



# APPLYING DATA SCIENCE TO INSTACART'S PUBLIC DATASET

Project Report

**Group D, APDS-01 [2017-18]**

Anurag Pandey, Prashant Nair, Rajiv Ranjan,  
Shaveta Sachdeva, Sonia Grewal, Sunil K  
Gupta, Vikas Kumar Rana

Advanced Programme in Data Sciences, IIM Calcutta

## Table of Contents

<b>Acknowledgment</b>	<b>3</b>
<b>Contact us</b>	<b>3</b>
<b>Code repository</b>	<b>3</b>
<b>What is INSTACART?</b>	<b>4</b>
<b>Objective</b>	<b>4</b>
The Instacart Online Grocery Shopping Dataset 2017	5
Data Dictionary	5
<b>Our Approach</b>	<b>9</b>
<b>EXPLORATORY DATA ANALYSIS</b>	<b>11</b>
Total number of orders placed by users	13
Overall maximum order number	13
No. of Order by week day	14
No. of orders by hour of day	14
Frequency of Day of week vs Hour of day	15
Frequency Distribution by Days since Prior Order	15
Most Important Aisles	16
Active / Inactive Users	17
Frequency of Products ordered	18
Reorder Ratio	19
When do people order?	23
When do they order again?	24
How many prior orders are there?	24
How many items do people buy?	25
Bestsellers	25
How often do people order the same items again?	26
Most often reordered	26
Which item do people put into the cart first?	27
Association between time of last order and probability of reorder	27
Association between number of orders and probability of reordering	28
Organic vs Non-Organic Products	28
Organic vs Non-Organic Orders	29
Reordering Organic vs Non-Organic	29
Visualizing the Product Portfolio	30
How often are products from the department/aisle sold?	31
<b>MARKET BASKET ANALYSIS</b>	<b>32</b>
<b>Introduction</b>	<b>32</b>
<b>Methodology</b>	<b>32</b>
<b>CONCLUSIONS</b>	<b>41</b>
<b>DATA MODELLING</b>	<b>42</b>
<b>Choice of Algorithm</b>	<b>42</b>
<b>What is Light GBM?</b>	<b>42</b>

How it differs from another tree based algorithm?	42
Why Light GBM is gaining extreme popularity?	42
What about its implementation?	42
Control Parameters	43
<b>Tuning Parameters</b>	<b>44</b>
For faster speed	44
For better accuracy	44
To deal with over-fitting	44
<b>Feature Extraction</b>	<b>45</b>
<b>Model Validation (Mean F1 Score)</b>	<b>46</b>
<b>RESULTS</b>	<b>47</b>

## ACKNOWLEDGMENT

It is our privilege to express our sincerest regards to Program directors and our teachers, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project. This has been a wonderful year in terms of learning and we are grateful towards IIM Calcutta for providing such an opportunity. This course has helped us to explore the world of analytics and its real-world application.

We also take this opportunity to thank all our professors who have directly or indirectly helped in our project. Last but not the least we express our thanks to our classmates for their cooperation and support.

## CONTACT US

- Anurag Pandey, [anuragpandey2489@gmail.com](mailto:anuragpandey2489@gmail.com)
- Prashant Nair, [prash.nair26@gmail.com](mailto:prash.nair26@gmail.com)
- Rajiv Ranjan, [rajiv.ranjan.liveindia@gmail.com](mailto:rajiv.ranjan.liveindia@gmail.com)
- Shaveta Sachdeva, [sachdeva.shaveta@gmail.com](mailto:sachdeva.shaveta@gmail.com)
- Sonia Grewal, [soniagrewal17@gmail.com](mailto:soniagrewal17@gmail.com)
- Sunil K Gupta, [skg467@gmail.com](mailto:skg467@gmail.com)
- Vikas Kumar Rana, [vkumarcsc95@gmail.com](mailto:vkumarcsc95@gmail.com)

## CODE REPOSITORY

<https://github.com/nairPrashant/InstaDataScience/>

## **WHAT IS INSTACART?**

Instacart is an American company that operates as a same-day grocery delivery service. Customers select groceries through a web application from various retailers and delivered by a personal shopper. Up until the end of 2017, Instacart only has operations and services in the United States.

**How they Service:** Instacart's service is mainly provided through a smartphone app, available on iOS and Android platforms, apart from its website. Customers can pay with Android Pay and Apple Pay on their respective platforms. Initially Instacart shoppers simply went to a store and purchased the ordered items at retail and, in addition to the delivery charge, added a mark-up of 10 to 20 percent. As the business has developed, the firm has established relationships with grocery firms which share their (store) existing mark-up, allowing Instacart users to shop at in-store prices

Instacart's data science team plays a big part in providing this delightful shopping experience. Currently they use transactional data to develop models that predict which products a user will buy again, try for the first time, or add to their cart next during a session. Recently, Instacart open sourced this data - see their blog post on 3 Million Instacart Orders, Open Sourced. "The Instacart Online Grocery Shopping Dataset 2017". This anonymized dataset contains a sample of over 3 million grocery orders from more than 200,000 Instacart users.

## **OBJECTIVE**

- To predict which products will be in a user's next order
- To predict which Users will try it for the first time.
- To predict which users will add products to cart next during a session.

The goal of this project is to predict, for everyone, which products that they have purchased before they will purchase again in their final order. This is a challenging multiclass multi label prediction problem where we are asked to predict a basket of about 50,000 potential items for being reordered.

Predictions are submitted as an order ID and a list of product IDs predicted for that order. Predictions can include none, a special product ID indicating that no previously ordered products have been ordered. None can be combined with other product IDs in prediction, but will of course only occur on its own in the ground truth.

We derived relevant features and experimented with different modelling approaches. Predictions are evaluated using the average F1-score across orders.

## The Instacart Online Grocery Shopping Dataset 2017

The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. The only information provided about users is their sequence of orders and the products in those orders. All the IDs in the dataset are entirely randomized, and cannot be linked back to any other ID. Only products that are bought by multiple people at multiple retailers are included, and no retailer ID is provided

### Data Dictionary

- **orders.csv**

orders (3.4m rows, 206k users)	
order_id	order identifier
user_id	customer identifier
eval_set	which evaluation set this order belongs in (see SET described below)
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, capped at 30 (with NAs for order_number = 1)

This file gives a list of all orders we have in the dataset. 1 row per order. For example, we can see that user 1 has 11 orders, 1 of which is in the train set, and 10 of which are prior orders. The orders.csv doesn't tell us about which products were ordered. This is contained in the order\_products.csv

```
Observations: 3,421,083
Variables: 7
$ order_id      <int> 2539329, 2398795, 473747, 2254736, 431534, 3367565, 5...
$ user_id      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,...
$ eval_set     <chr> "prior", "prior", "prior", "prior", "prior", "prior",...
$ order_number <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1, 2, 3, 4, 5, 6, ...
$ order_dow    <int> 2, 3, 3, 4, 4, 2, 1, 1, 1, 4, 4, 2, 5, 1, 2, 3, 2, 2,...
$ order_hour_of_day <int> 8, 7, 12, 7, 15, 7, 9, 14, 16, 8, 8, 11, 10, 10, 10, ...
$ days_since_prior_order <dbl> NA, 15, 21, 29, 28, 19, 20, 14, 0, 30, 14, NA, 10, 3,...
```

- **products.csv**

products (50k rows)	
product_id	product identifier
product_name	name of the product
aisle_id	foreign key
department_id	foreign key

This file contains the names of the products with their corresponding product\_id. Furthermore, the aisle and department are included.

```
Observations: 49,688
Variables: 4
$ product_id    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
$ product_name  <chr> "Chocolate Sandwich Cookies", "All-Seasons Salt", "Robust Gold...
$ aisle_id      <int> 61, 104, 94, 38, 5, 11, 98, 116, 120, 115, 31, 119, 11, 74, 56...
$ department_id <int> 19, 13, 7, 1, 13, 11, 7, 1, 16, 7, 7, 1, 11, 17, 18, 19, 12, 1...
```

- **order\_products\_SET**

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

where SET is one of the four following evaluation sets (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)

This file gives us information about which products (product\_id) were ordered. It also contains information of the order (add\_to\_cart\_order) in which the products were put into the cart and information of whether this product is a re-order (1) or not (0).

For example, we see below that order\_id 1 had 8 products, 4 of which are reorders.

Still we don't know what these products are. This information is in the products.csv

```
Observations: 32,434,489
Variables: 4
$ order_id      <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4...
$ product_id    <int> 33120, 28985, 9327, 45918, 30035, 17794, 40141, 1819, 4366...
$ add_to_cart_order <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3...
$ reordered     <int> 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1...
```

- **aisles.csv**

aisles (134 rows)	
aisle_id	aisle identifier
aisle	the name of the aisle

This file contains the different aisles.

aisle_id	aisle
1	prepared soups salads
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat preparation
6	other
7	packaged meat
8	bakery desserts
9	pasta sauce
10	kitchen supplies

```
Observations: 134
Variables: 2
$ aisle_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, ...
$ aisle      <chr> "prepared soups salads", "specialty cheeses", "energy granola bars"...
```

- **Departments.csv**

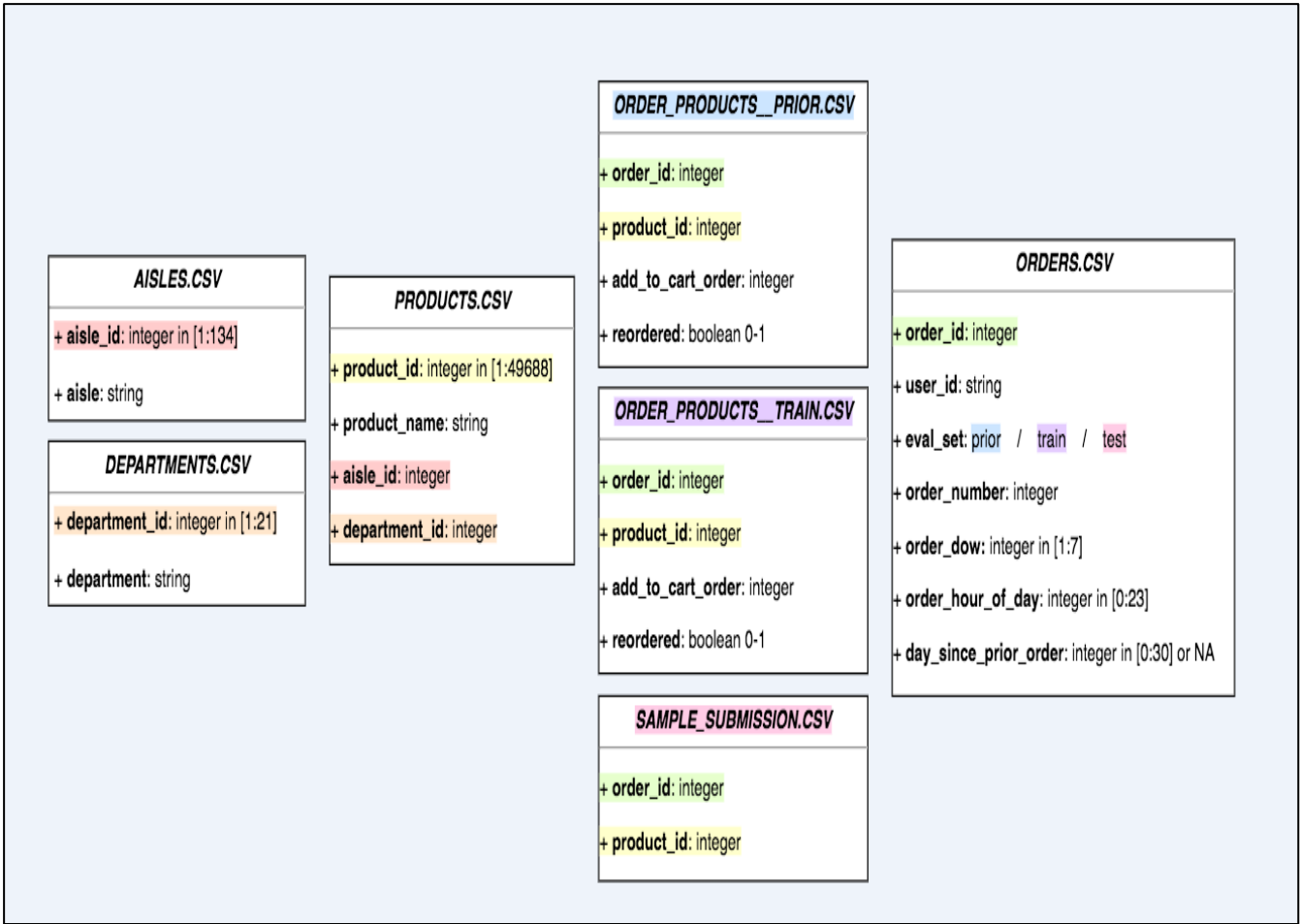
departments (21 rows)	
department_id	department identifier
department	the name of the department

This file contains department id along with department name.

department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol
6	international
7	beverages
8	pets
9	dry goods pasta
10	bulk

```
Observations: 21
Variables: 2
$ department_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
$ department    <chr> "frozen", "other", "bakery", "produce", "alcohol", "internatio..."
```





## OUR APPROACH

### Feature Extraction

Based on the data provided by Instacart, we crafted mainly three categories of features about products, users and orders. Given these complex data files relationships, it involves multiple steps of data munging to efficiently create features and join them.

