2018

Instacart Analysis

MSBA 326. Predictive Analytics and Machine Learning

■ **MAOYI SONG**

Executive summary

The instacart project will be combine two part, which are data exploration analysis and model generation by using feature engineering technology. The objective of this project is to get insights based on customer orders over time to predict which previously purchased products will be in a user's next order.

Introduction

Instacart, which launched in 2012, provides customers a personal shopper that scours grocery store aisles, purchases items and delivers them. In other word, it is a delivery company that pairs customers with personal shoppers who deliver food to their homes. In 2016, it signed a five-year contract with Whole Foods to become its exclusive delivery partner. And Instacart now goes head-to-head with Amazon's own Whole Foods delivery service through Amazon Prime Now (O'Brien, 2018). There are 300 chains include pet and drug stores who are partner with Instacart. The company has continued to build out other partnerships. It now has deals with more than 300 nationals, regional and local retailers, including Kroger, Sam's Club, Publix, Albertsons and Walmart Canada, among others. Until now, Instacart now services 15,000 grocery stores across 4,000 cities with 50,000 shoppers (Hanbury, 2018).

In the 2nd winner's interview, the workflow that Onodera build is very clear. The main approach is that he used XGBoost to create two gradient boosted tree models. One is combine products, users and reordered, which predict which products that customer will buy again. Another is only users and reordered that cover the problem of whether customer will buy previously purchased products. First one is prediction model and second is none prediction model. He average the results as probabilities and maximize the F1 score. He explore the data through several graphs in order to gain some insights about how hot are items and how often are they ordered. Then, created four types of features based on different perspective, which are user features (what is this user like), item features (what is this item like), user and item features (how does this user feel about this item) and date time features (what is this day and hour like). For example, under the user features, there are time between orders and time of day the user visits. He choose the top 30 features to build the model. After that, he randomly choose a users' purchase history in order to figure out purchase pattern and understand when and why a user does not repurchase an item. And for evaluation metric, he used F1 score which need to capture both precision and recall in a single metric and then find the properly threshold for both models.

For this project, I will focus on two parts. The first part is data exploration that easily gain some information about data by creating graphs. What products that people usually buy through the website? And which department/ aisle? When people will visit the website? How often? Also, by calculating ratio, we can compare the pervious purchase with reordered purchase. The second part is model generation. I use first 5 features to build the model and choose the best classifier by comparing the score of accuracy, precision and recall.

Columns and Data description

The dataset contains a sample of over 3 million orders history from more than 200,000 users. For each user's id, the detailed information has been stored to different labels. The data has been provided in 6 files.

orders (3.4m rows, 206k users):

- order_id: order identifier
- user_id: customer identifier
- eval_set: which evaluation set this order belongs in
- order_number: the order sequence number for this user (1 = first, n = nth)
- order_dow: the day of the week the order was placed on
- order_hour_of_day: the hour of the day the order was placed on
- days_since_prior: days since the last order

products (50k rows):

- product_id: product identifier
- product_name: name of the product
- aisle_id: foreign key
- department_id: foreign key

aisles (134 rows):

- aisle_id: aisle identifier
- aisle: the name of the aisle

departments (21 rows):

- department_id: department identifier
- department: the name of the department

order_products__SET (30m+ rows):

- order_id: foreign key
- product_id: foreign key
- add_to_cart_order: order in which each product was added to cart
- reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

eval_set in orders:

- "prior": orders prior to that users most recent order (~3.2m orders)
- "train": training data supplied to participants (~131k orders)
- "test": test data reserved for machine learning competitions (~75k orders)

Data Exploratory Analysis

Pre-process data:

I generate the different table together by using the method of merge. Below, it shows a table which contains all other information that prior dataset has for each order.

	order_id	product_id	add_to_cart_order	reordered	product_name	aisle_id	department_id	aisle	department	user_id	e
0	2	33120	1	1	Organic Egg Whites	86	16	eggs	dairy eggs	202279	
1	2	28985	2	1	Michigan Organic Kale	83	4	fresh vegetables	produce	202279	
2	2	9327	3	0	Garlic Powder	104	13	spices seasonings	pantry	202279	
3	2	45918	4	1	Coconut Butter	19	13	oils vinegars	pantry	202279	
4	2	30035	5	0	Natural Sweetener	17	13	baking ingredients	pantry	202279	

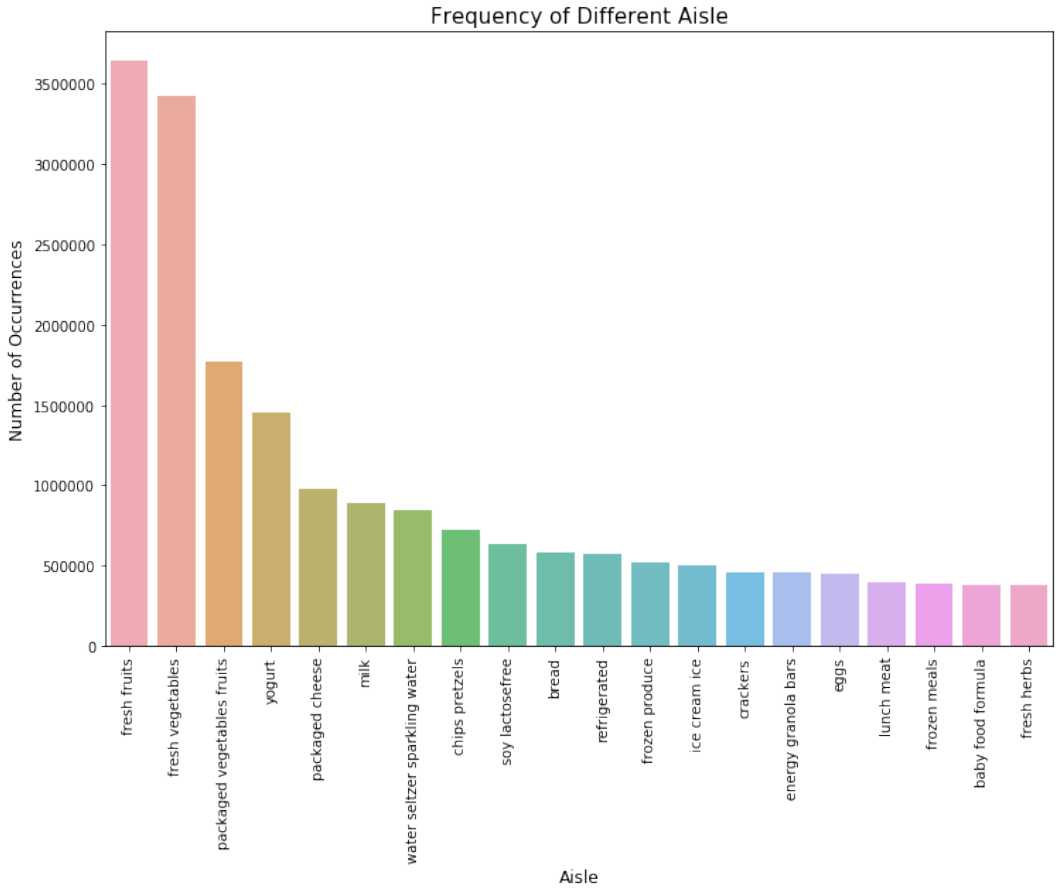
In this table we got 32434489 records and total 15 columns. And then, check the null value that the table contains.

