Capstone Project 2 Final Report

Predicting the Products an Online Grocery Shopper Will Purchase Again

Yi Li

1. Introduction

Online grocery shopping is growing rapidly these years. According to the U.S. Online Grocery Survey 2020, released on May 6 2020, one half (52.0%) of all respondents had bought groceries online - more than double the shopper numbers from two years ago. The coronavirus pandemic is transforming consumers' needs and behaviors, and has encouraged more grocery shoppers to start buying or buying more online.

There are many grocery delivery apps in the market today such as Instacart, Shipt, Amazon prime now, and Walmart grocery delivery etc. Features that help customers' shopping experience more easy and efficient will make the app stand out from others. Correctly predicting customers' shopping behavior using machine learning, and incorporate it into the features of the apps will make their consumers' shopping experience more pleasant.

In this project, I am going to use a dataset from a Kaggle competition to predict the products a customer will buy again. This dataset is provided by Instacart. This dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, 4 to 100 of their orders are provided, with the sequence of products purchased in each order. The week and hour of day the order was placed are also provide, and a relative measure of time between orders.

Dataset: https: "The Instacart Online Grocery Shopping Dataset 2017", Accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on <2020/05/>

2. Exploratory Data Analysis

2.1 Basic Structure of the Datasets

The Instacart Online Grocery Shopping Dataset consists of 7 datasets, in which "order_products__train" is held of Kaggle for analyzing the result of competition.

The structure and description each dataset are as follow:

column	description	dtype
aisle_id	aisle identifier	integer in [1:134]
aisle	the name of the aisle	string

Table 1. Basic structure of aisles.csv

aisle_id	aisle
1	prepared soups salads
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat preparation

Table 2. Head of aisles.csv

column	decription	dtype
department_id	department identifier	integer in [1:21]
department	the name of the department	string

Table 3. Basic structure of department.csv

department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol

Table 4. Head of department.csv

column	decription	dtype
product_id	product identifier	integer in [1:49688]

product_name	name of the product	string
aisle_id	aisle identifier	integer
department_id	department identifier	integer

Table 5. Basic structure of products.csv

product_id	**product_name**	**aisle_id**	**department_id**
1	Chocolate Sandwich Cookies	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Unsweetened Oolong Tea	94	7
4	Smart Ones Classic Favorites Mini Rigatoni Wit...	38	1
5	Green Chile Anytime Sauce	5	13

Table 6. Head of products.csv

column	**decription**	**dtype**
order_id	order identifier	integer in [1: 3421083]
user_id	customer identifier	integer in [1: 206209]
eval_set	which evaluation set this order belongs in	category(prior/train/test)
order_number	the order sequence number for this user (1 = first, n = nth)	integer in [1:100]
order_dow	the day of the week the order was placed on	integer in [1:7]
order_hour_of_day	the hour of the day the order was placed on	integer in [0:23]
days_since_prior	days since the last order, capped at 30 (with NAs for order_number = 1)	float in [0:30] or NA

Table 7. Basic structure of orders.csv

order_id	**user_id**	**eval_set**	**order_number**	**order_dow**	**order_hour_of_day**	**days_since_prior_order**
2539329	1	prior	1	2	8	NaN
2398795	1	prior	2	3	7	15.0
473747	1	prior	3	3	12	21.0
2254736	1	prior	4	4	7	29.0
431534	1	prior	5	4	15	28.0

Table 8. Head of orders.csv

column	**decription**	**dtype**
order_id	order identifier	integer

product_id	customer identifier	integer
add_to_cart_order	order in which each product was added to cart	integer
reordered	1 if this product has been ordered by this user in the past, 0 otherwise	integer(0/1)

Table 9. Basic structure of orders_products__prior.csv and order_products__train.csv

order_id	product_id	add_to_cart_order	reordered
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1

Table 10. Head of orders_products__prior.csv

order_id	product_id	add_to_cart_order	reordered
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1

Table 11. Head order_products__train.csv

After importing all the dataset as pandas DataFrames, the only DataFrame that has NaN is the "order" DataFrame. The "days_since_prior_order" value of each user_id's first order is NaN. There is no other missing values in the DataFrames.