Table 1: **Product Related Features**

Features	Description
Aisle id	Aisle category for a product
Department id	Department category for a product
Product Orders	How many times a product was ordered
Department wise reorder	How many time products was reordered in each department

Table 2: **User Related Features**

Features	Description
Total Orders	Total orders from a user
Total Items	Total products ordered by a user
Total distinct Items	Number of distinct products ordered by user.
Re-ordered Products	Products being reordered most frequently before.
Average basket	Average number of products in an order for a user

Table 3: **Orders related Features**

Features	Description
Day of Week	Day of week for an order
Order hour of day	Hour of day for an order
Days since prior order	Number of days since the last order for same user
Day of week vs hour of day	Frequency of day of week divided by hour of day

## Why Light GBM?

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'.

## Major Advantages

**Faster training speed and higher efficiency:** Light GBM use histogram based algorithm i.e. it buckets continuous feature values into discrete bins which fasten the training procedure.

**Lower memory usage:** Replaces continuous values to discrete bins which result in lower memory usage.

**Better accuracy than any other boosting algorithm:** It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max\_depth parameter.

**Compatibility with Large Datasets:** It can perform equally good with large datasets with a significant reduction in training time as compared to XGBOOST.

Parallel learning supported.

### Leaf wise tree growth in LightGBM as compared to Level wise tree growth in XGBOOST

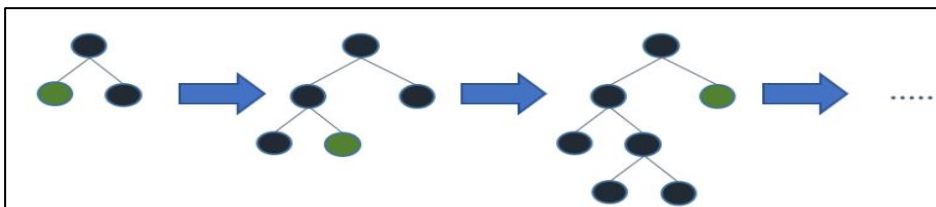


Figure 1: Leaf wise tree growth

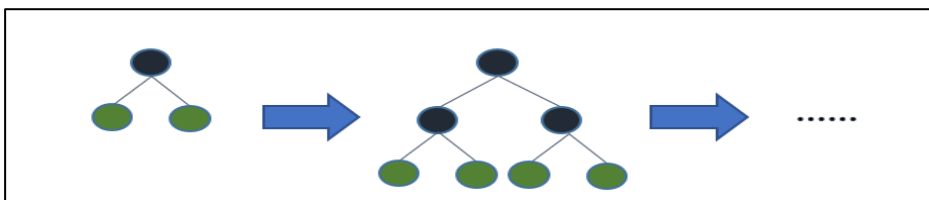


Figure 2: Level wise tree growth

## Exploratory Data analysis

We started by analysing each data set to summarize the features, characteristics and gain useful insights, often with visualizations. As mentioned in Data section above, there are six data tables provided, which are:

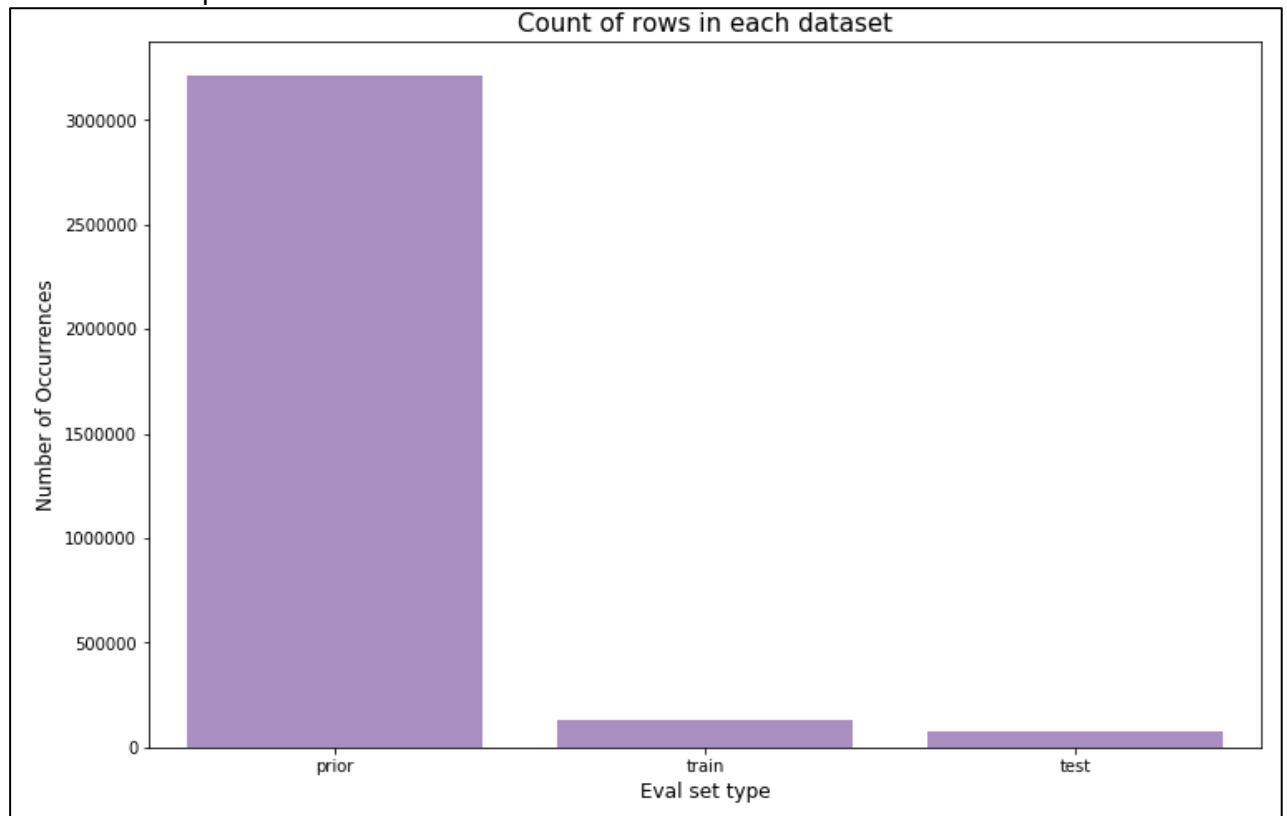
- Aisles
- Departments
- Order\_Products\_Prior
- Order\_Product\_Train
- Orders
- Products

We first analysed the tables and columns to see which kind of data we have. There are 19 columns in total. Below screen-shot contains header of each csv file along with first 2 records.

Orders						
	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day \
0	2539329	1	prior	1	2	8
1	2398795	1	prior	2	3	7
days_since_prior_order						
0			nan			
1			15.000			
*****						
Order_products_prior						
	order_id	product_id	add_to_cart_order	reordered		
0	2	33120	1	1		
1	2	28985	2	1		
*****						
Order_products_train						
	order_id	product_id	add_to_cart_order	reordered		
0	1	49302	1	1		
1	1	11109	2	1		
*****						
Products						
	product_id	product_name	aisle_id	department_id		
0	1	Chocolate Sandwich Cookies	61	19		
1	2	All-Seasons Salt	104	13		
*****						
Aisles						
	aisle_id	aisle				
0	1	prepared soups	salads			
1	2	specialty cheeses				
*****						
Department						
	department_id	department				
0	1	frozen				
1	2	other				

We have unique IDs for each order and product. In Order\_products\_prior and order\_product\_train tables, we have the same columns but the only difference is that one of them is trained data while the other one is the test data.

Below is the representation of total number of rows in each data-set:



## Grouping of data-sets

This is an important part of our exploratory analysis, wherein we put together different data-sets/features together to identify few major transactions. Such as:

- Total number of orders placed by users.
- Overall maximum order number
- Frequency of order by week day
- Percentage of products that are reordered.
- Number of products in given order

Let's now go over these grouping of data-sets through visualization that we performed.

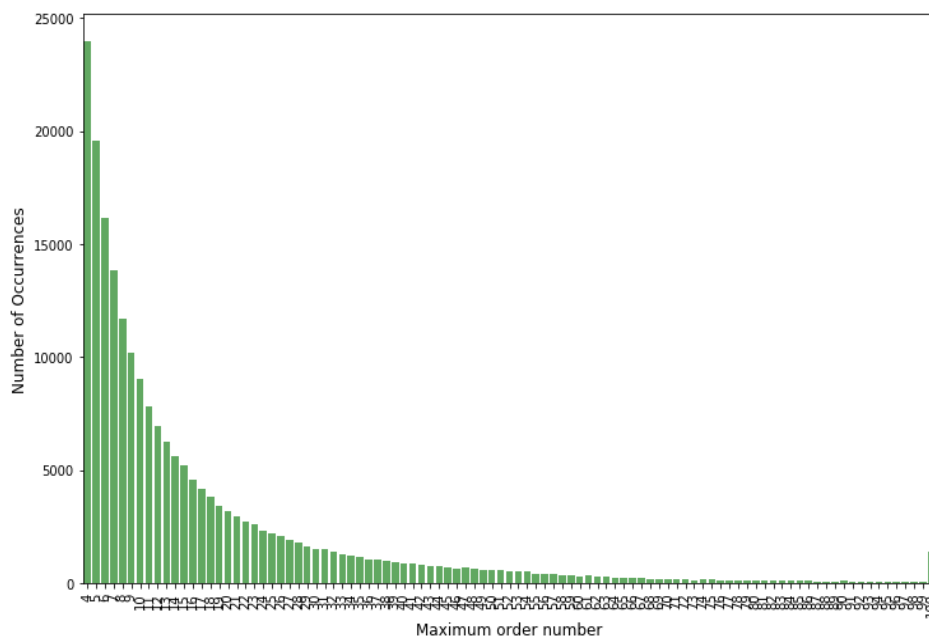
### Total number of orders placed by users

We grouped order number from order data set with user id to identify user id with maximum number of orders and displayed outcome of first four users in below screen:

	user_id	order_number
0	1	11
1	2	15
2	3	13
3	4	6

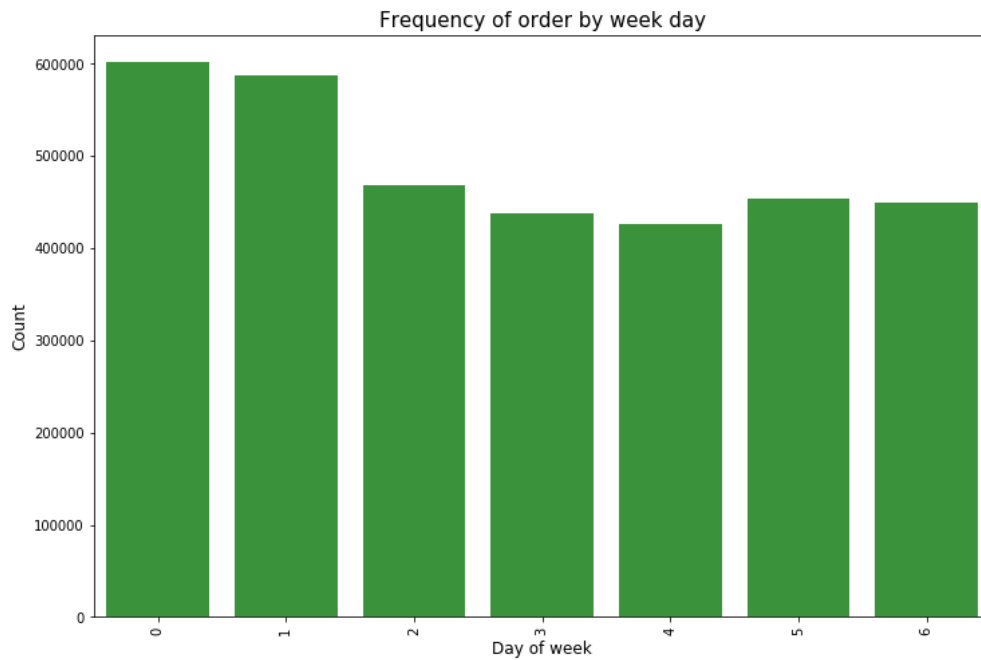
### Overall maximum order number

Here we can see the number of orders placed by all users. We concluded that maximum no of orders placed is capped at 100 and minimum is 4.