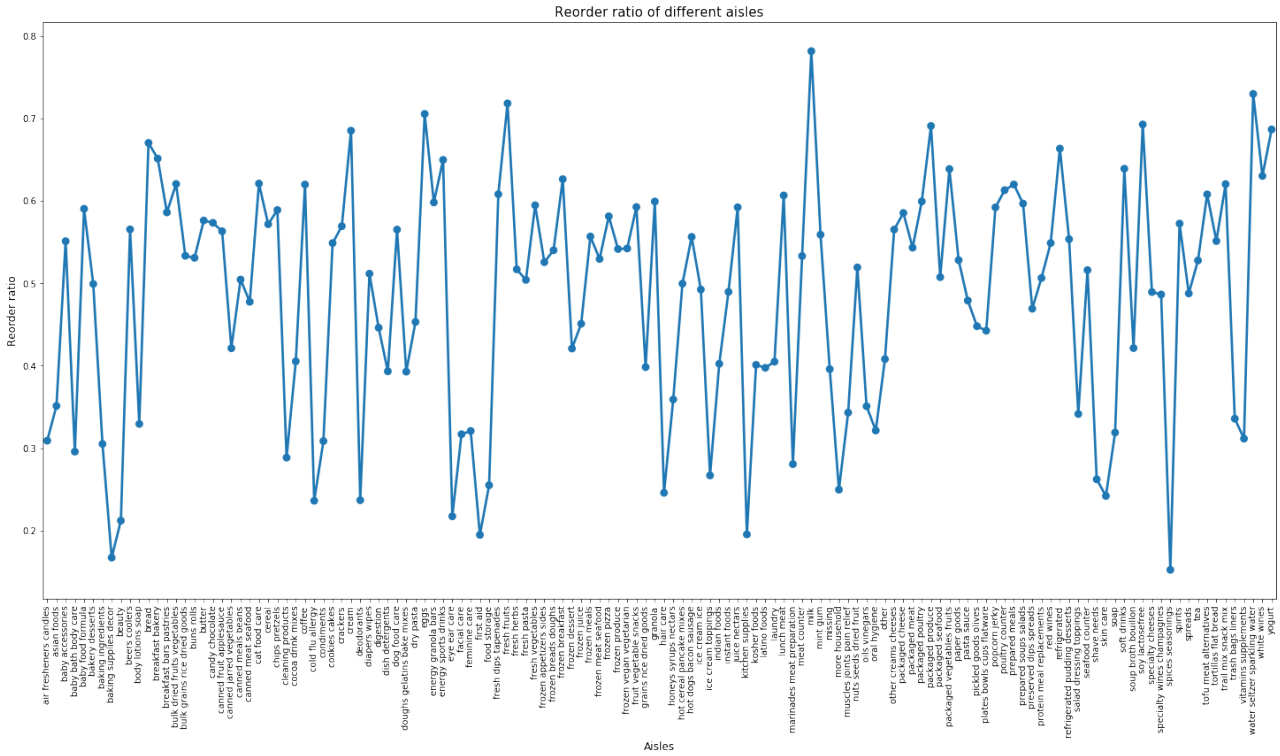
```
order_id                      0
product_id                    0
add_to_cart_order             0
reordered                     0
product_name                  0
aisle_id                      0
department_id                 0
aisle                         0
department                    0
user_id                       0
eval_set                      0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    2078068
dtype: int64
```

In the column of days since prior order, there are 2078068 records are NA. Because the column of "days since prior order" can be described as the time between orders, we decide to fill out all the NA with a number 0, which represent the user did not placed any orders recently.
Next, let's look at each column to gain the business insights based on the different categories.

	product_name	frequency_count
0	Banana	472565
1	Bag of Organic Bananas	379450
2	Organic Strawberries	264683
3	Organic Baby Spinach	241921
4	Organic Hass Avocado	213584
5	Organic Avocado	176815
6	Large Lemon	152657
7	Strawberries	142951
8	Limes	140627
9	Organic Whole Milk	137905
10	Organic Raspberries	137057