2.2 Exploratory Data Analysis and Statistical Inference

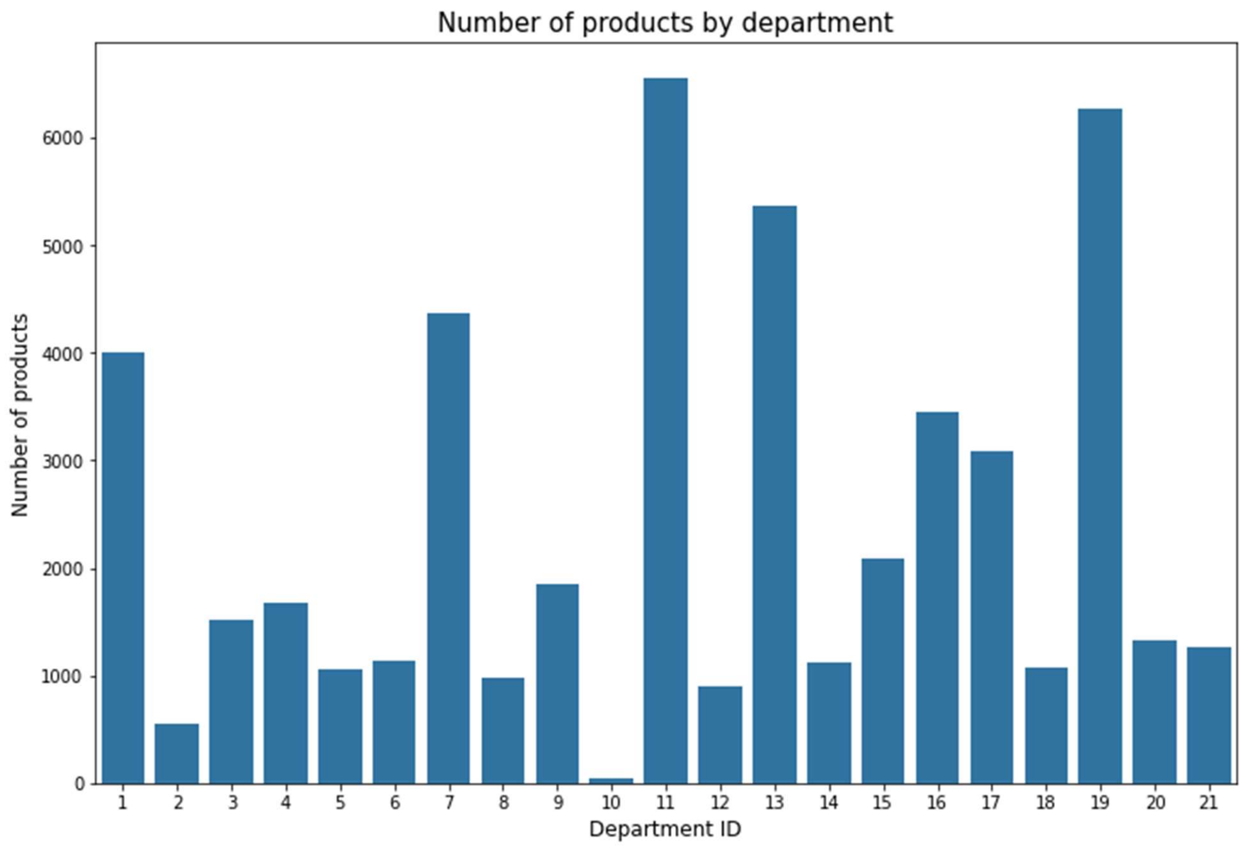

FIGURE 1. Number of products in each department.

By plotting the number of products in each department, we can see that department of personal care and department of snacks have the most types of products, and department of bulk has least types of products.

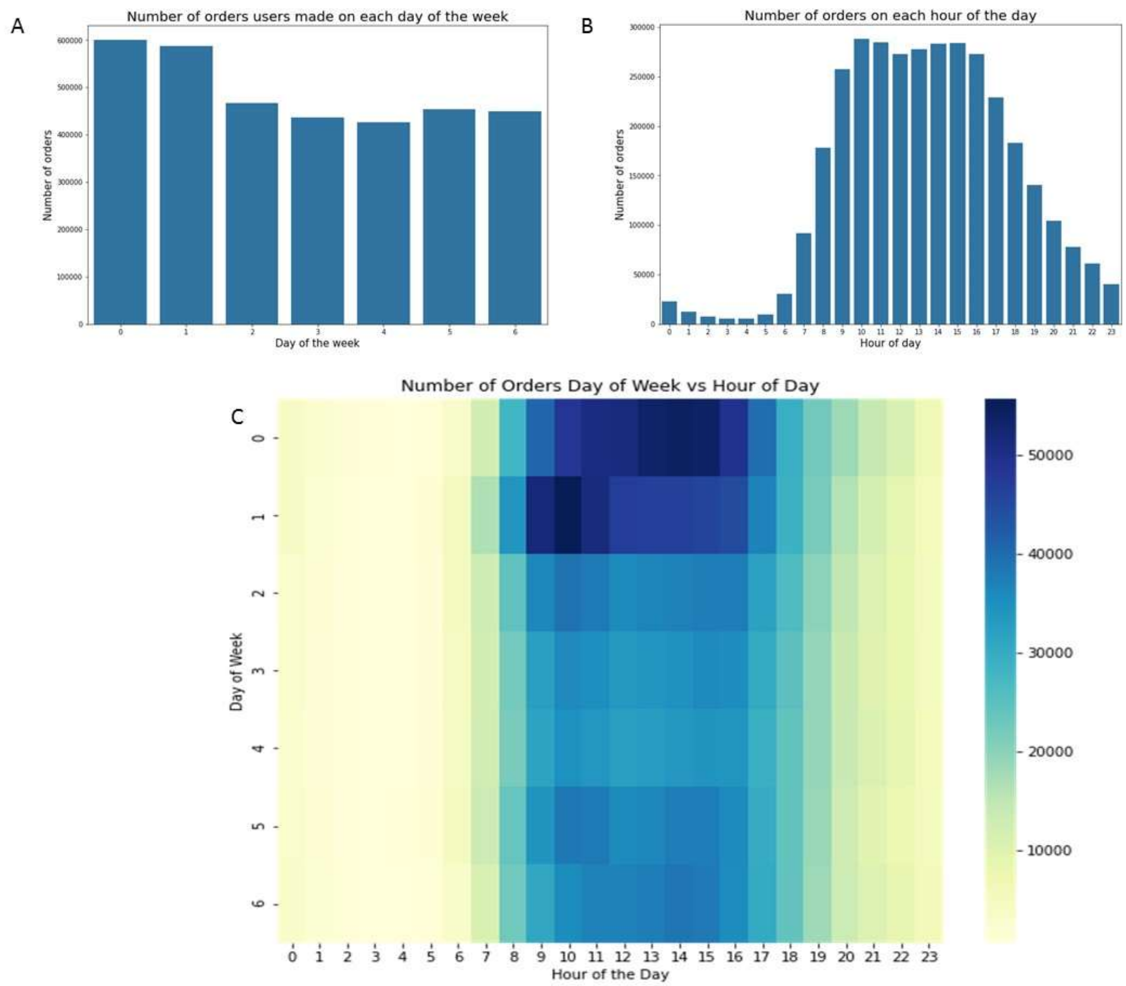

FIGURE 2. Number of products by time

By plotting the number of orders placed on each day of a week and each hour of a day, we can see that between 9AM and 16PM on Saturday and Sunday is the most popular time to place orders.

