cosine\_similarity(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

   where $\vec{i}$ and $\vec{j}$ are two row vectors representing purchase histories of two users.

5. *Top K Users similar to the target one* is selected by extracting the largest K values from Cosine Similarity vector (in our experiment, $K = 20$).

6. *Top N products of similar users* are recommended according to overall sales from candidate recommendations, which are consisting of the least number of the most similar users and of which the size is larger than or equal to N (in our experiment, $N = 10$).

3.2 Matrix Factorization using Singular Value Decomposition(SVD)

Why Matrix Factorization for recommendations?
Since our utility matrix is more than 99.8% sparse, we explore matrix factorization using SVD to uncover latent features that might help generate better recommendations.

How SVD for recommendations?
We use SVD to generate a low rank approximation of the utility matrix with 50 and 100 latent factors. Below, A is the preference Matrix for (product,user) U (product factors) and V (user factors) are the latent singular vectors and D is the latent singular values.

$$A = UDV^T$$

Product preferences for a user i can be found out by below formula. Using $A^T[i]$ we recommend K most preferred products which were not previously bought by that user.

$$A^T[i] = UDV[i]$$

Drawbacks
SVD considers explicit data where the user has rated both things they like and dislike, but since our data has the count of product purchased by a user, we cannot treat low values in the Utility Matrix as dislikes. Also, SVD treats all missing values as unknown. However, since the data is about purchase of a product, we may have to treat a user not buying a product as being a signal that the user might not like that product instead.

3.3 Matrix Factorization for Implicit Feedback

The Instacart dataset consists only of implicit feedback data in the form of past and current grocery orders of users. Implicit feedback data is different from explicit feedback data such as ratings in a few important ways:

1. There is a lack of negative feedback. A user that did not purchase a certain product might have done so because he dislikes the product or just because he did not know about the product.
2. Implicit feedback is inherently noisy. A user's purchase of a product does not necessarily indicate a positive view. It may have been purchased as a gift, or perhaps the user was disappointed with the product.

Hence, we implement a matrix factorization method proposed by Hu et al.[2], which was designed specifically for datasets with implicit feedback data. Their method has two key characteristics:

1. Implicit feedback data is treated as binary preferences with higher confidence placed on observations that include a greater number of interactions. Frequencies of user-item interactions r_{ui} are separated into two quantities with distinct interpretations: preferences p_{ui} and the confidence of preferences c_{ui}. Preferences p_{ui} are binary variables that indicate a user u's preference for item i. They are derived by binarizing r_{ui} values. We transform our original utility matrix R_{ui} into a matrix with each cell representing the confidence of user's preference for an item as follows:

   $$C_{ui} = 1 + \alpha R_{ui}$$

   where α represents a linear change in confidence of preferences as r_{ui} changes.

2. Similar to other matrix factorization methods, the goal is then to find user-factor vectors $x_u \in \mathbb{R}^f$ and item-factor vectors $y_i \in \mathbb{R}^f$ for each user and item, so that a prediction of the confidence of user u's preference for item i is $x_u^T y_i$. The factorization method suggested in the paper accounts for varying confidence levels for all user-item pairs in the dataset, by minimizing the below cost function in linear running time, using an Alternating Least Squares approach.

   $$\min_{x,y} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

4 Results

4.1 Evaluation

We generate a list of products to each user. Then, we present a prefix of the list to the user as the recommended products. It is important to realize that we do not have reliable feedback regarding which products are disliked, as not purchasing a product can stem from multiple different reasons. Thus, precision based metrics are not very appropriate, as they require knowing which products are undesired to a user. However, purchasing a product is an indication of liking it, making recall-oriented measures applicable.

$$\text{Mean Recall} = \sum_{\text{test users}} \frac{|\{\text{recommended}\}| \cap |\{\text{actual}\}|}{|\{\text{actual}\}|}$$

We generate recommendations for every user in the test dataset, compute the Mean Recall and compare each of our recommender systems against a baseline model that recommends the 10 most popular products to each user. *Table 1* documents the results of our methods.