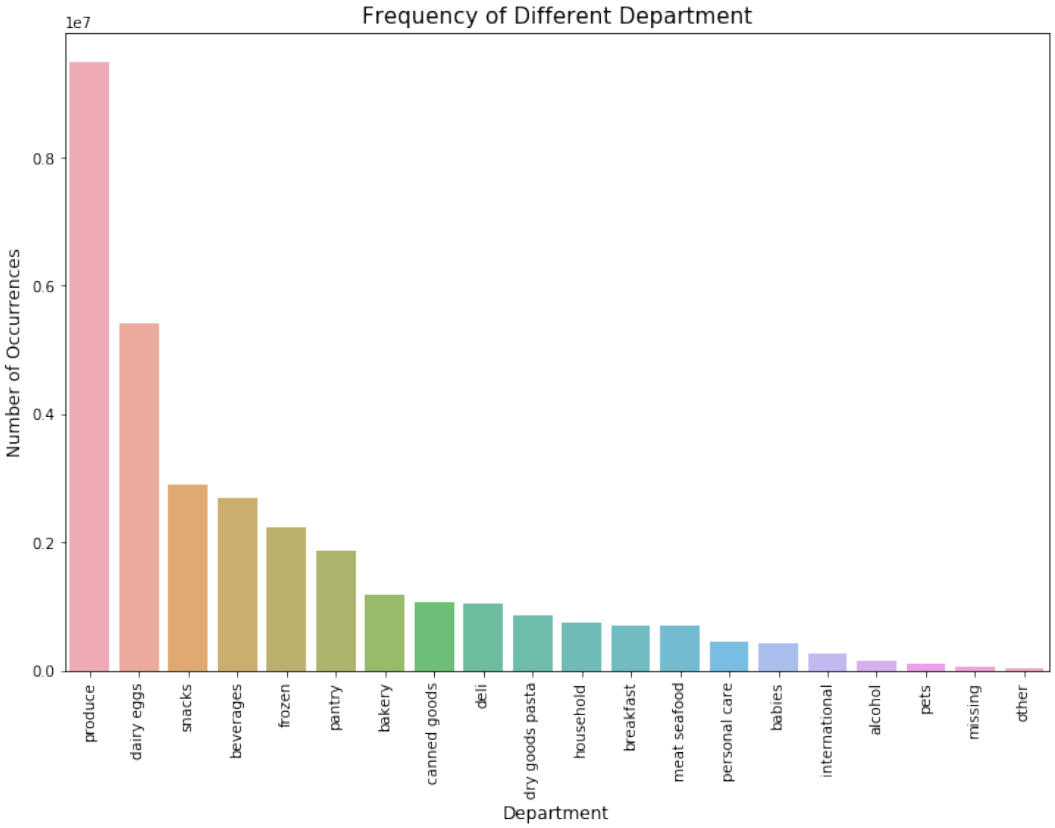
Based on the information that table give to us, the majority of the food are organic products. And most of them are fruits.