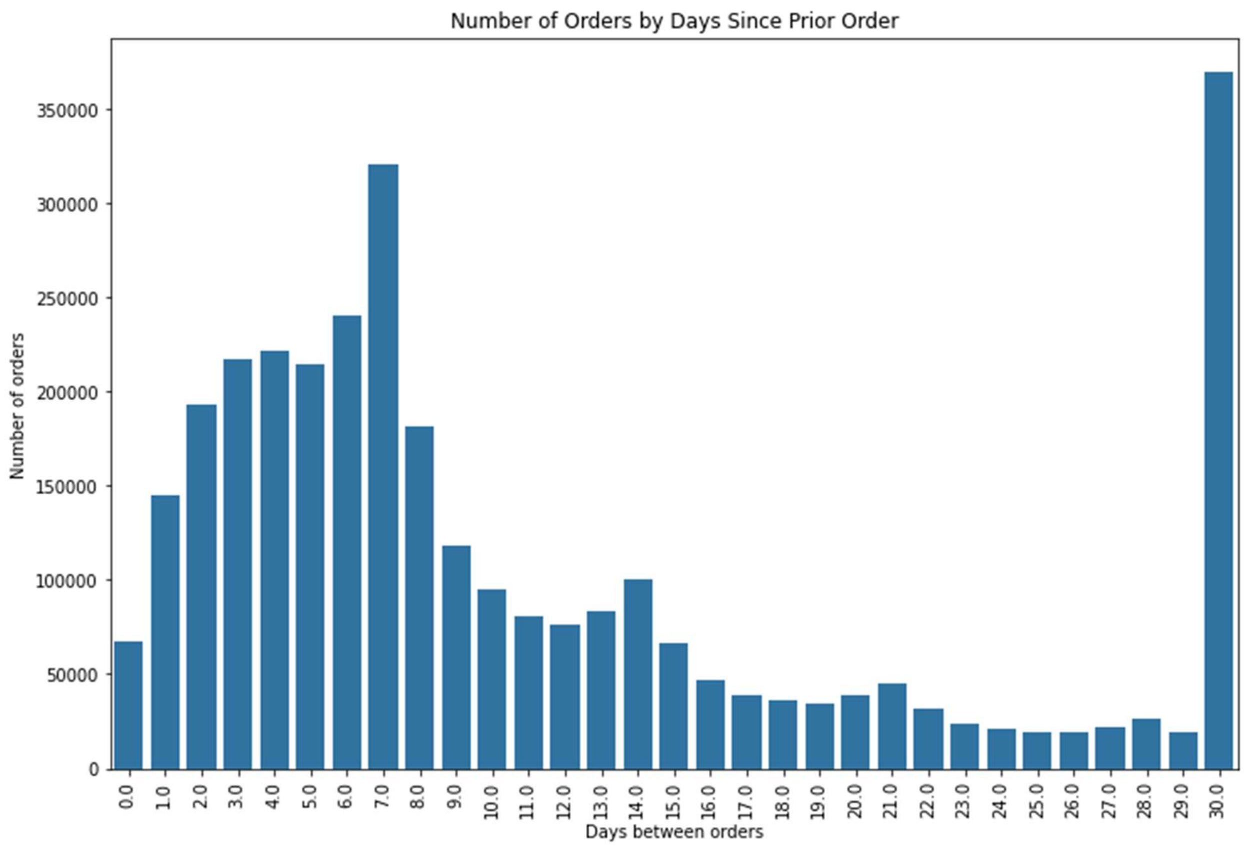

FIGURE 3. Number of products by days since prior order

By plotting the number of orders by days since prior order, we can see that lots of users order once in every week (local peak at 7 days). We can also see smaller local peaks at 14, 21 and 28 days. There seems to be a cut off value of 30 days for days since prior order.

In prior orders, there are about 59.0% of products are reordered, and about 87.9% of ordered containing reordered products. In train orders, there are about 59.9% of products are reordered, and about 93.4% of orders containing reordered products.