Table 1: Final Results of methods as compared to the baseline

	Mean Recall (%)	Running time(minutes)
Baseline	2.62	-
TF-IDF Neighborhood	20.08	169
MF using SVD with 50 factors	2.84	12
MF using SVD with 100 factors	2.39	16
MF for Implicit Feedback using ALS	4.13	2

4.2 Analysis

TF-IDF Neighborhood To our surprise, this method performs 8 times better than the baseline model, which might mean that users product purchases are greatly dependent on purchases made by other similar users. A possible explanation for this could be Instacart itself using a neighborhood based approach to suggest products to user.

The recall of this method is also highly dependent on the selection of candidate recommendations, a superset of final ones. Currently, we iterate the top K similar users from the most to the least similar ones, we extend the candidate recommendations with the set difference of product set of similar user and that of target user. Once the size of candidate recommendations exceeds N, the iteration halts. Finally, select the top N products based on overall previous sales in the dataset from candidate recommendations, which guarantees high similarities.

What will happen if candidate recommendations included all the products purchased by top K similar users and not by target user? With the same method to select top N products, it will lead to undesirable recall score, being merely around mean recall of baseline. This is resulting from the loss of similar users. Given a large candidate recommendations ($size \gg N$) formed by all the products of top K similar users, there's great possibility that filtered by overall sales, the top products are actually purchased by the users not the most similar to the target one.

SVD SVD with 50 factors is just slightly better than the popularity based baseline model. This result was expected because the data we have is implicit, but SVD on a raw utility matrix relies on data to be explicit.

MF for Implicit Feedback We expected this method to be better than the baseline model and SVD as it weighs the utility matrix in a way suited for implicit feedback data. Hu et al[2] achieved better results as compared to neighborhood based models as well. However, for the same reasons as highlighted above for TF-IDF neighborhood, users seem to be interested items that other similar users have purchased. Also, it is important to note that this method is 85 times faster than the neighborhood method due to the linear running time of the Alternating Least Squares optimization method.

5 Discussion

5.1 Performance

It is clear from the evaluation of our recommender systems that even a simple Model like SVD performs better than recommending just the popular products. To answer our question we posed initially: Yes, personalized recommendations based of purchase history are more indicative of user purchases on Instacart as compared to recommending the 10 most popular products to each user.

5.2 Improvements and Further Research

Collaborative Filtering methods suffer from cold-start problems, due to their inability to address products and users new to the system. To address this, a hybrid approach in tandem with content-based methods would work best. Consequently, collecting more data that would help profile users and items would be an important step. Also, other approaches like Learning to Rank, Deep Neural Networks and Social Recommendations could be explored.

References

[1] “The Instacart Online Grocery Shopping Dataset 2017”, Accessed from https://www.instacart.com/datasets/grocery-shopping-2017

[2] Y.F. Hu, Y. Koren, and C. Volinsky, “Collaborative Filtering for Implicit Feedback Datasets,” Proc. IEEE Int’l Conf. Data Mining (ICDM 08), IEEE CS Press, 2008, pp. 263-272.

Appendix

Data Analysis

The Dataset

Orders from Instacart are available in four .csv files: "orders.csv", "order_products__train.csv", "order_products__prior.csv" and "sample_submission.csv". The key to understand the dataset and the train / test split is the orders table ("orders.csv").

Take for example User 1[Fig 1] , who happens to be a train user. User 1 has 10 prior orders, and 1 train order whose details are provided in "order_products__prior.csv" and in "order_products__train.csv" respectively.

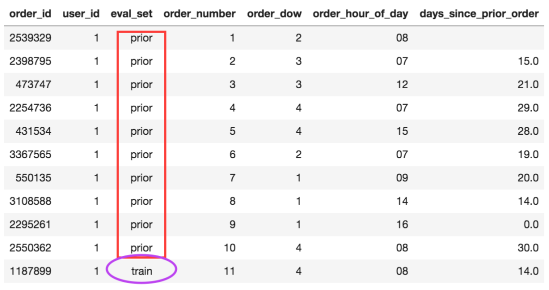
Similarly, User 4 is a test user. He has 5 prior orders, and his 6th is a test order. Their details are available in "order_products__prior.csv" and "sample_submission.csv" respectively.

Figure 2 is a glimpse at "order_products__prior.csv" when merged with three other .csv files that represent products, aisles and departments. The format of "sample_submission.csv" and "order_products__train.csv" is exactly the same.