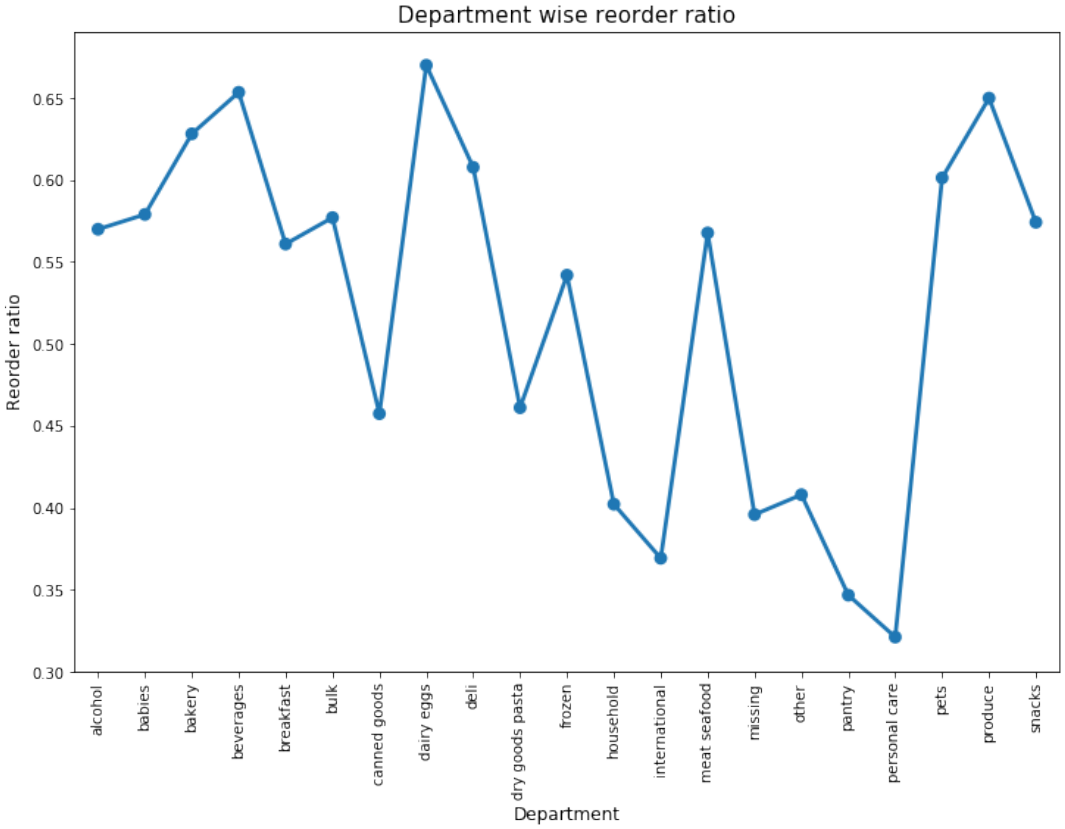




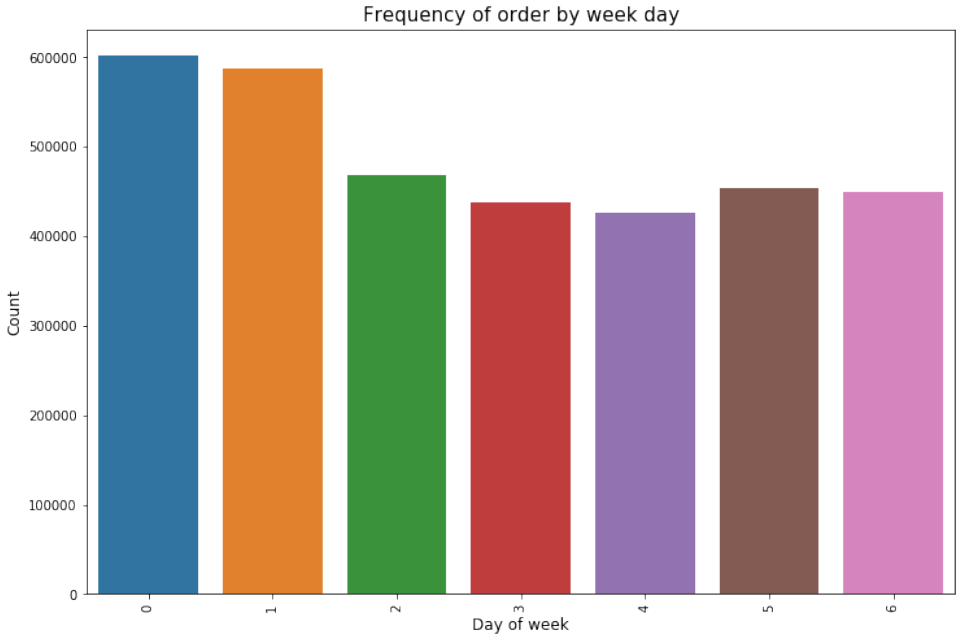
Look at the frequency of each aisle, we can see there are a big percentage that people more likely to buy fresh fruits and vegetables.

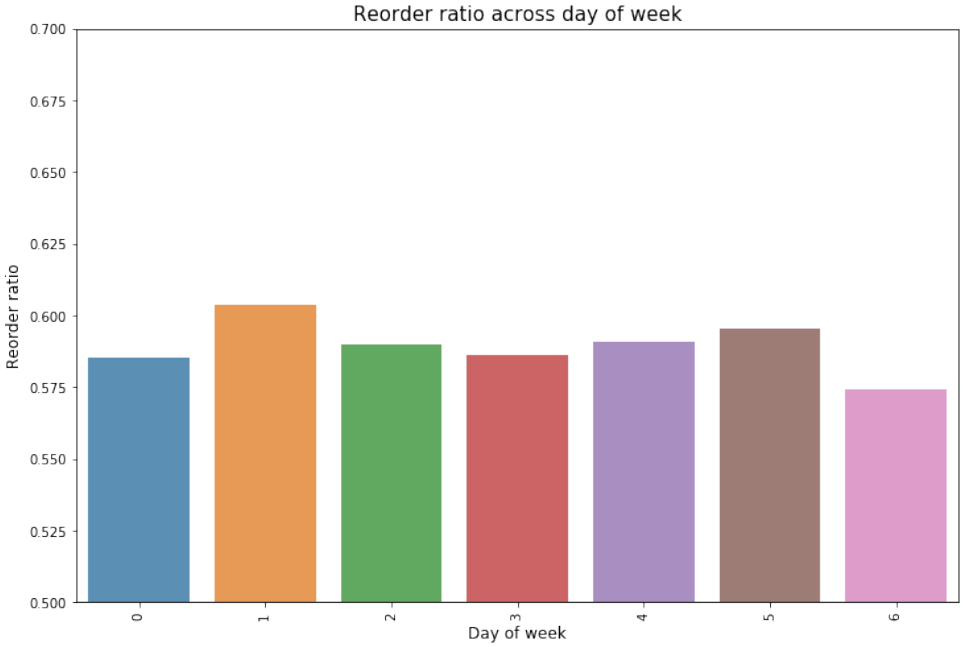
Also, when we related the different aisles with reorder ratio, we can see that most of people reorder the milk, and followed by fresh fruits, sparkling water and eggs.


Frequency of Different Department
1e7
0.8
0.6
0.4
0.2
0.0
Number of Occurrences
produce
dairy eggs
snacks
beverages
frozen
pantry
bakery
canned goods
deli
dry goods pasta
household
breakfast
meat seafood
personal care
babies
international
alcohol
pets
missing
other
Department

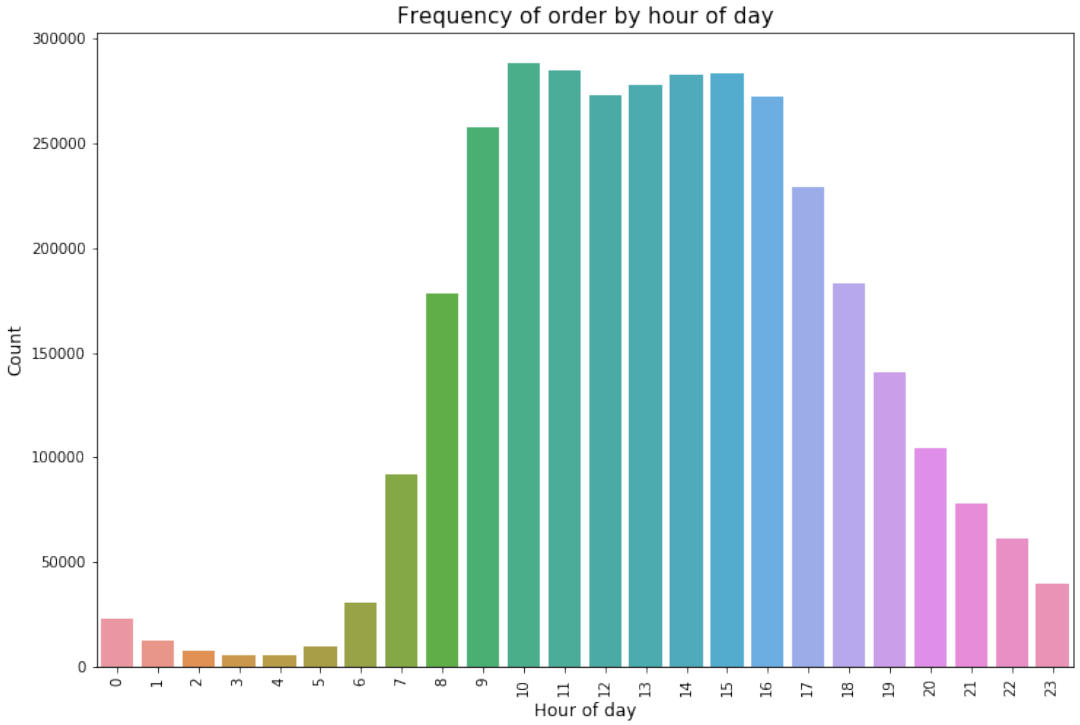

Department wise reorder ratio
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Reorder ratio
alcohol
babies
bakery
beverages
breakfast
bulk
canned goods
dairy eggs
deli
dry goods pasta
frozen
household
international
meat seafood
missing
other
pantry
personal care
pets
produce
snacks
Department