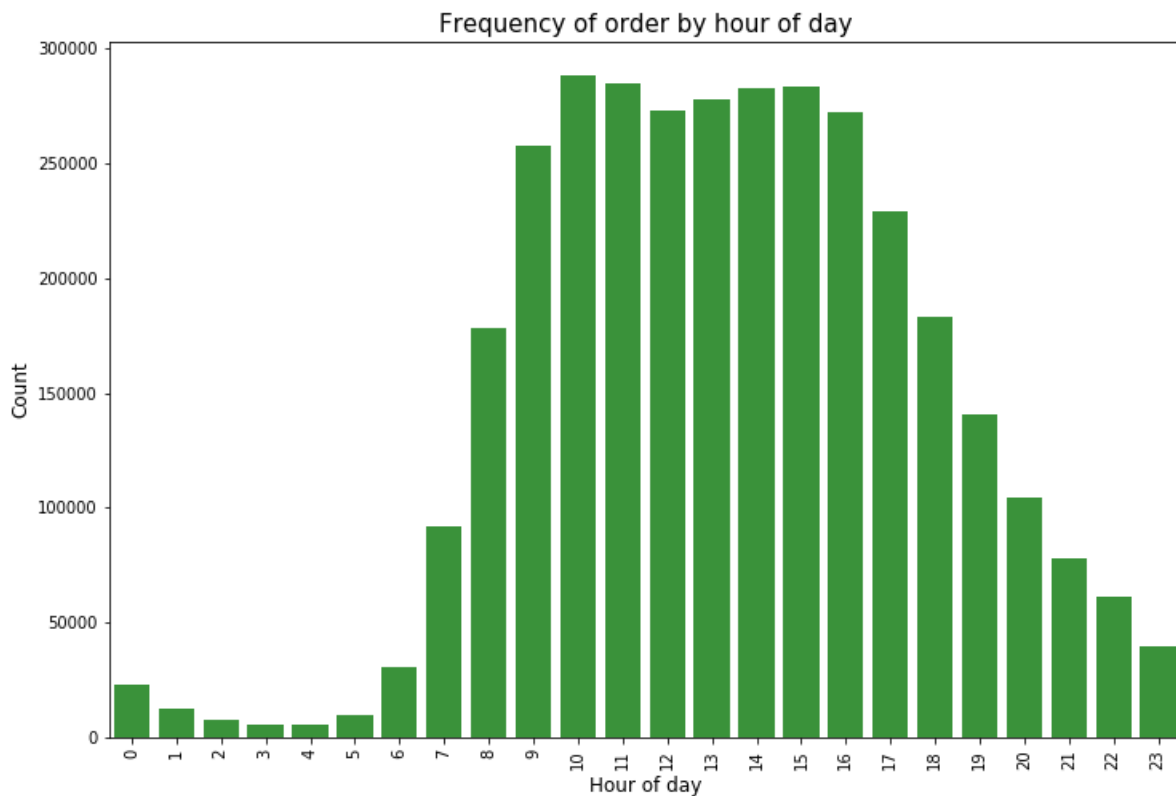
## No. of Order by week day

We are now determining on which day of week; more number of orders was placed and we figured out that maximum numbers of orders were placed during weekend.



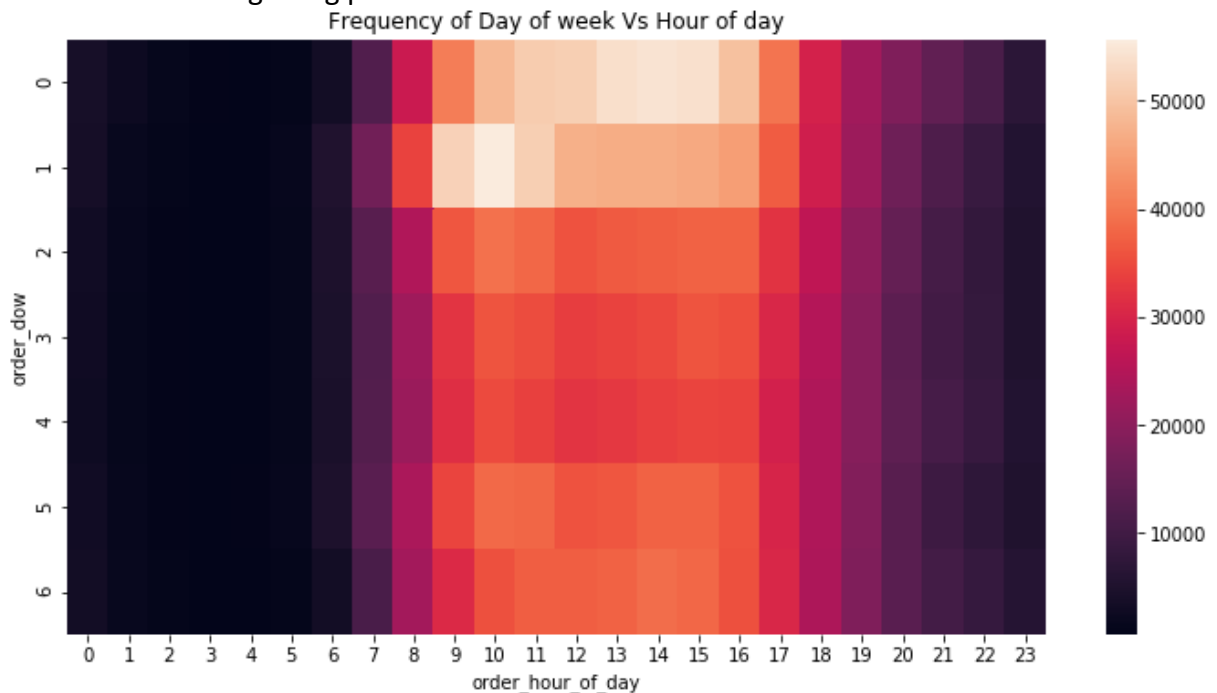
## No. of orders by hour of day

The below graph is plotted to show during which hour of day, more number of orders were placed.



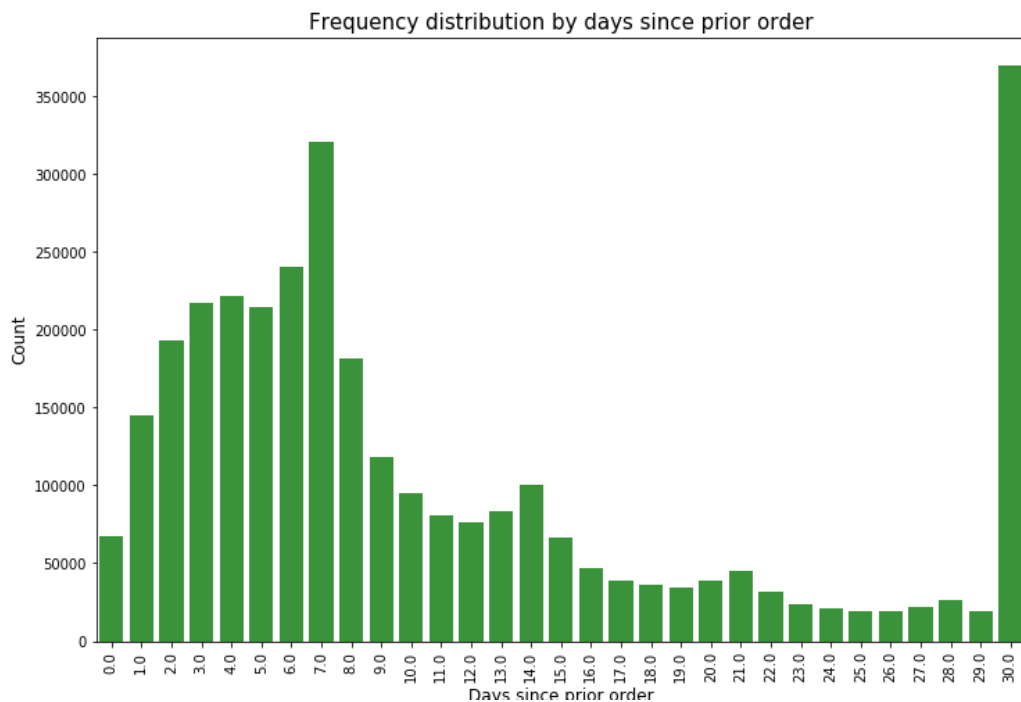
## Frequency of Day of week vs Hour of day

To analyse during which day in week and at which period, we are getting maximum orders, we combine day of week and hour of day and figured out that during weekends and at day time more orders are getting placed.



## Frequency Distribution by Days since Prior Order

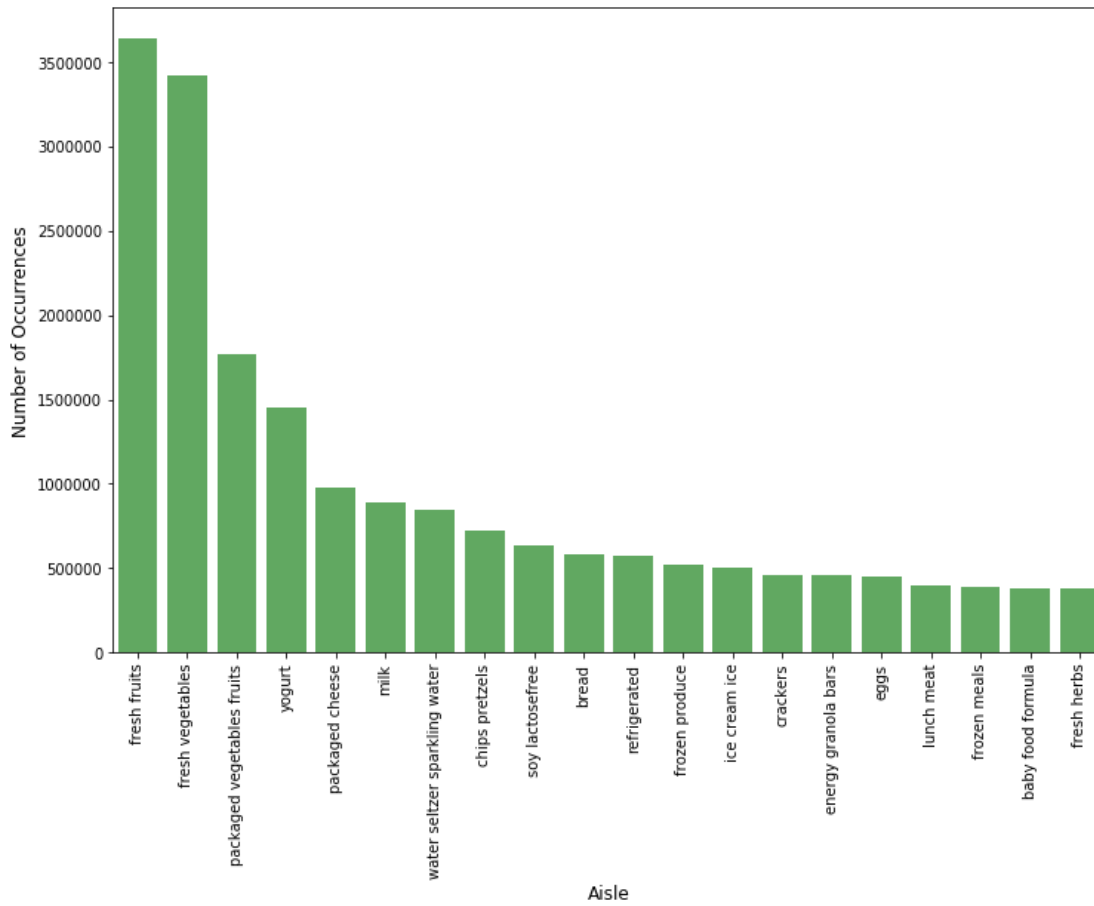
We plotted frequency distribution by days since last order by counting `days_since_prior_order` in order data set. We found that there is abrupt rise in peaks at 7th, 14th and 30th.





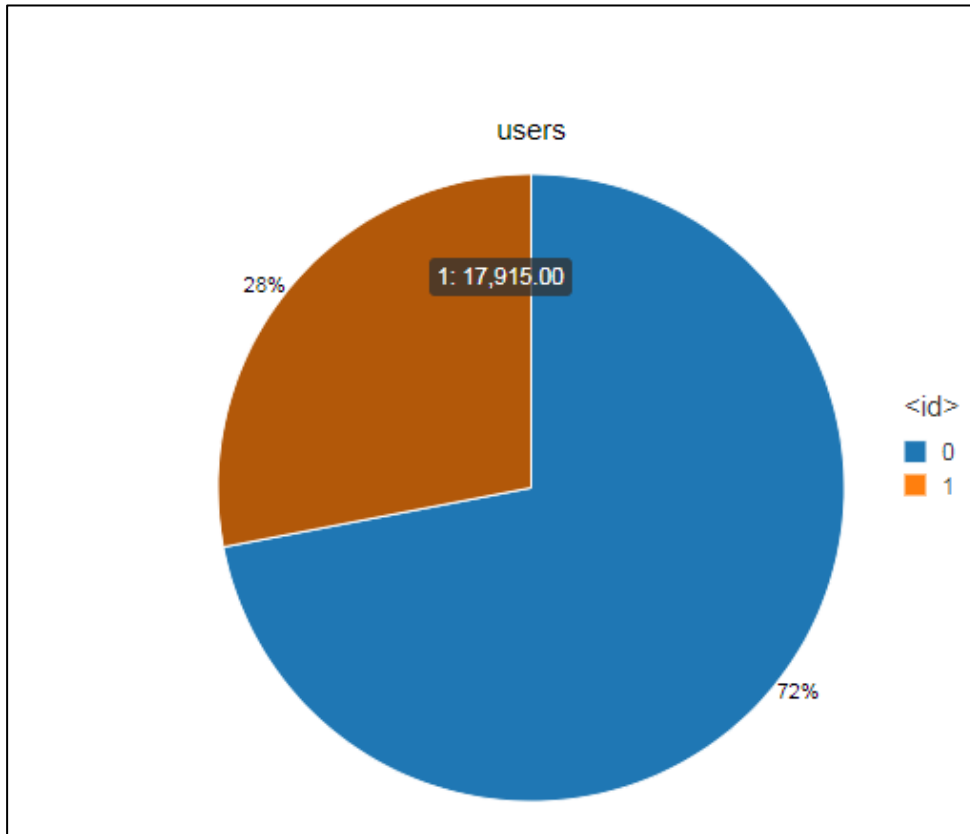
## Most Important Aisles

Below graphs shows all important aisles and number of its occurrences:



## Active / Inactive Users

We want to see how many active and inactive users we have in our dataset. There are 17915 distinct inactive users with a share of 28% and 46017 distinct active users with a share of 72%, so we can set our focus mainly on this 18% share of users because their orders will probably provide us more certain information to make estimations and they are the ones who will likely do shopping regularly and continuously.