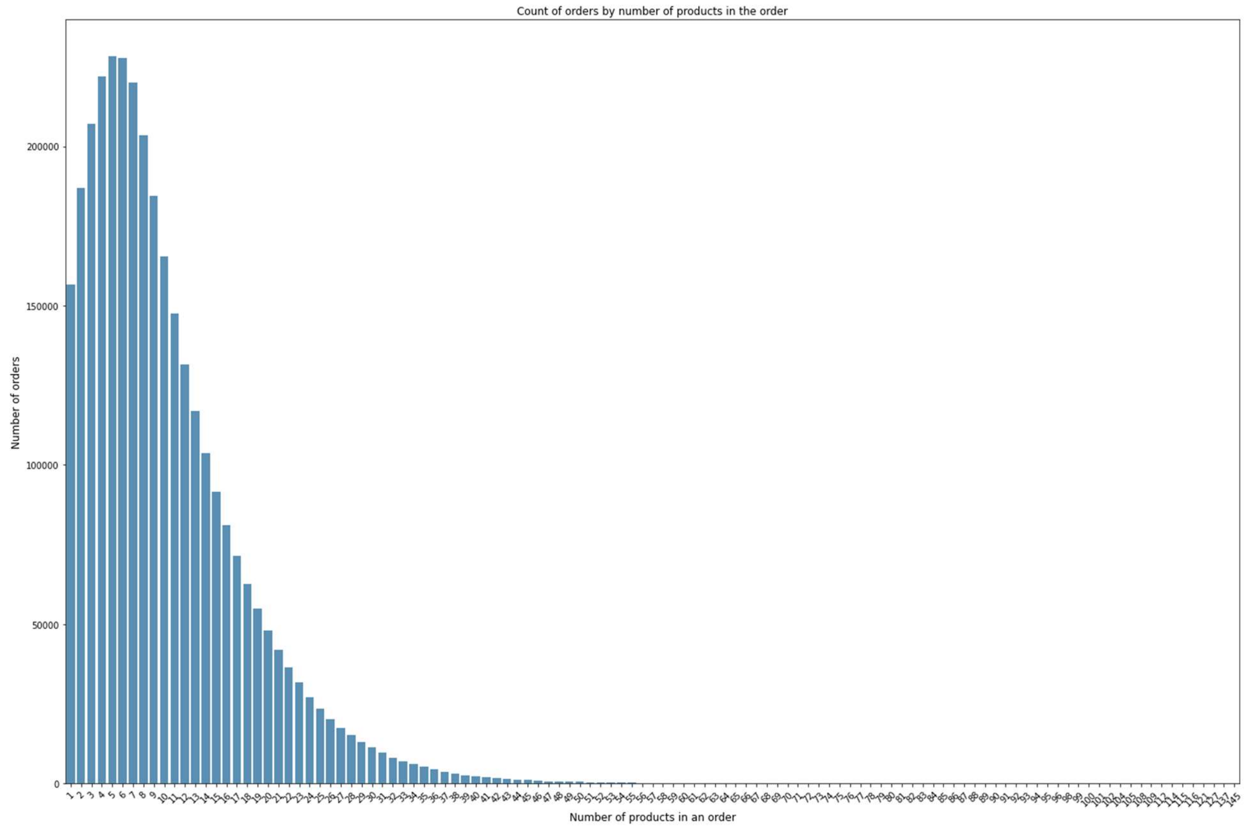

FIGURE 4. Number of orders by number of products in the order

The distribution of number of orders by number of products in the order is right-skewed, very few orders containing more than 50 products, the mode is 5, and the median is 8.

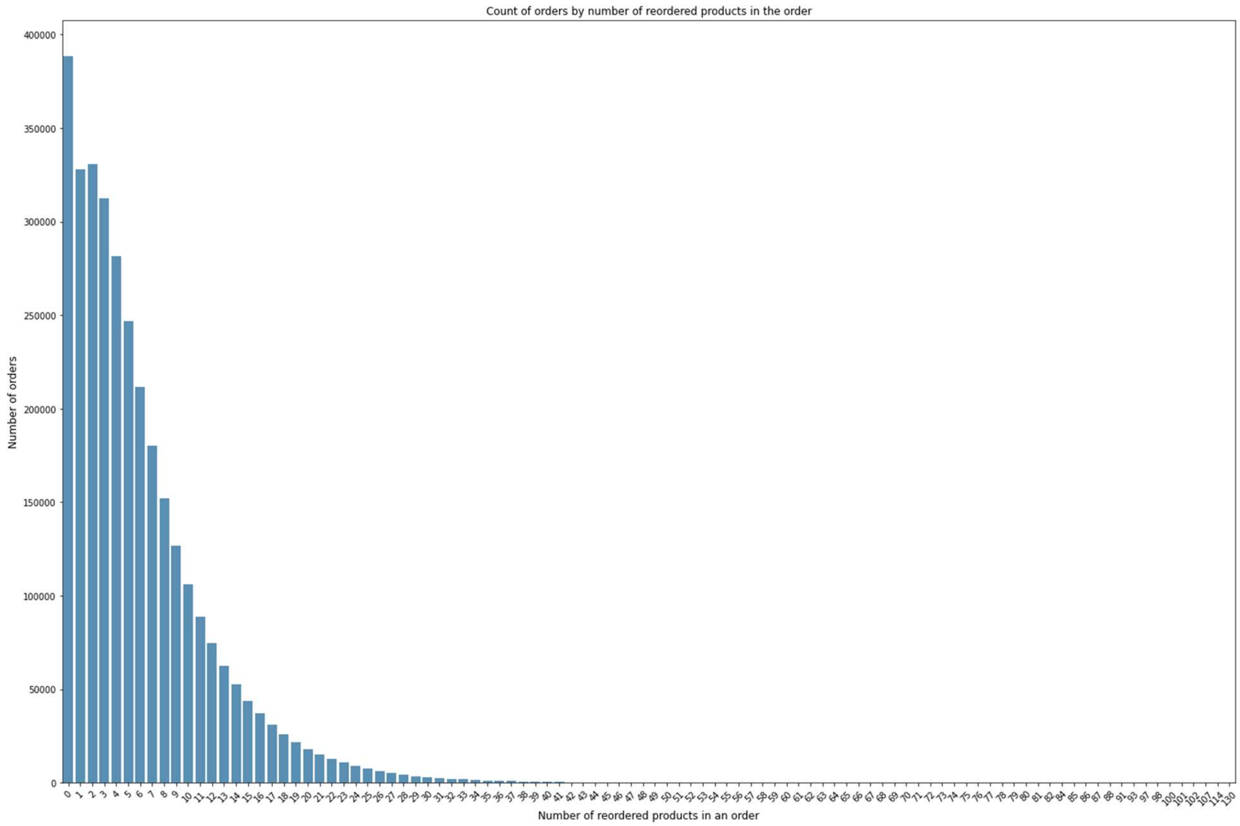

FIGURE 5. Number of orders by number of reordered products in the order

The distribution of number of orders by number of reordered products in the order is also right-skewed, the mode is 0 (no reordered product in the order), and the median is 4.

The orders and order_products_prior DataFrames were merged together in one DataFrame called prior_products, and One-way ANOVA was performed to test whether there is difference in mean number of reordered products in orders placed on different time or day. The p value for both tests is 0.0, this indicated that there is statistical significant difference in mean number of reordered products in orders placed on different time or day.