Exploration and Analysis

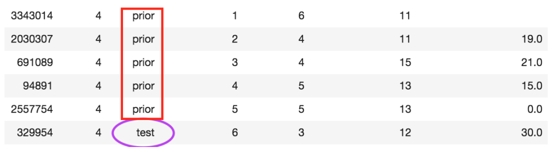
How many orders have users placed?

The below histogram validates the claim that 4 to 100 orders of a customer are given.



order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
2539329	1	prior	1	2	08	
2398795	1	prior	2	3	07	15.0
473747	1	prior	3	3	12	21.0
2254736	1	prior	4	4	07	29.0
431534	1	prior	5	4	15	28.0
3367565	1	prior	6	2	07	19.0
550135	1	prior	7	1	09	20.0
3108588	1	prior	8	1	14	14.0
2295261	1	prior	9	1	16	0.0
2550362	1	prior	10	4	08	30.0
1187899	1	train	11	4	08	14.0

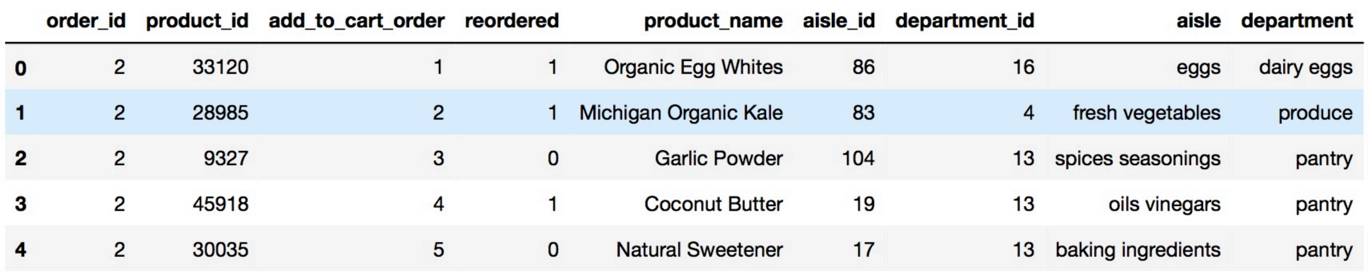
(a) User 1



3343014	4	prior	1	6	11	
2030307	4	prior	2	4	11	19.0
691089	4	prior	3	4	15	21.0
94891	4	prior	4	5	13	15.0
2557754	4	prior	5	5	13	0.0
329954	4	test	6	3	12	30.0