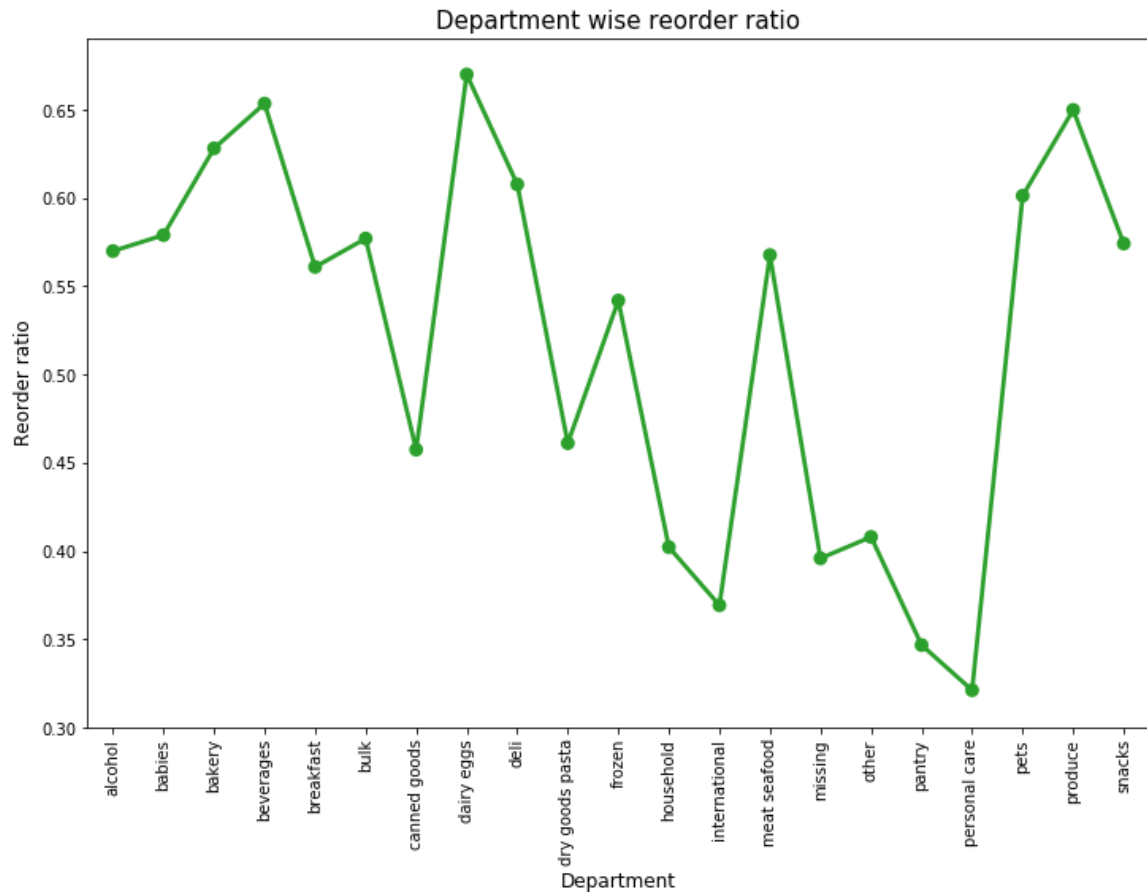
## Frequency of Products ordered

	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
11	Organic Yellow Onion	113426
12	Organic Garlic	109778
13	Organic Zucchini	104823
14	Organic Blueberries	100060
15	Cucumber Kirby	97315
16	Organic Fuji Apple	89632
17	Organic Lemon	87746
18	Apple Honeycrisp Organic	85020
19	Organic Grape Tomatoes	84255

## Reorder Ratio

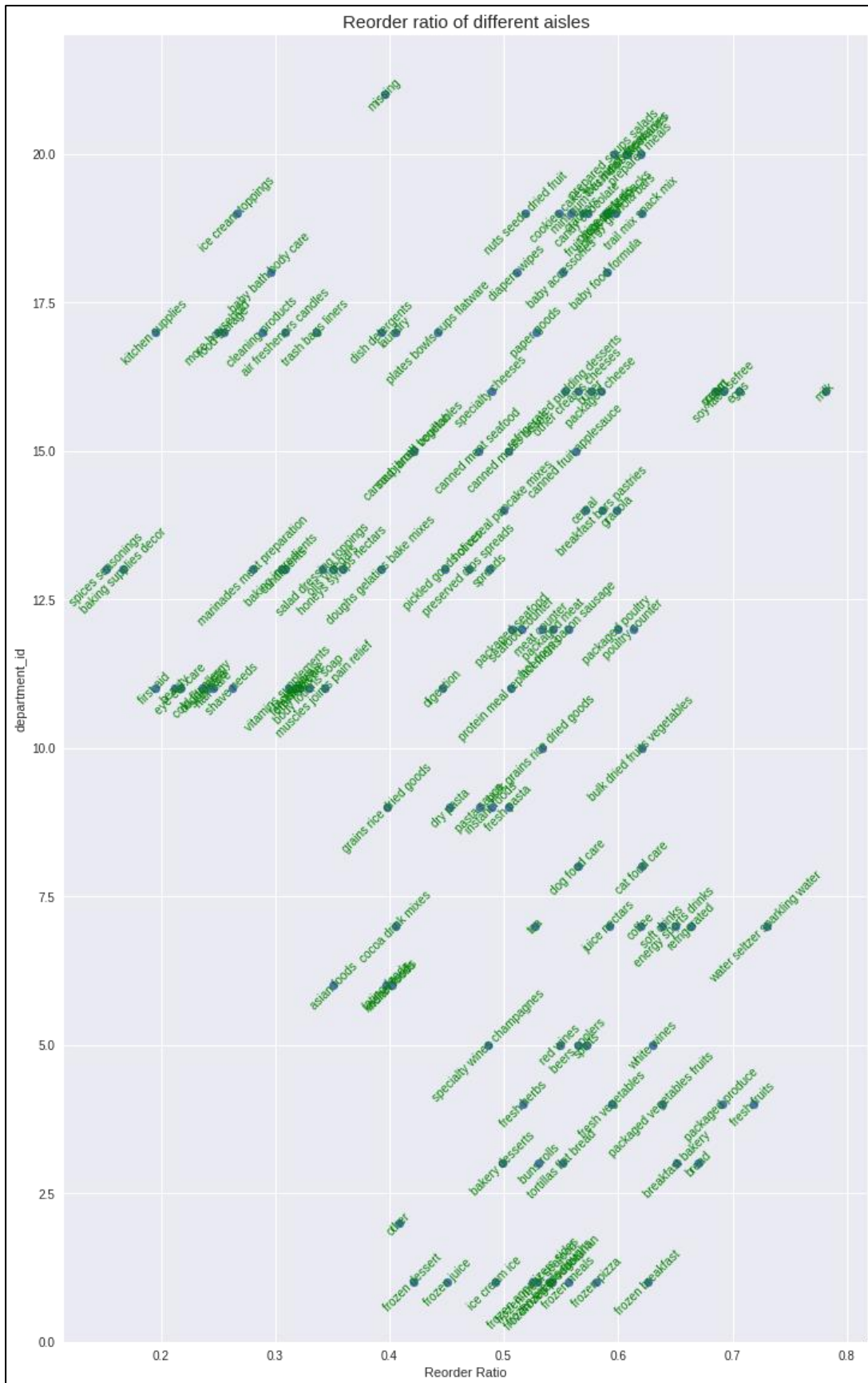
- Department wise reorder Ratio

We determine which department had maximum reordered item by grouping department and reordered column in order\_product\_prior data set by aggregating its mean and below results were obtained:



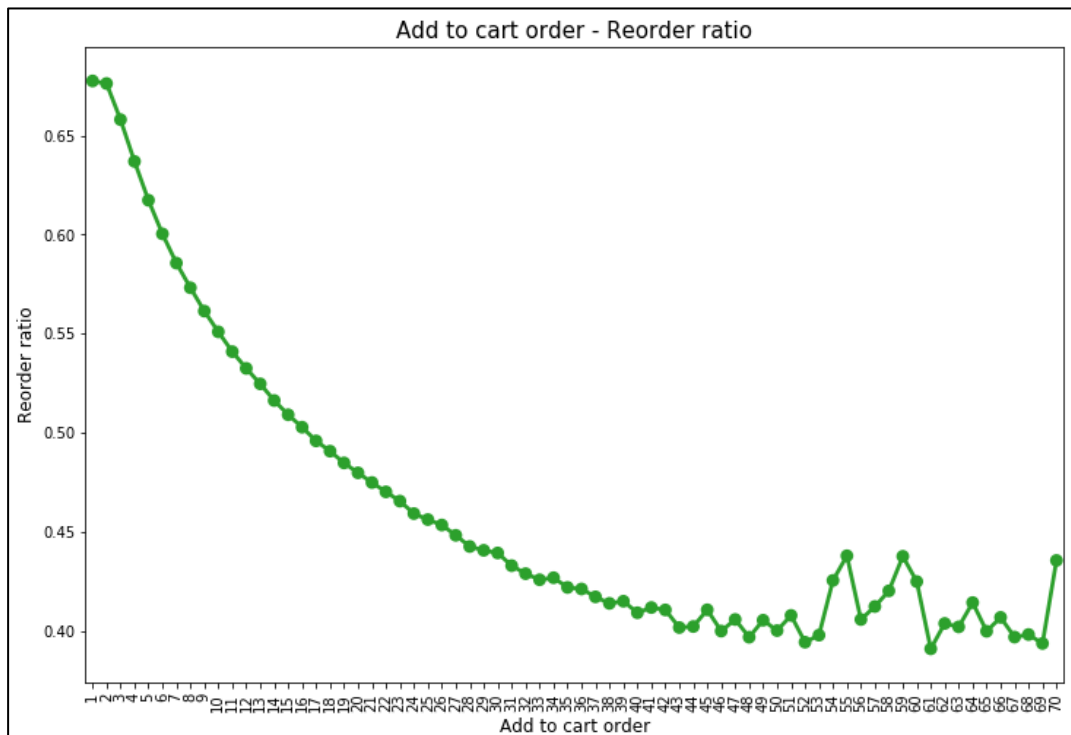
- **Reorder Ratio of Different Aisles + Department**

Similarly, we determined aisle with maximum number of reordered.



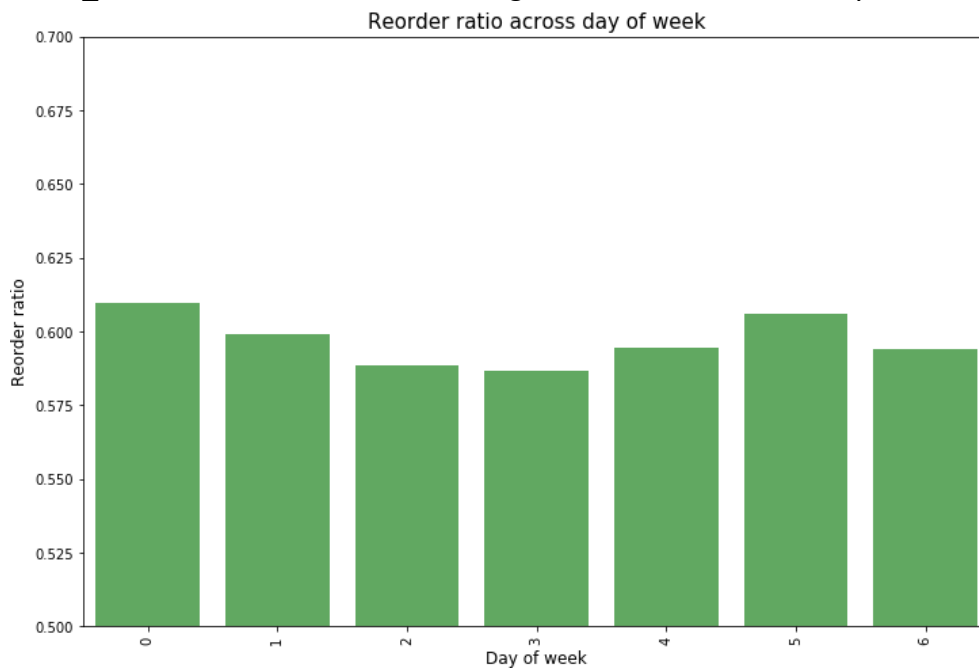
- **Add to Cart Order- Reorder Ratio**

We then explored the relationship between how order of adding the product to the cart affects the reorder ratio. Looks like the products that are added to the cart initially are more likely to be reordered again compared to the ones added later.