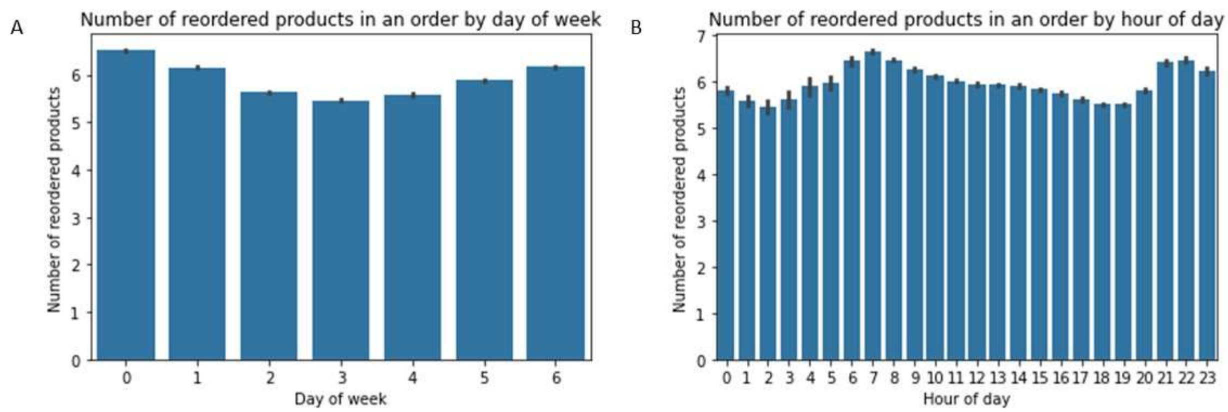

FIGURE 6. Bar plots present the difference in mean of number of reordered products in an order by day of week or hour of day.

3. Data Selection and Train Test Split

As I mentioned above, the orders (orders['eval_set'] == 'prior') and order_products_prior DataFrames were merged together into one DataFrame called "prior_products". This DataFrame contains all products information in prior orders. The orders (orders['eval_set'] == 'train') and order_products_train DataFrames were merged together into one DataFrame called "train_products". This DataFrame contains all products information in last orders (the orders need to be predicted).

The prior_products DataFrame has 32434489 rows, indicated there are 32434489 user_products pairs in the DataFrame. There will be over 30 features after feature engineering, if I use all the data for model training, it would be extremely slow. Therefore, for this project, I randomly selected 1/6 users for study.

Since I could not get access to the order_products__test.csv. I randomly split the users I selected in to training and testing set for this study.

The target of this study is for all the user_products in the users prior orders, whether or not they would get reordered in the last orders.

4. Feature Engineering

I generated a large number of features to describe the characteristics of the users, the products, the user product interactions and users'last orders.

The description of these features are shown in the following tables.

feature	description
U_num_of_orders	total number of orders each user has placed
U_num_of_products	total number of products each user has purchased
U_products_mean	average number of products each user has purchased per order
U_products_std	std of the above
U_unique_products	number of unique products each user has purchased
U_num_of_reordered_products	number of total products each user has purchased which are reordered
U_reordered_mean	average number of reordered products each user has purchas ed per order
U_reordered_std	std of the above
U_reordered_products_ratio	proportion of products each user has purchased which are reordered
U_order_dow_mean	mean of order_dow for each use
U_order_dow_std	std of order_dow for each use
U_order_hour_of_day_mean	mean order_hour_of_day for each user
U_order_hour_of_day_std	std order_hour_of_day for each user
U_days_between_orders_mean	mean of days_since_prior_order for each user
U_days_between_orders_std	std of days_since_prior_order for each user
U_num_of_orders_containg_reorder	number of the orders containing reordered items for each users
U_reordered_order_ratio	proportion of orders each user has placed which has reordered products

Table 12. User features

feature	description
P_num_of_orders	number of times the product has been purchased
P_num_of_users	number of users who have purchased the product
P_reorder_ratio	proportion between the times the product was reordered and the total time the product was ordered.
P_order_hour_of_day_mean	average order_hour_of_day of each product
P_order_hour_of_day_std	std of the above
P_order_dow_mean	average order_dow of each product
P_order_dow_std	std of the above
P_add_to_cart_order_mean	average add_to_cart_order of each product
P_add_to_cart_order_std	std of the above
P_days_between_orders_mean	average days_between_orders of each product
P_days_between_orders_std	std of the above
P_num_of_products_mean	average number of products in the same order
P_num_of_products_std	std of the above

Table 13. Product features

UxP_num_of_orders	number of times the user has purchased the product
UxP_order_ratio	proportion of orders the user purchased the product
UxP_orders_since_last	number of orders since last purchase of the product
UxP_orders_since_last_ratio	number of orders since last purchase of the product/u_num_of_orders
UxP_reordered	whether the product was reordered by the user before
UxP_order_dow_mean	average order_dow the user purchased the product
UxP_order_dow_std	std of the above
UxP_order_hour_of_day_mean	average order_hour_of_day the user purchased the product
UxP_order_hour_of_day_std	std of the above
UxP_add_to_cart_order_mean	average add_to_cart_order the user purchased the product
UxP_add_to_cart_order_std	std of the above

UxP_last_order	order number the user last purchased the product
UxP_last_order_ratio	UxP_last_order/U_num_of_orders

Table 14. User Product Interaction features

feature	**description**
LO_dow	dow of user's last order
LO_hour_of_day	hoy of user's ultimate order
LO_days_since_prior_order	days since user's previous order

Table 15. Product features

The features were generated and merged for training and testing set individually to prevent data leaking.