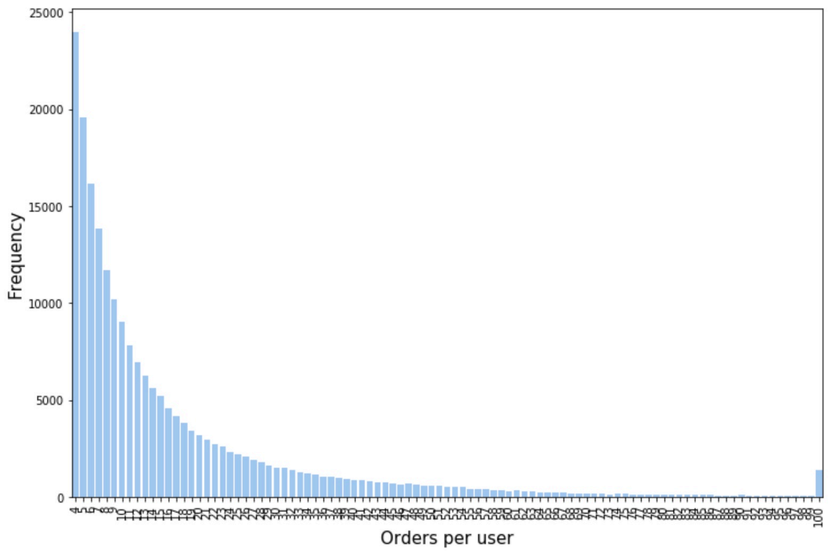
(b) User 4

Figure 1: Train/Test Split

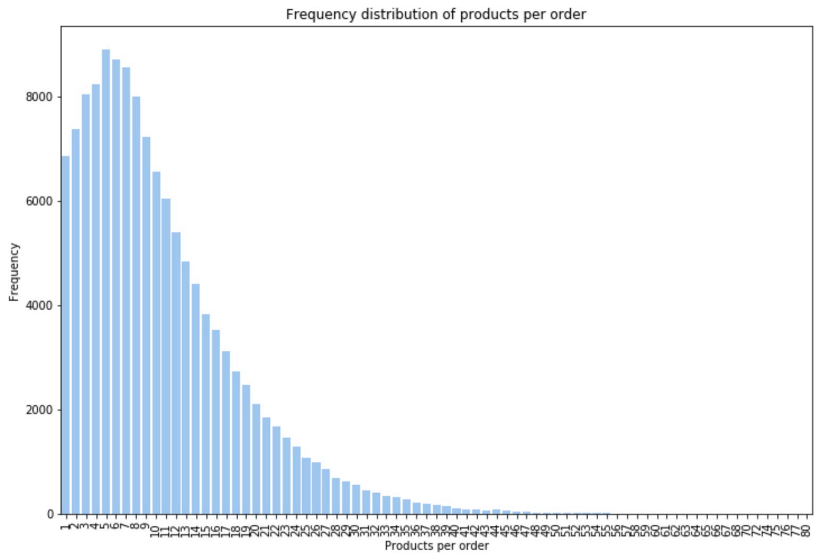


	order_id	product_id	add_to_cart_order	reordered	product_name	aisle_id	department_id	aisle	department
0	2	33120	1	1	Organic Egg Whites	86	16	eggs	dairy eggs
1	2	28985	2	1	Michigan Organic Kale	83	4	fresh vegetables	produce
2	2	9327	3	0	Garlic Powder	104	13	spices seasonings	pantry
3	2	45918	4	1	Coconut Butter	19	13	oils vinegars	pantry
4	2	30035	5	0	Natural Sweetener	17	13	baking ingredients	pantry

Figure 2: Merged Prior Orders

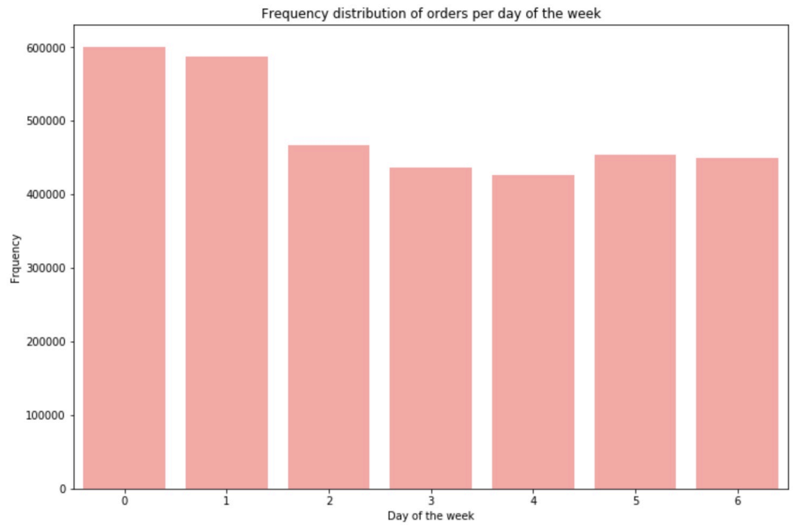


How many products does an order have??
The "long tail" phenomenon is clearly visible here.



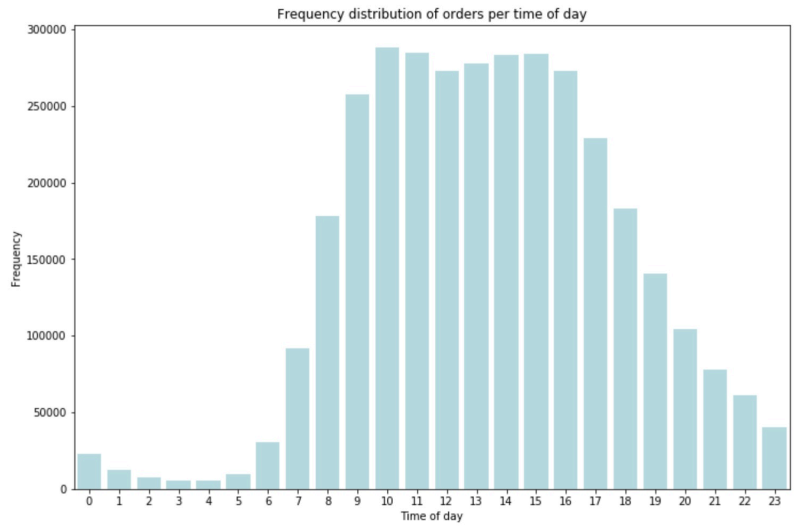
Does day of the week influence user order habits?