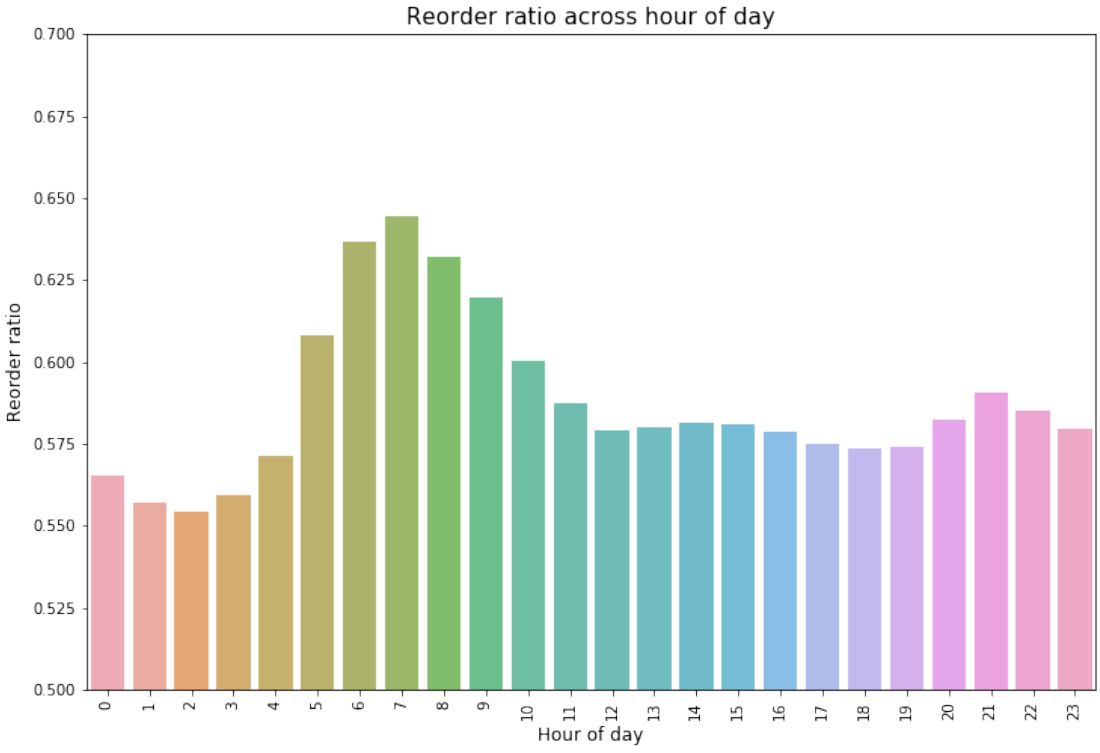
Also, when we look at different department, the same result comes out. Most of customer buy the produces and egg and reordered them again, however, we can see the ratio that customer reordered the beverages higher produces.
After we get extra information about the department and aisles, let's move to the order detail to gain information about customers' purchase behavior across the timeline.



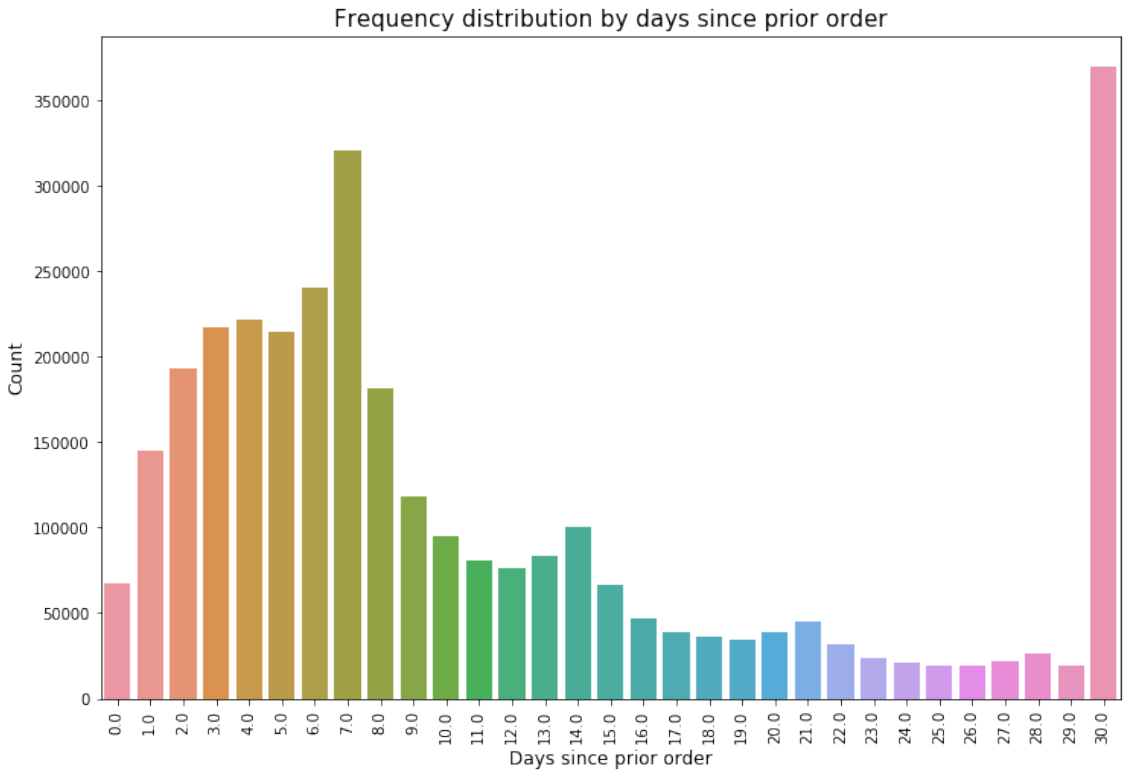


0=Saturday, 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday
Most of customers usually placed an order during the weekends and reordered on Sunday.