Before training a model, I removed some features that are colinear with others, including 'u_num_of_products','u_num_of_reordered_products','u_reordered_order_ratio','UxP_order_ratio','UxP_orders_since_last_ratio','UxP_last_order' and 'UxP_last_order_ratio'. For features 'p_order_hour_of_day_mean', 'p_order_hour_of_day_std', 'p_order_dow_mean', 'p_order_dow_std', 'p_add_to_cart_order_mean', 'p_add_to_cart_order_std', 'p_days_between_orders_mean','p_days_between_orders_std', I think they are less important than similar ones in User Product interaction features, so I excluded them from the model training.

5. Machine Learning

Supervised classification algorithms were used for this project.

5.1. Classification metrics

For this project, only the total accuracy is not enough for evaluating an algorithm. So I employed the following classification metrics. For model tuning, I use F1-score to take into account both recall and precision.

	Diagnosis	
	positive	negative
Test outcome positive	TP (True positive)	FP (False positive)
Test outcome	FN	TN

negative	(False negative)	(True negative)

Table 16. Basic terminology and derivations

Term	Formula
Accuracy	(TP + TN)/(P+N)
Recall	TP/(TP+FN)
Precision	TP/(TP+FP)
F1-score	(2 x recall x precision) / (recall+precision)

Table 17. Basic terminology and formula

Confusion Matrix were used to present the TP, FP, FN and TN of the prediction. Classification Report were used to show the precision, recall, F1-score and support for each class.

5.2 Random Forest

After tuning the model, the random forest classifier still suffered with low Recall, precision and F1 scores for test set.

accuracy_score = 0.91

recall_score = 0.16

f1_score = 0.25

precision_score = 0.63

Based on ROC curve (Figure. 5), I manually set some thresholds for prediction from 0.2 to 0.5.

And it seems threshold of 0.25 got the best result of recall (0.46) and f1 (0.44), without sacrificing too much on precision (0.41) and accuracy (0.88).

Figure 9 presents the feature importance for random forest model. It seems user product interaction features and user features are more important than product features in random forest model.