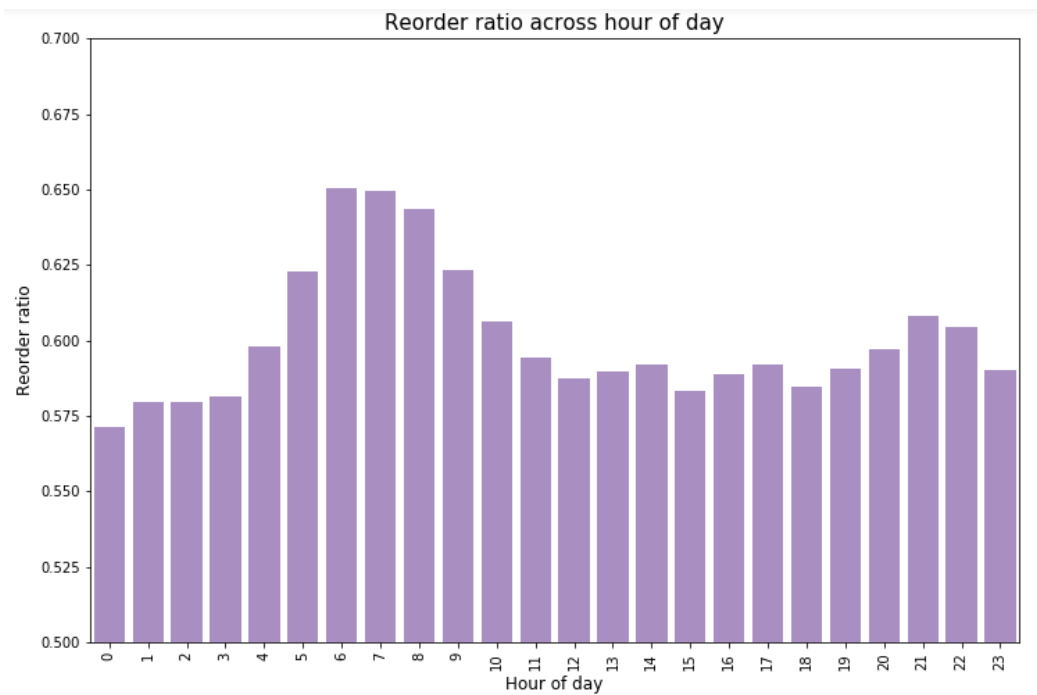
- **Reorder Ratio across day of week**

To evaluate reorder ratio across week, we merge train dataset with order data set and grouped order\_dow and reordered columns to get reorder ratio across day of week.

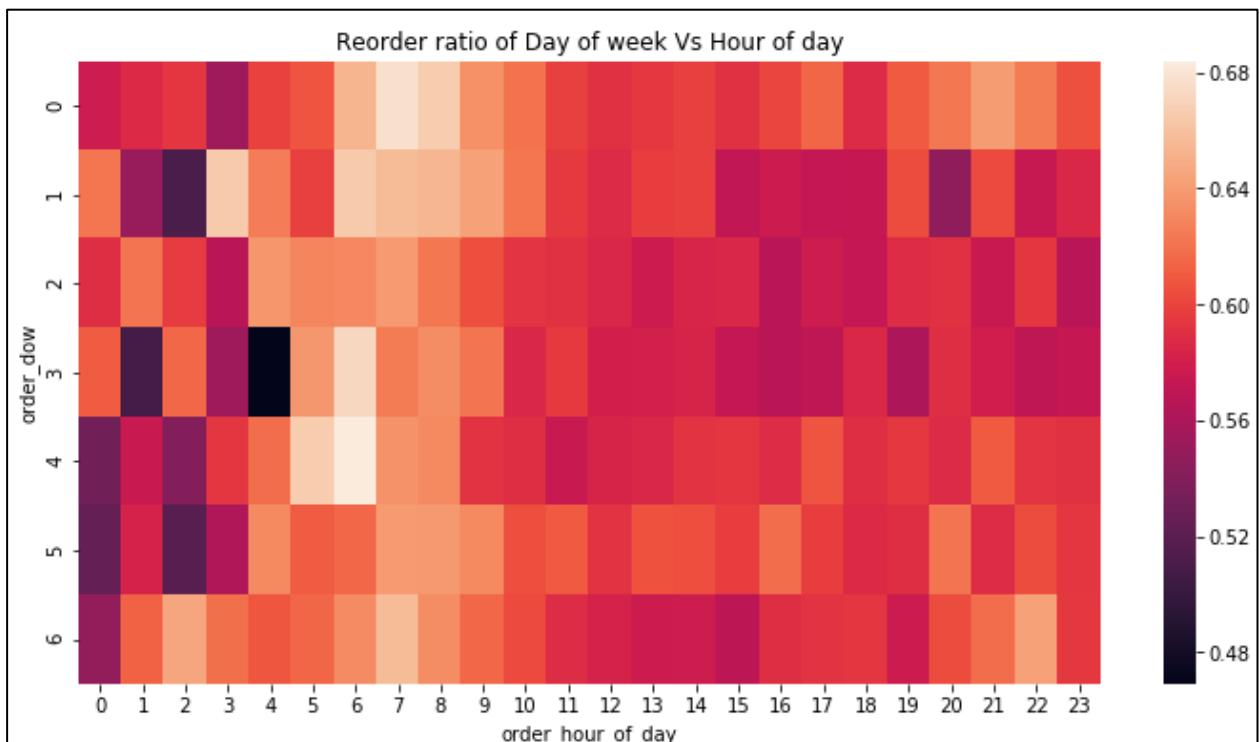


- **Reorder Ratio across hour of day**

Similarly, we grouped `order_hour_of_day` and reordered columns to find out reorder ratio across hour of day.



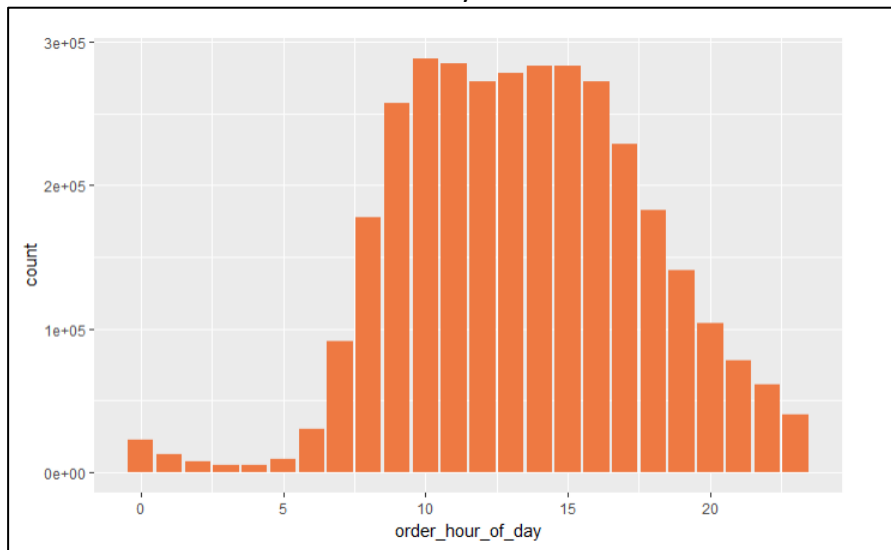
- **Reorder Ratio of Day of week vs Hour of day**



## When do people order?

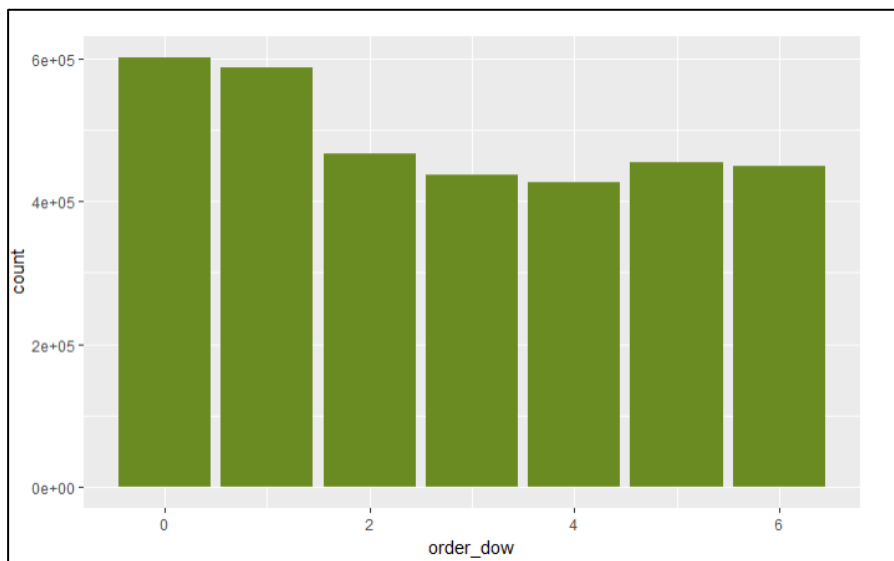
### Hour of Day

There is a clear effect of hour of day on order volume. Most orders are between 8.00-18.00



### Day of Week

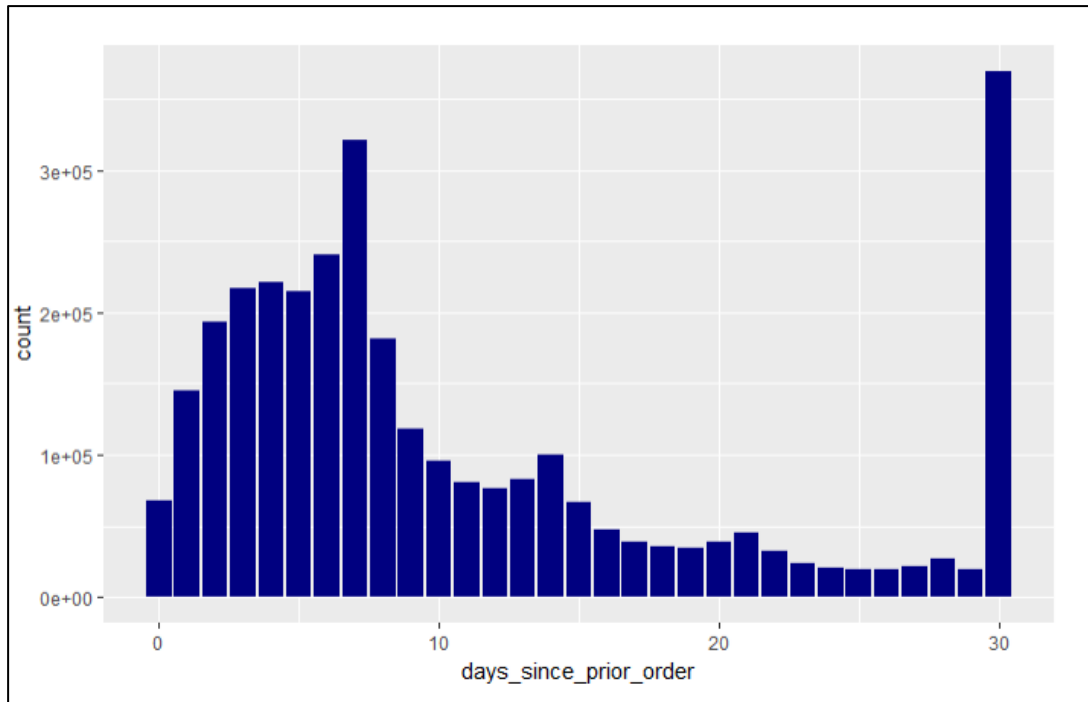
There is a clear effect of day of the week. Most orders are on days 0 and 1. Unfortunately there is no info regarding which values represent which day, but one would assume that this is the weekend.





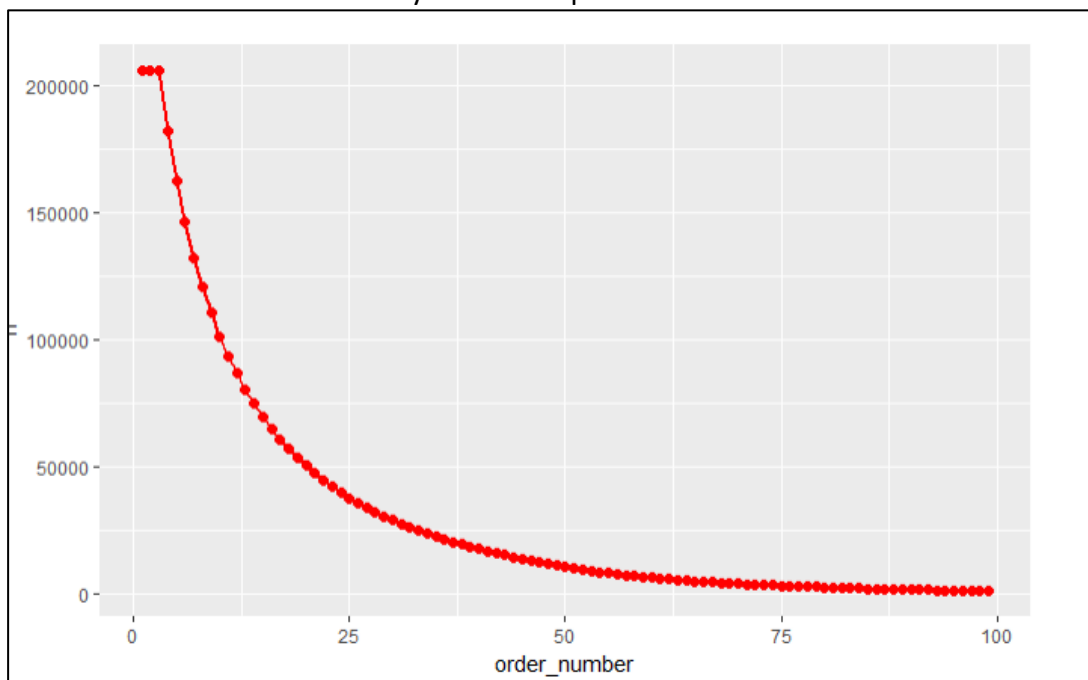
## When do they order again?

People seem to order more often after exactly 7 days, 14 days and 30 days.