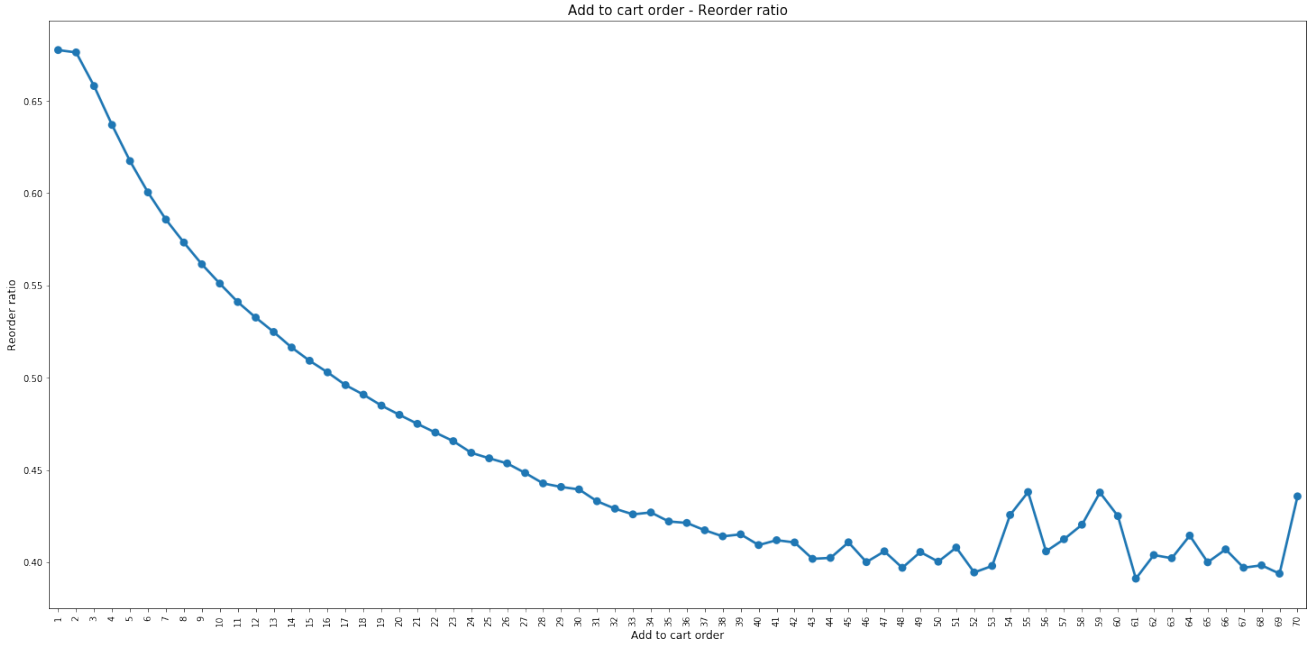




Most of customers usually placed an order between the morning 8am and afternoon 6pm, and reordered in the morning (6am to 9am).



The time between the orders, some of them are 0, which means customers did not come back to place an order again. However, we can see most of customers reordered a week after—7 days.



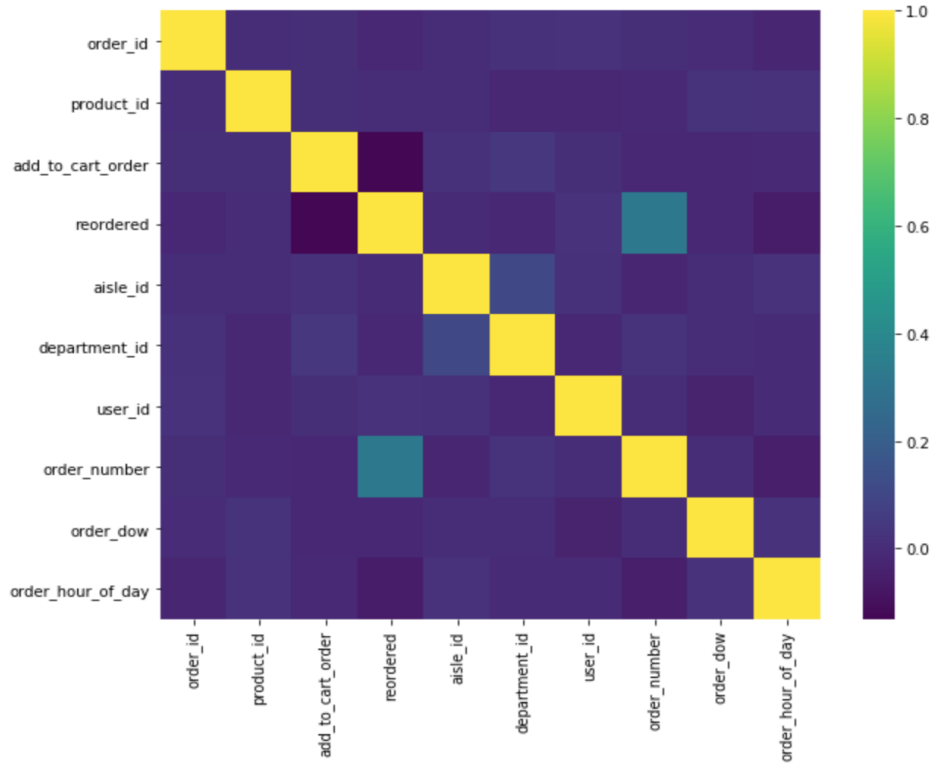
Based on the add to cart order ratio, we can see customer are more likely to reorder the products we used to buy compare to the new products.