There is a clear effect of day of the week. Most orders are on days 0 and 1. However, there is no information about which values represent which day, but, it's reasonable to assume they are weekends.



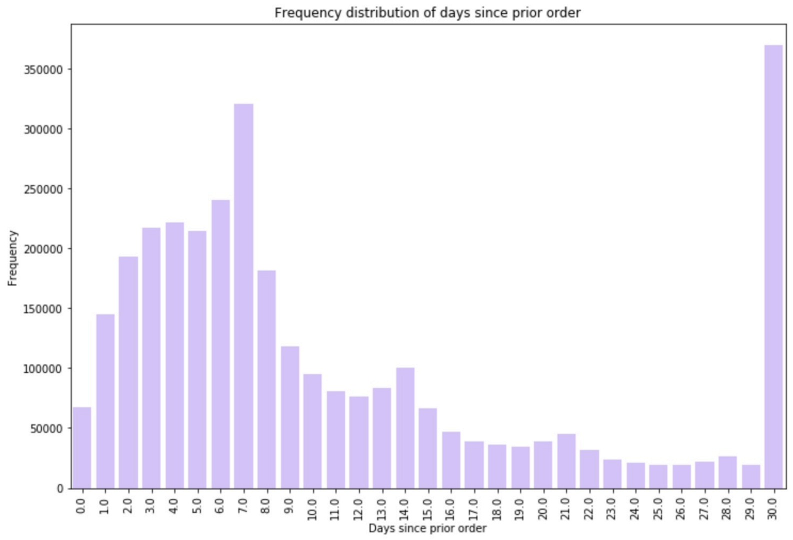
Does time of day influence user order habits?

There is a clear effect. Most of the orders are placed between 7.00 am and 10.00pm.

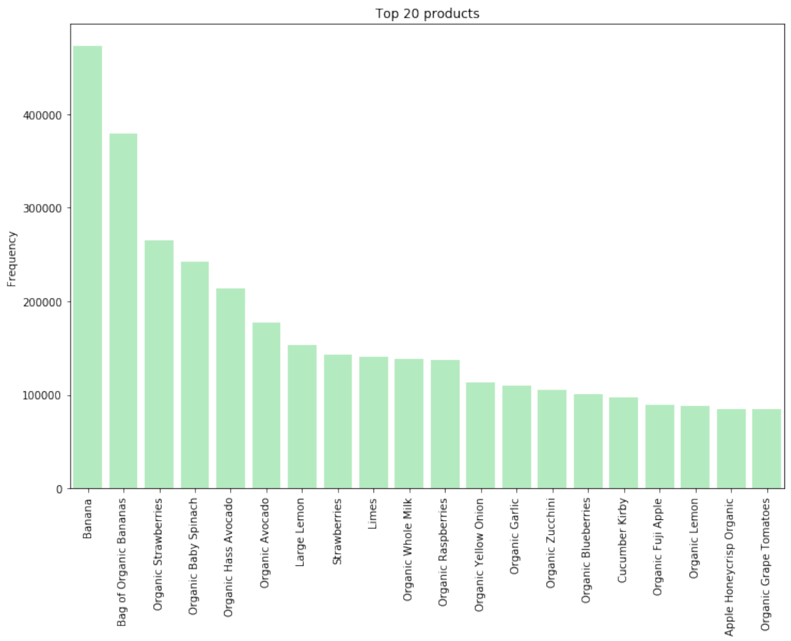


When do users reorder?

Users seem to order on a weekly and monthly basis.



What are the top 20 Products?


Top 20 products
Frequency
Banana
Bag of Organic Bananas
Organic Strawberries
Organic Baby Spinach
Organic Hass Avocado
Organic Avocado
Large Lemon
Strawberries
Limes
Organic Whole Milk
Organic Raspberries
Organic Yellow Onion
Organic Garlic
Organic Zucchini
Organic Blueberries
Cucumber Kirby
Organic Fuji Apple
Organic Lemon
Apple Honeycrisp Organic
Organic Grape Tomatoes