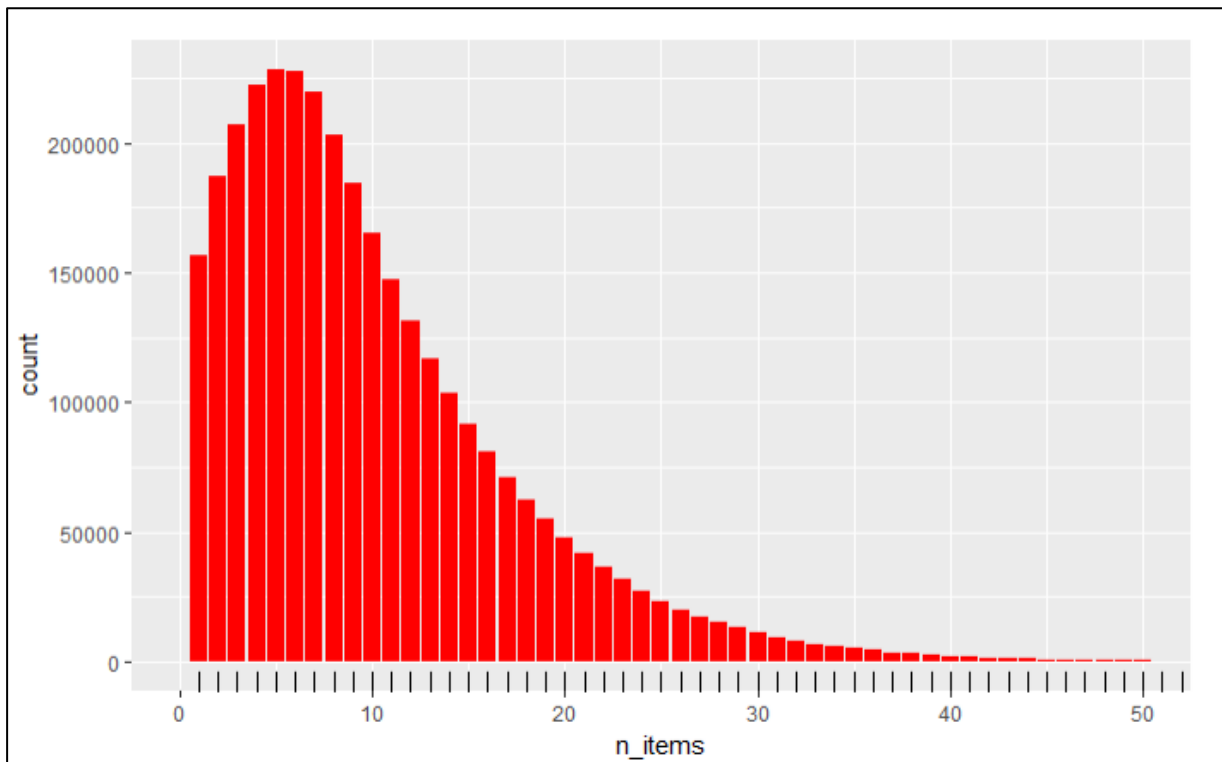
## How many prior orders are there?

We can see that there are always at least 3 prior orders.



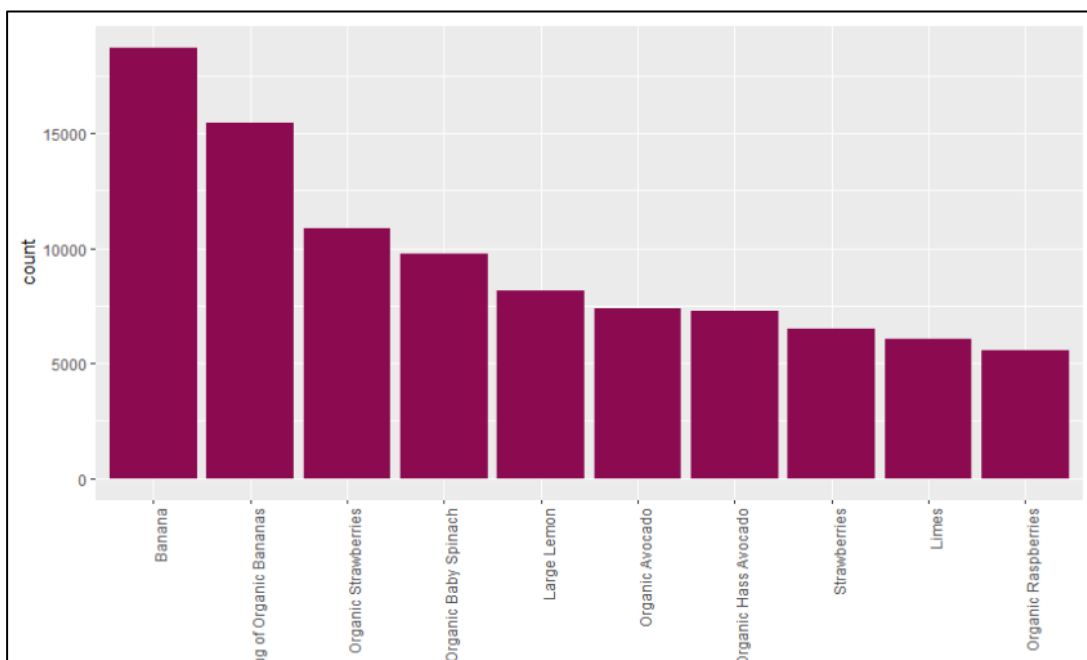
## How many items do people buy?

We can see that people most often order around 5 items. The distributions are comparable between the train and prior order set.



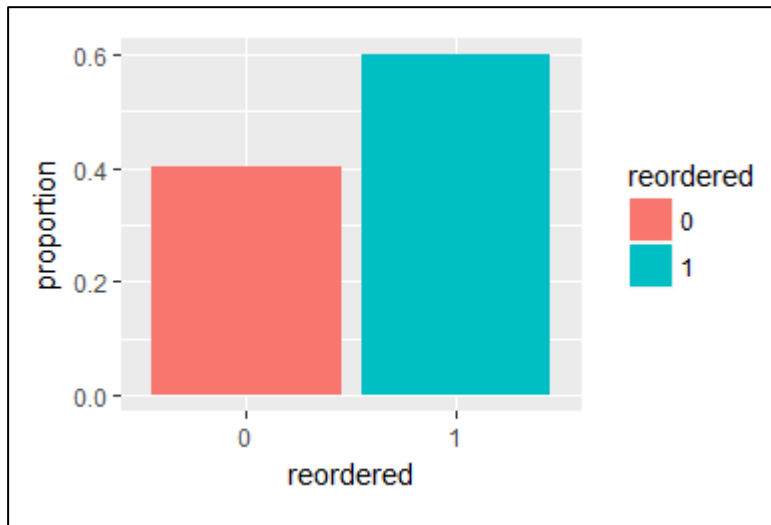
## Bestsellers

Let's have a look which product are sold most often (top10). And the clear winner is **Bananas**



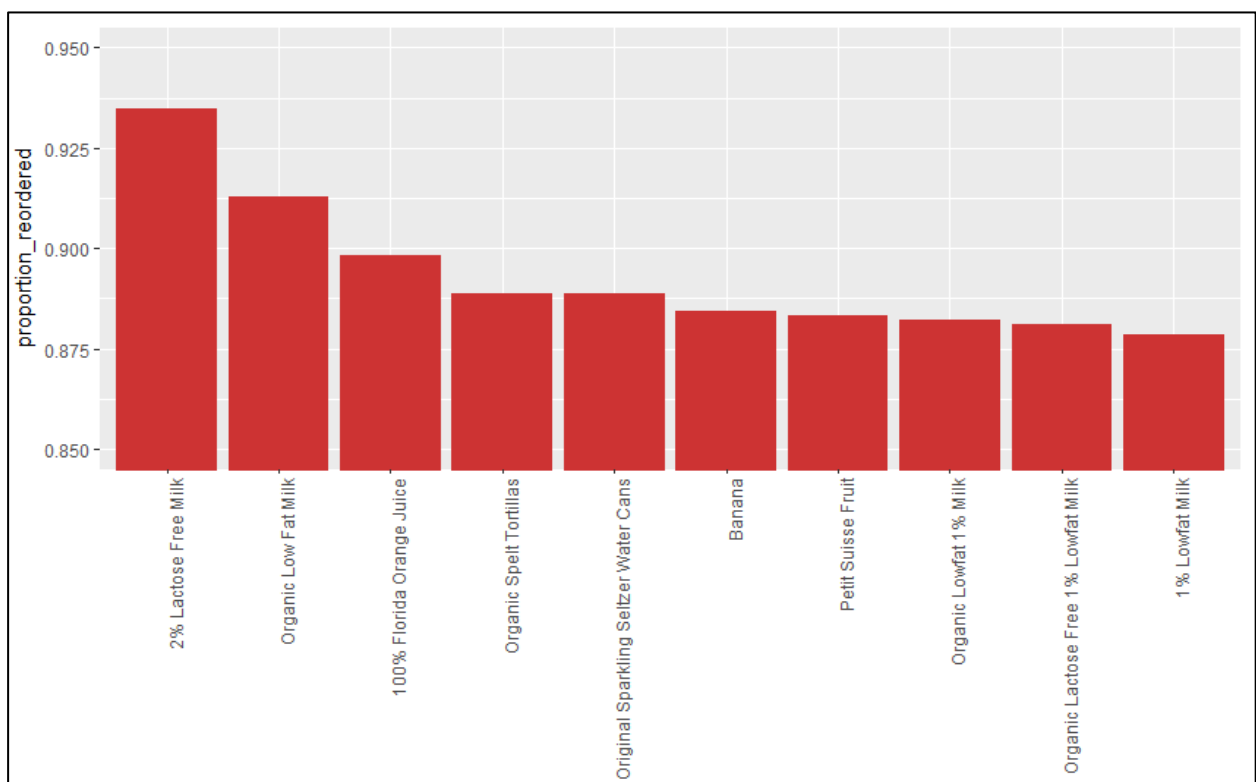
## How often do people order the same items again?

59% of the ordered items are reorders.



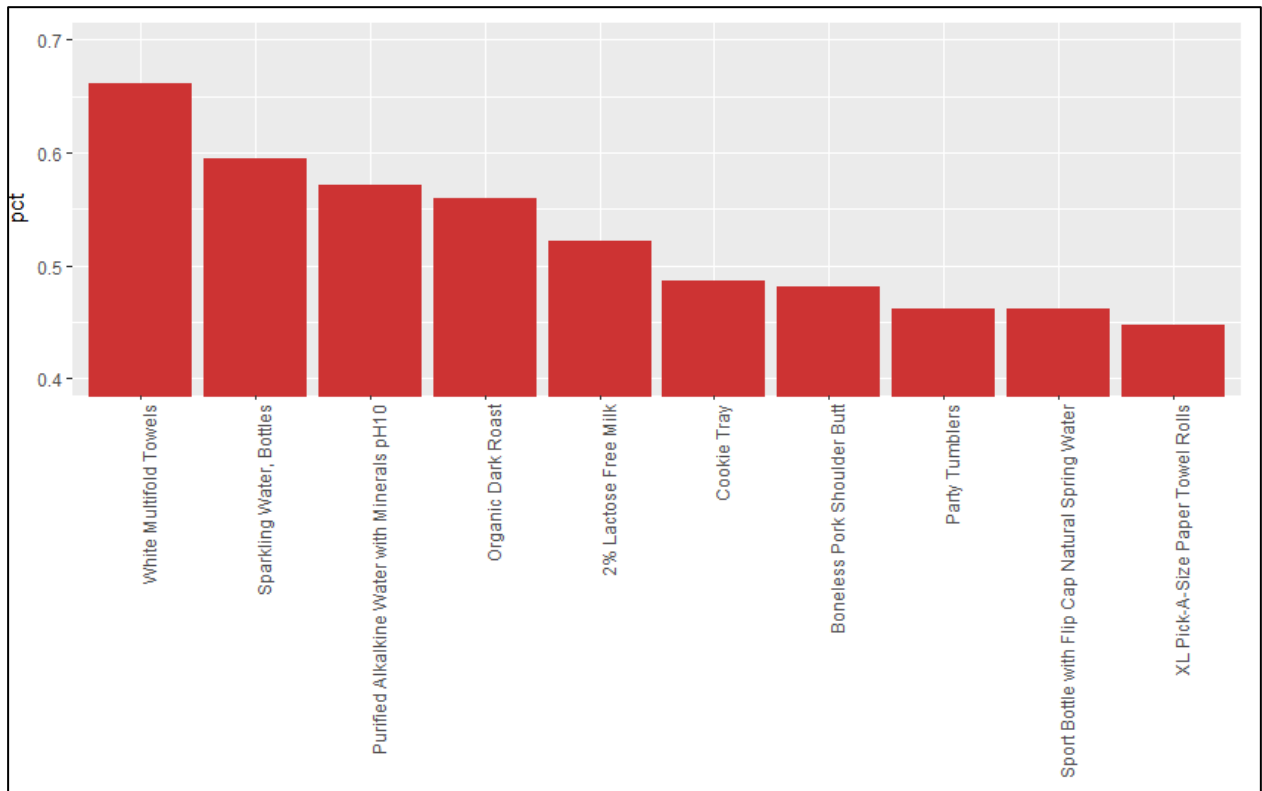
## Most often reordered

These 10 products have the highest probability of being reordered.