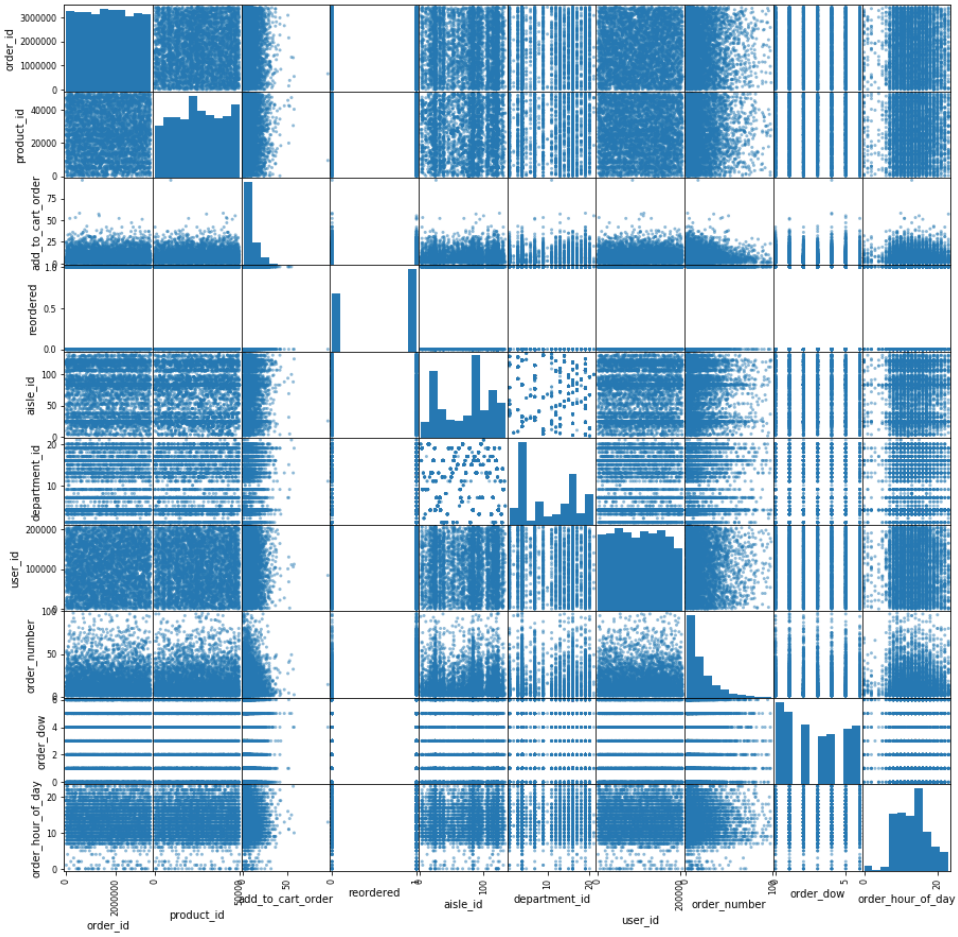
Model Generation

Because the data that labeled as train may not contain products that previously ordered by that user, I only use the data which labeled as prior to build the model. And, as we can see, the prior dataset has huge records (memory usage: 3.9+ GB), so I decide to randomly choose 5000 records.

	order_id	product_id	add_to_cart_order	reordered	product_name	aisle_id	department_id	aisle	department	us
3846192	406039	24954	1	1	Total 0% with Honey Nonfat Greek Strained Yogurt	120	16	yogurt	dairy eggs	9
16009955	1689173	49521	7	1	Lentil Filled Bolani 2 Pack	13	20	prepared meals	deli	8
21090475	2224589	16283	9	1	Fresh Asparagus	83	4	fresh vegetables	produce	1
28697339	3026497	5883	1	1	Organic Strawberry Lemonade	98	7	juice nectars	beverages	10
28043420	2957217	18523	21	1	Total 2% All Natural Greek Strained Yogurt wit...	120	16	yogurt	dairy eggs	20

And then, I drop the categorical column, such as product_name, aisle, department , and eval_set. Next, plot some graphs to see the correlation between these features.





Through this graph, we barely can see the relationship between these features. The only two features has the correlation in the heat map are reorder and order number, but less than 0.4.
Next, in order to select effective features to build the model, performing feature selection by using the method of SelectKBest and RFE.

SelectKBest:

```
Ranking of features is
1 . Score   order_number  is  569.5038002711212
2 . Score   add_to_cart_order  is  88.72887191132432
3 . Score   order_hour_of_day  is  16.501063103934953
4 . Score   user_id  is  2.104847970313727
5 . Score   order_dow  is  1.2615704383096802
6 . Score   order_id  is  1.0580328401207415
7 . Score   days_since_prior_order  is  0.9290843446853068
8 . Score   department_id  is  0.8292373948244601
9 . Score   aisle_id  is  0.07776832905367613
10 . Score   product_id  is  0.005947000083305333
['order_id' 'product_id' 'add_to_cart_order' 'aisle_id' 'department_id'
 'user_id' 'order_number' 'order_dow' 'order_hour_of_day'
 'days_since_prior_order']
```

RFE:

```
Num Features: 5
Selected Features: [False False  True False False False  True  True  True  True]
Feature Ranking: [6 4 1 3 2 5 1 1 1 1]
[(1, 'add_to_cart_order'), (1, 'days_since_prior_order'), (1, 'order_dow'), (1, 'order_hour_of_day'), (1, 'order_numb
er'), (2, 'department_id'), (3, 'aisle_id'), (4, 'product_id'), (5, 'user_id'), (6, 'order_id')]
```

By using the RFE, I keep reducing the number of features until a large drop in performance. It is an indication of an important feature that needs to be included in the model. And I got an optimum solution of 5 features.

Compare with RFE, the result of using SelectKBest method are similar. The important features are order number (score 569), followed by add to cart order and order hour of the day.