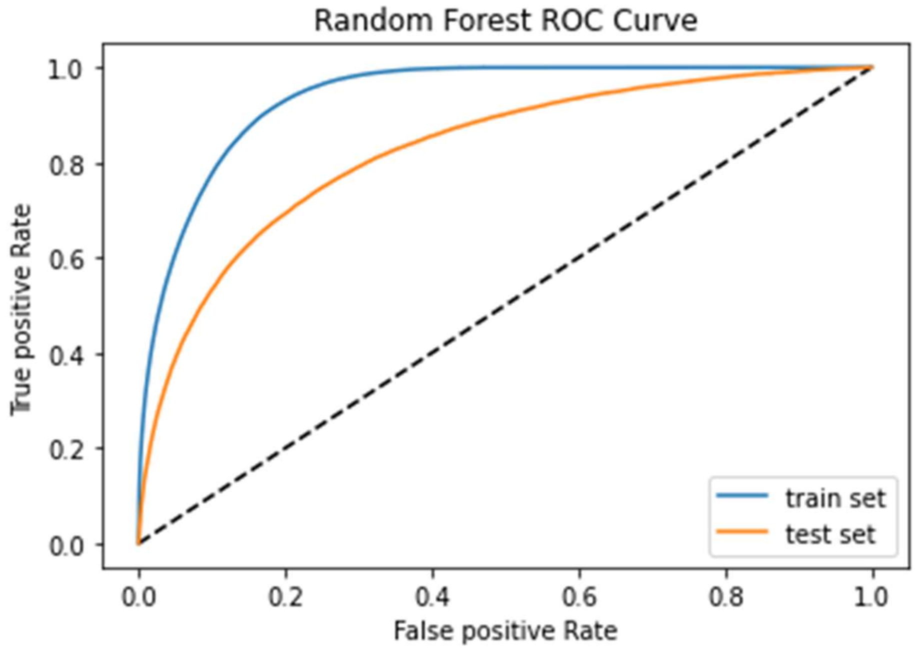

FIGURE 7. Random Forest ROC Curve

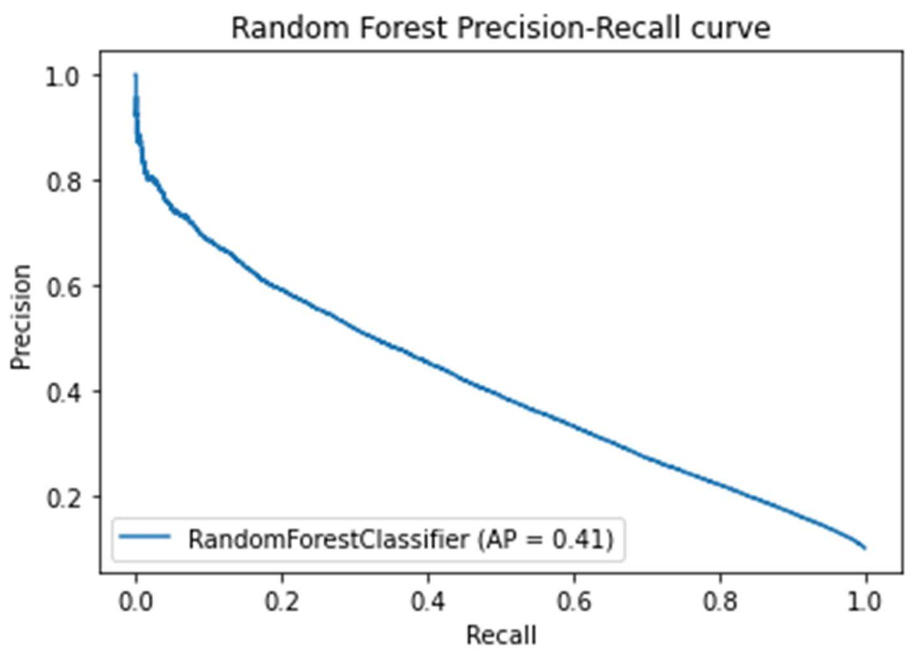

FIGURE 8. Random Forest Precision-Recall Curve

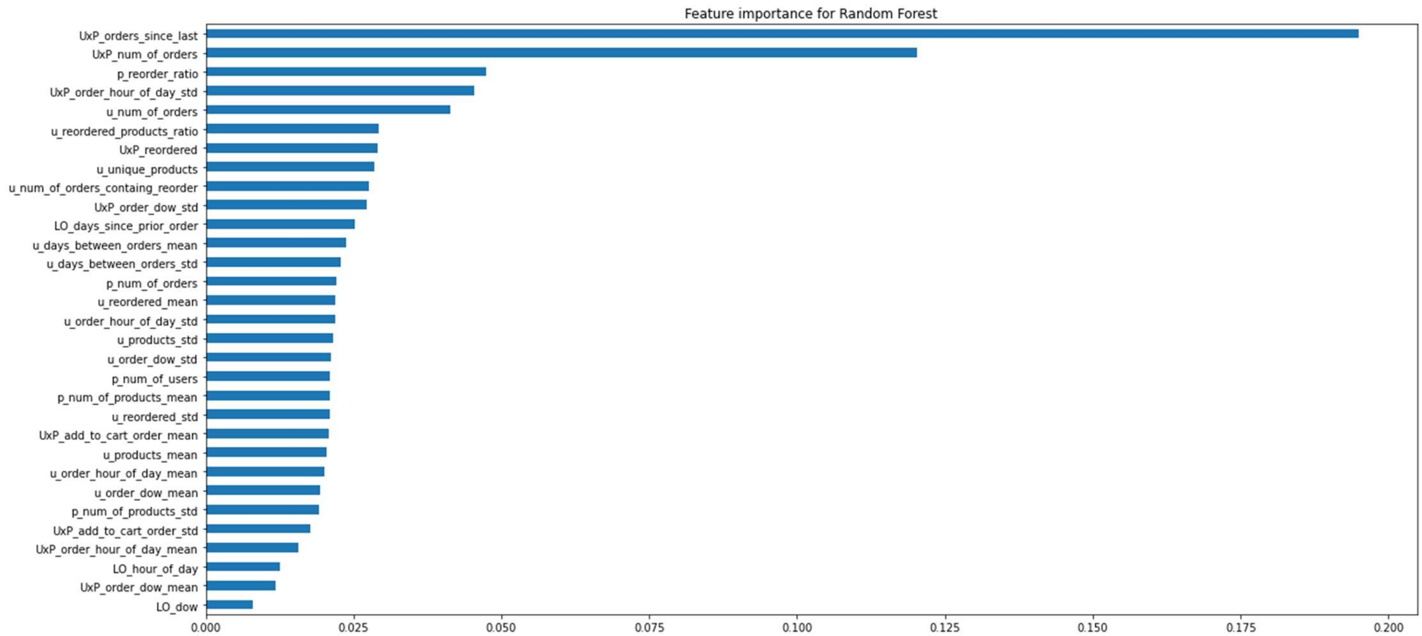

FIGURE 9. Random Forest Feature importance

5.3 XGboosting

There is a parameter in XGboosting classifier to balance positive and negative weights: the scale_pos_weight. I set it to 3 for training a model. And the results for test set are as follow:

accuracy_score= 0.890

recall_score= 0.421

f1_score 0.430

precision_score= 0.439

After tuning the model, I were able to get better scores:

accuracy_score= 0.890

recall_score= 0.437

f1_score 0.436

precision_score= 0.436

Figure 12 presents the feature importance for XGboosting model. It seems user product interaction features are less important in this model, which is quite different from random forest.