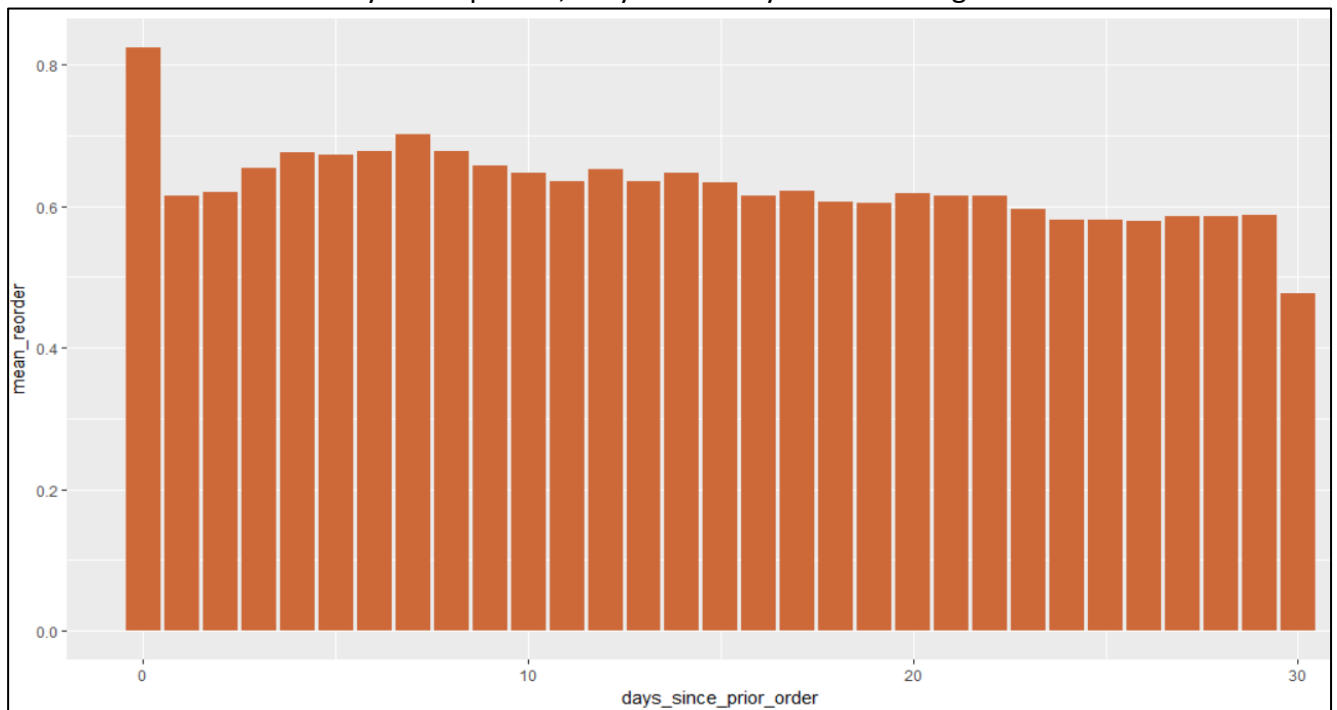
## Which item do people put into the cart first?

People seem to be quite certain about Multifold Towels and if they buy them, put them into their cart first in 66% of the time.



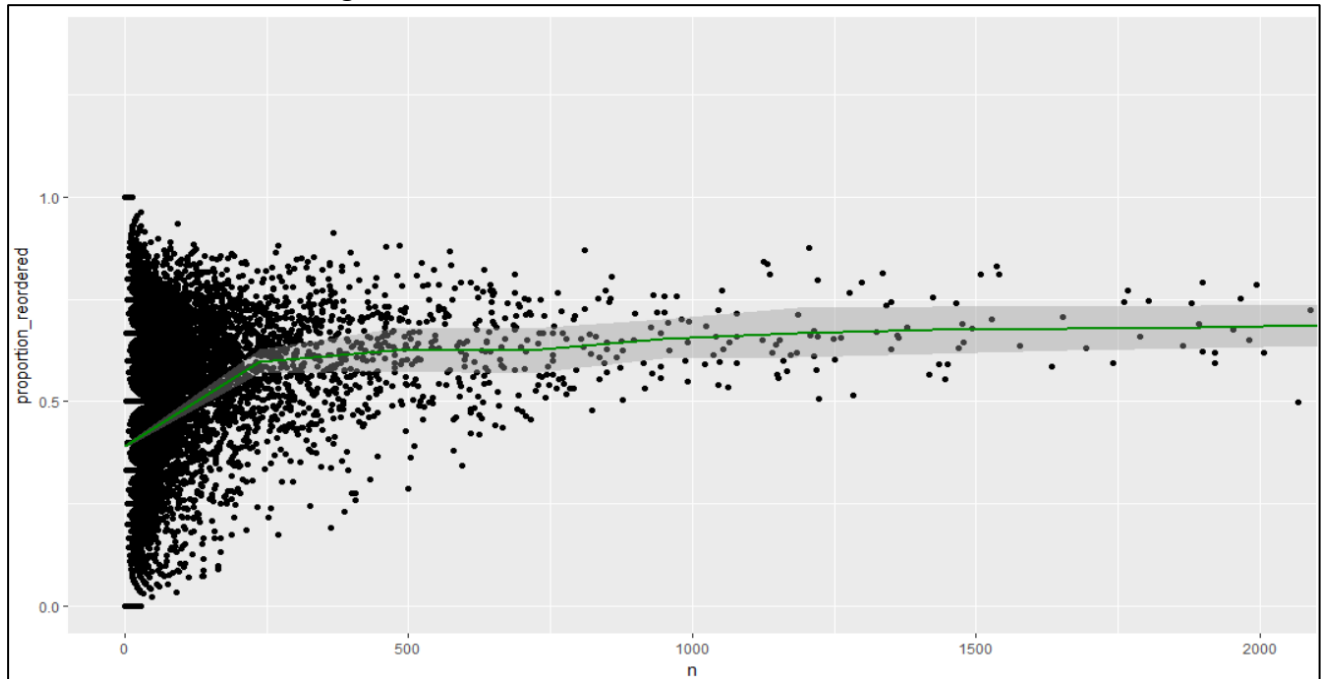
## Association between time of last order and probability of reorder

We can see that if people order again on the same day, they order the same product more often. Whereas when 30 days have passed, they tend to try out new things in their order.



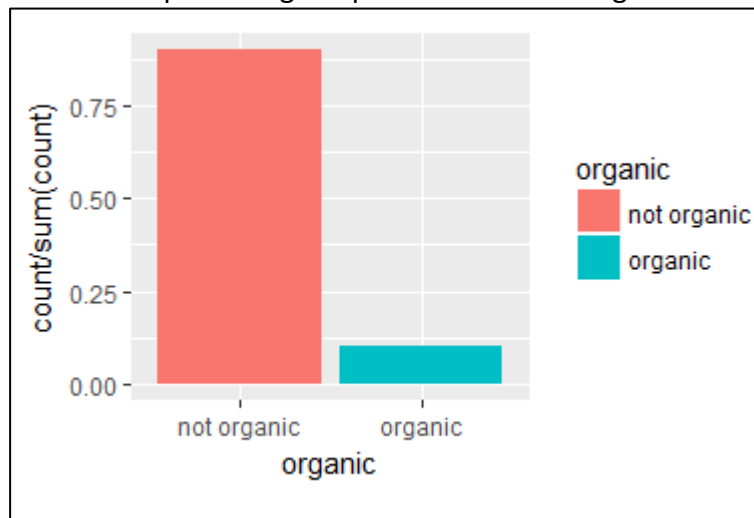
## Association between number of orders and probability of reordering

Products with a high number of orders are naturally more likely to be reordered. However, there seems to be a ceiling effect.



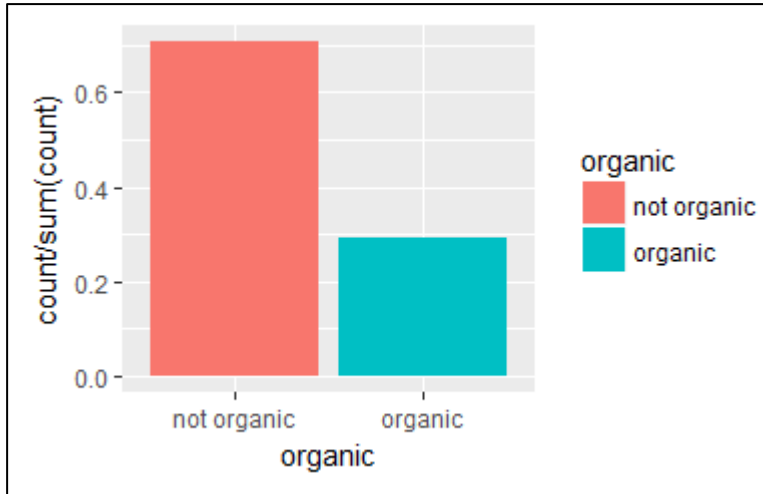
## Organic vs Non-Organic Products

What is the percentage of products that are organic vs. not organic?



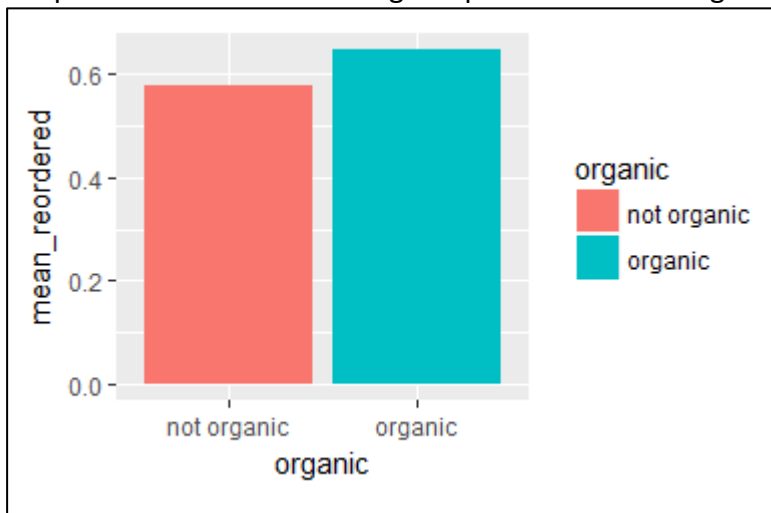
## Organic vs Non-Organic Orders

What is the percentage of orders that are organic vs. not organic?



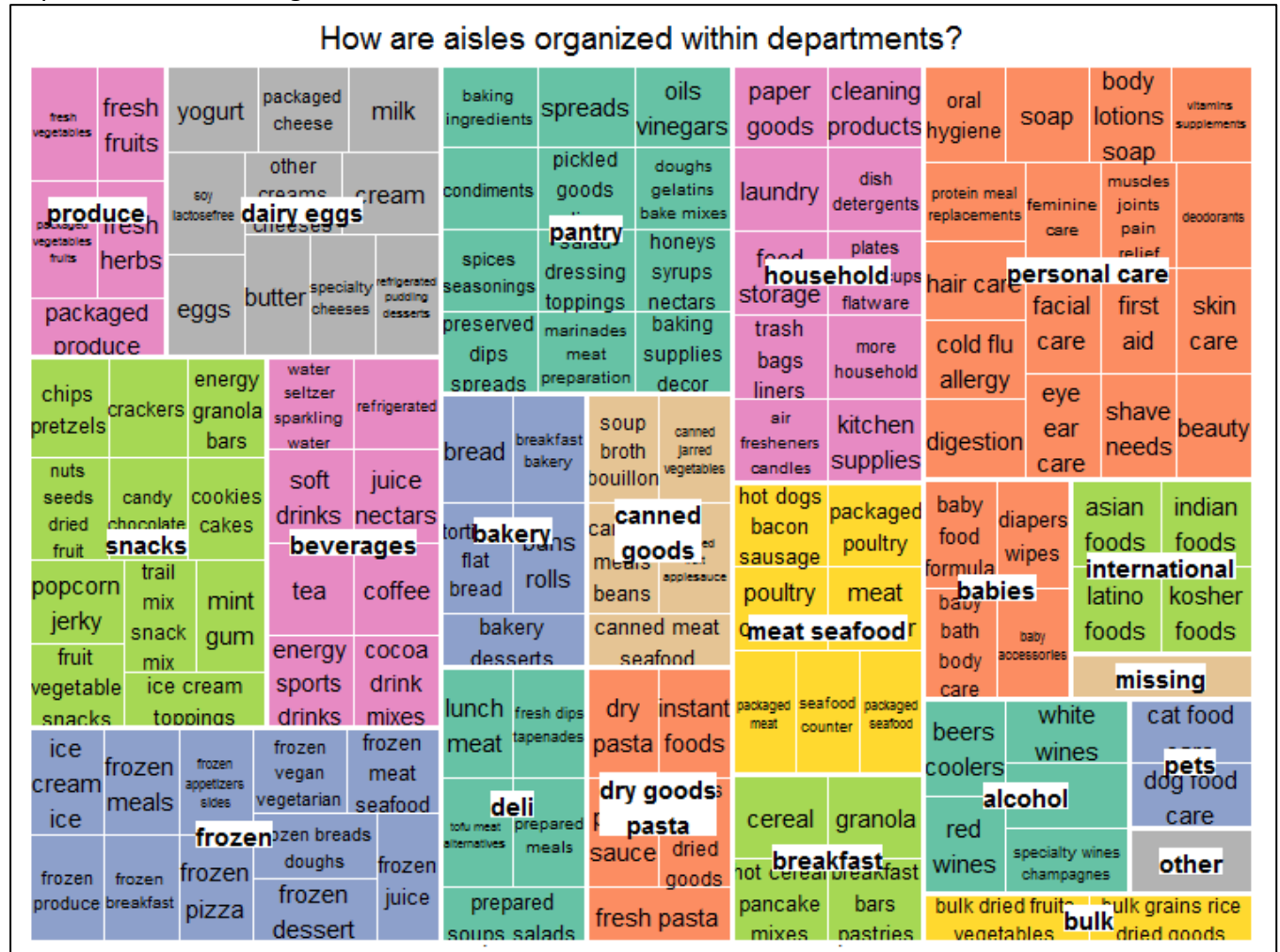
## Reordering Organic vs Non-Organic

People more often reorder organic products vs non-organic products.