Therefore, I decide to use the first **5 features** to build the model which are **add to cart order, days since prior order, order dow (the day of the week the order was placed on), order hour of day and order number.**

I build a model list in order to choose the most effective model after calculate the accuracy score. And using cross validation method and set cv=10 to get the accuracy score.

```
models = []
models.append(('Kernel SVC',SVC(class_weight='balanced',kernel='rbf')))
models.append(('Decision Tree',DecisionTreeClassifier(class_weight='balanced',min_samples_split=5)))
models.append(('Logistic Regression',LogisticRegression(random_state = 0,max_iter=500,class_weight='balanced')))
models.append(('BOOSR',AdaBoostClassifier(DecisionTreeClassifier(), n_estimators=200,algorithm="SAMME.R", learning_rate
models.append(('RF2', RandomForestClassifier(n_estimators=120,max_depth=2, random_state=0)))
models.append(('SVM linear', LinearSVC(random_state=0,class_weight='balanced')))
models.append(('KNN',KNeighborsClassifier()))
```

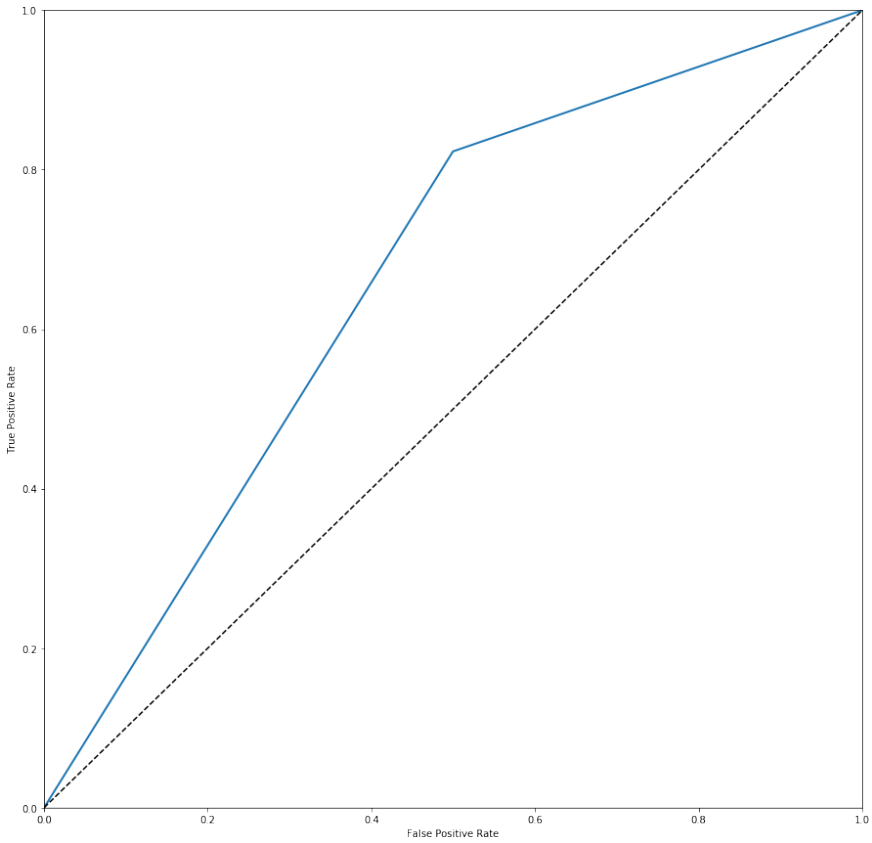
```
Kernel SVC: 0.671600 (0.018326)
Decision Tree: 0.617000 (0.021359)
Logistic Regression: 0.656600 (0.021634)
BOOSR: 0.633200 (0.024983)
RF2: 0.710000 (0.014913)
SVM linear: 0.640800 (0.066748)
KNN: 0.682600 (0.020240)
```

And the result shows us the random forest is the best classifier. And then, I set n_estimators=500 and test maximum depth of tree (max_depth) in different level.

```
RF4: 0.716800 (0.016618)
RF5: 0.716400 (0.017884)
RF6: 0.718800 (0.016690)
RF7: 0.717000 (0.016205)
RF8: 0.713600 (0.020742)
RF15: 0.701200 (0.026686)
RF20: 0.698800 (0.024710)
RF25: 0.699200 (0.026656)
RF30: 0.697600 (0.025750)
```

As we can see, by increasing the max depth, the result does not change that much. It keep around 0.7. And the best score we get is when we set max depth=6. And I got the precision score 0.708 and recall 0.881. Not bad, but I want to reach a balance between the score of precision and recall. So, I decide to see which max depth will get better precision and recall score.

Max_Depth	Accuracy	Precision	Recall
6	0.717200	0.708116232	0.8813438800000002
7	0.718400	0.7067940160000001	0.8698236030000001
8	0.714200	0.7072816879999999	0.8647099449999999
15	0.699600	0.70242016	0.7420421880000001
20	0.697200	0.704825221	0.8207419530000001
25	0.698200	0.703227621	0.8186106540000001
30	0.698600	0.704279866	0.819467174
35	0.698200	0.7041761870000001	0.8190398240000001



Based on the table, we got the best result when we set max depth equals 15, and the precision is around 0.702 and recall is around 0.742. And when set the max depth =15 and cv=10, we got roc auc score equals to 0.66. Therefore, the Random Forest finalized params that I got is: “RandomForestClassifier (n_estimators=500, max_depth=15, random_state=0)”.

Reference

262588213843476. (n.d.). The Instacart Online Grocery Shopping Dataset 2017 Data Descriptions. Retrieved from https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b

Hanbury, M. (2018, November 07). We compared Whole Foods' grocery delivery from Amazon Prime Now and Instacart, and it was clear which service does it better. Retrieved from https://www.businessinsider.com/whole-foods-groceries-amazon-prime-now-vs-instacart-2018-10

Instacart Market Basket Analysis, Winner's Interview: 2nd place, Kazuki Onodera. (2017, September 21). Retrieved from http://blog.kaggle.com/2017/09/21/instacart-market-basket-analysis-winners-interview-2nd-place-kazuki-onodera/

O'Brien, S. (2018, October 16). Instacart is now valued at $7 billion. Retrieved from https://www.cnn.com/2018/10/16/tech/instacart-funding/index.html