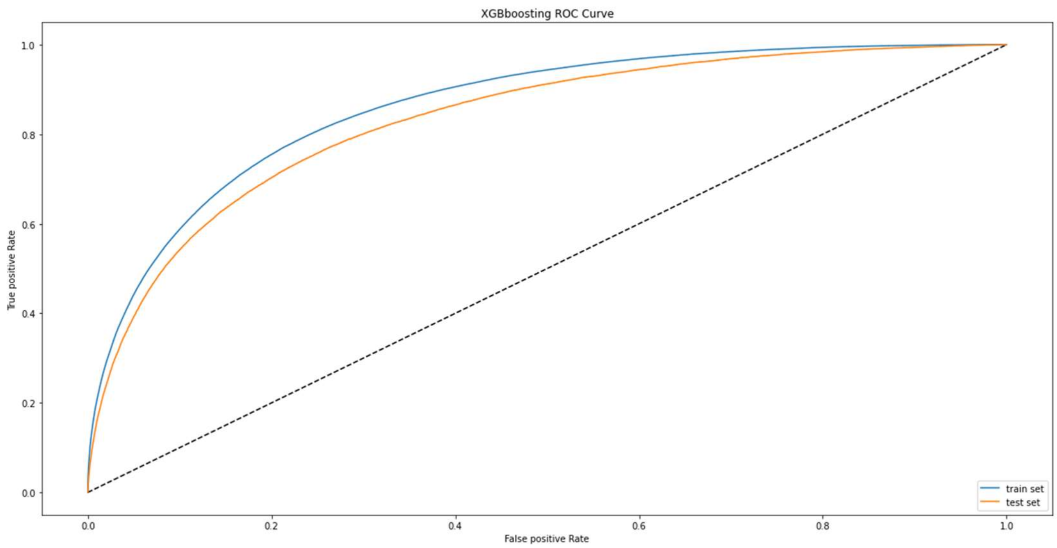

FIGURE 10. XGboosting ROC Curve

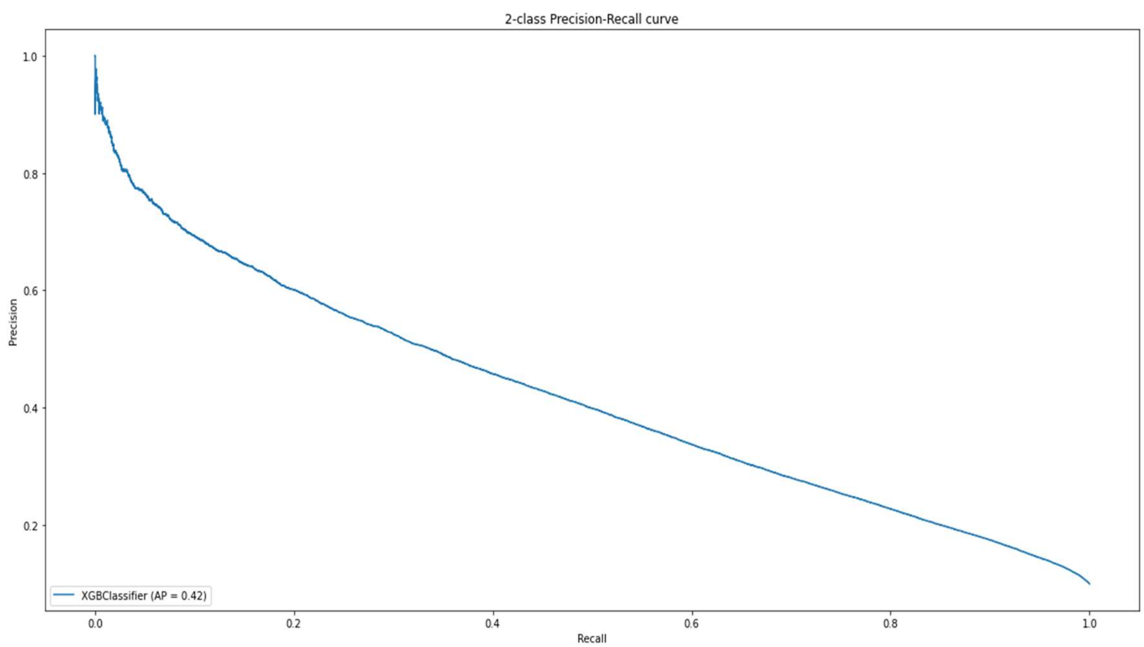

FIGURE 11. XGboosting Precision-recall Curve

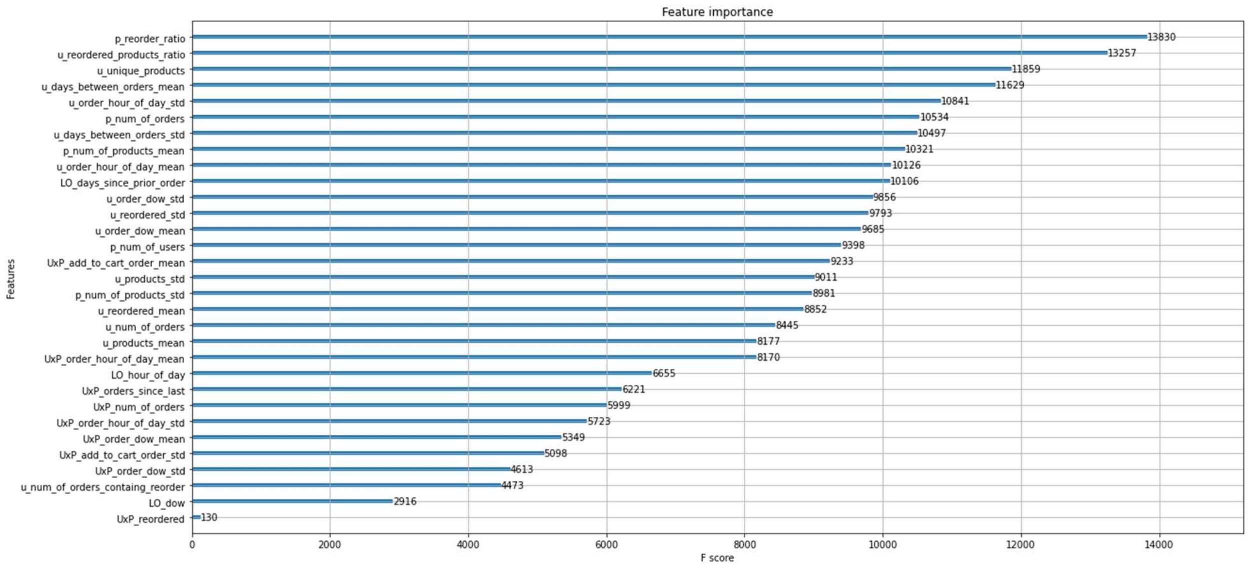

FIGURE 12. XGboosting Feature Importance

6. Summary

In this project, I product reorder prediction using "The Instacart Online Grocery Shopping Dataset 2017". Different machine learning algorithms has been used, including random forest and XGboosting. After tuning the model and resetting prediction threshold, both classifiers were able to get reasonable precision, recall and f1 score.

7. Ongoing Works

Some features need to be modified, UxP_days_since_last seems to be a better feature than UxP_orders_since_last. More features can be created, such as using the aisles and department information. I will try to model after feature selection. Training a XGboosting model with scale_pos_weight = 1, and manully set the prediction threshold, see if similar result will get with higher scale_pos_weight.

8. Acknowledgement

The author would like to thank Springboard and especially her mentors for the advice and support.