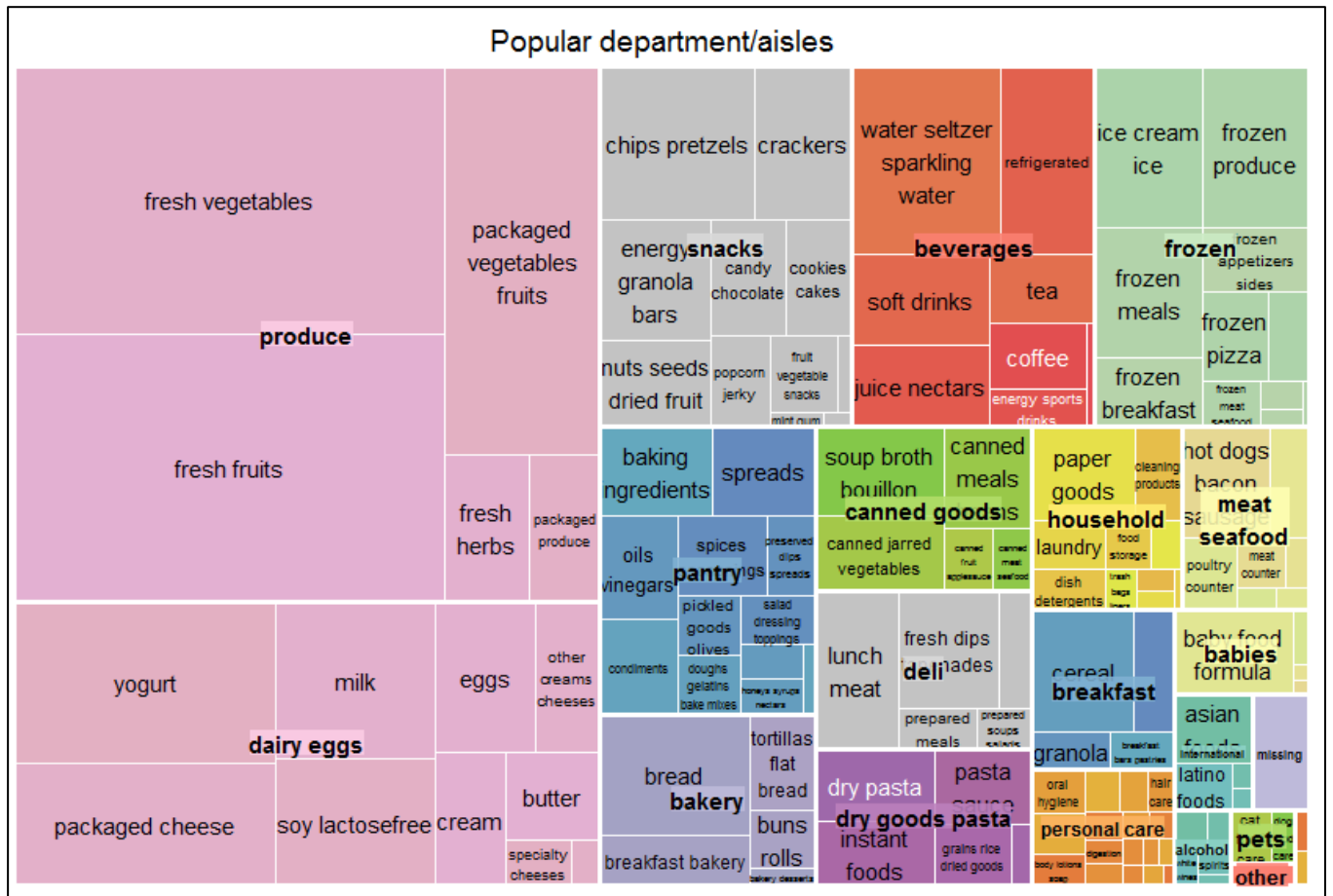
## Visualizing the Product Portfolio

We used treemap package to visualize the structure of product portfolio. In total there are 21 departments containing 134 aisles.



## How often are products from the department/aisle sold?

The size of the boxes shows the number of sales.





# Market Basket Analysis

## INTRODUCTION

Mining frequent itemsets and association rules is a popular and well researched approach for discovering interesting relationships between variables in large databases.

Understanding buying patterns can help to increase sales in several ways. If there is a pair of items, X and Y, that are frequently bought together.

- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items.
- Advertisements on X could be targeted at buyers who purchase Y.
- X and Y could be combined into a new product, such as having Y in flavours of X.

Association rules analysis is a technique to uncover how items are associated to each other. There are three common ways to measure association.

**Support:** This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. If it is discovered that sales of items beyond a certain proportion tend to have a significant impact on profits, one might consider using that proportion as the support threshold.

**Confidence:** This says how likely item Y is purchased when item X is purchased, expressed as  $\{X \rightarrow Y\}$ . This is measured by the proportion of transactions with item X, in which item Y also appears.

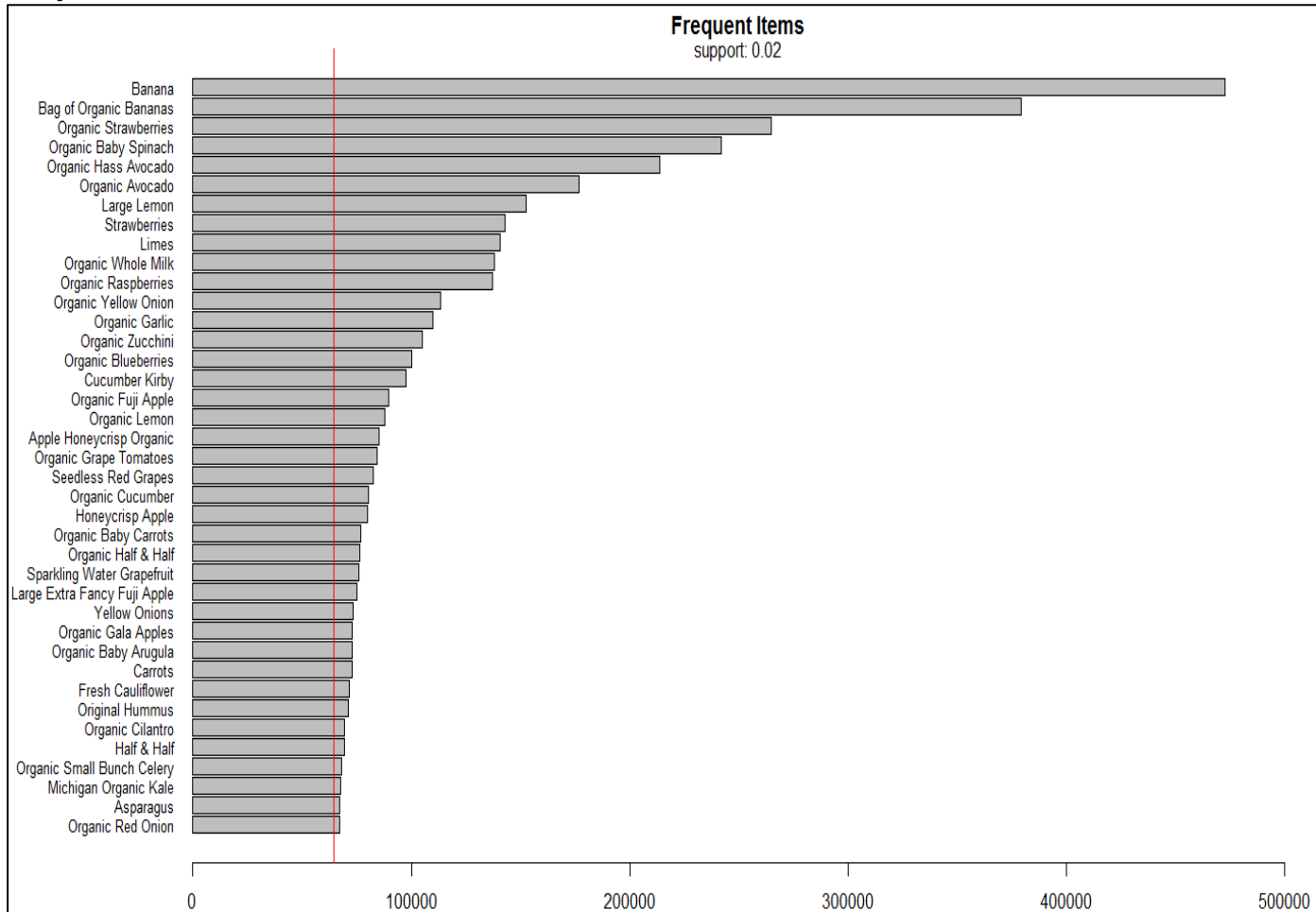
**Lift:** This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. A lift value greater than 1 means that item Y is likely to be bought if item X is bought, while a value less than 1 means that item Y is unlikely to be bought if item X is bought.

## METHODOLOGY

We performed market basket analysis on Instacart data using R. We used the popular library `arules`; which is a computational environment for mining association rules and frequent item sets.

Market Basket Analysis allows us to identify items that are frequently bought together. Typically, the output of an MBA is in the form of rules. The rules can be simple  $\{A \Rightarrow B\}$ , when a customer buys item A then it is (very) likely that the customer buys item B. More complex rules are also possible  $\{A, B \Rightarrow D, F\}$ , when a customer buys items A and B then it is likely that he buys items D and F.

**Step 1:** Find frequent items in the shopping baskets. The support is set to be 0.02



**Figure 1: Frequent Items: Support=0.02**

Banana is the most favourite item followed by Strawberries! Clearly vegetables and fruits are the most ordered products.

**Step 2:** Now we apply apriori algorithm to compute the frequent itemsets. We decrease the support threshold to consider the small probability of observing a frequent itemset of at least size 2.

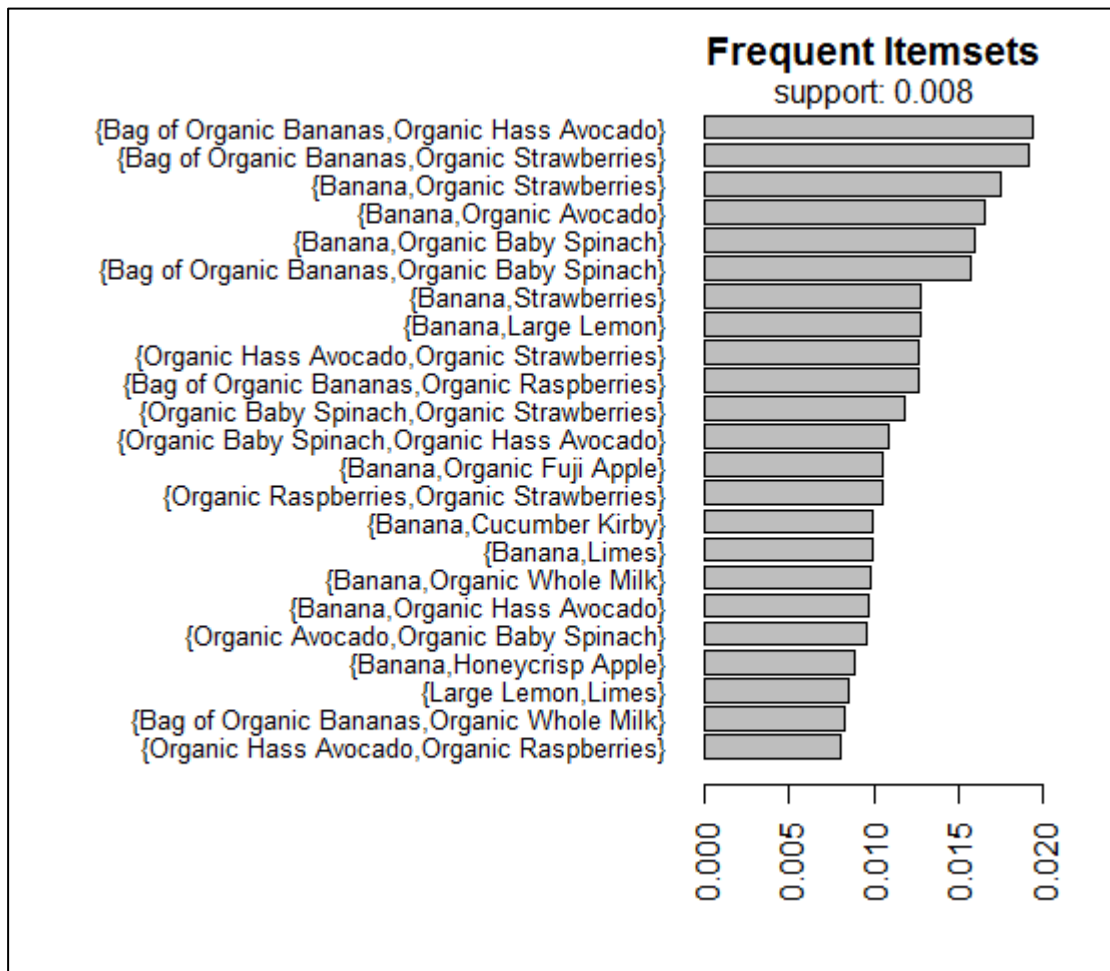


Figure 2: Frequent Itemsets; Support=0.008

Bananas being the most favourite item, rules the show here too! It is part of almost every frequent item set.

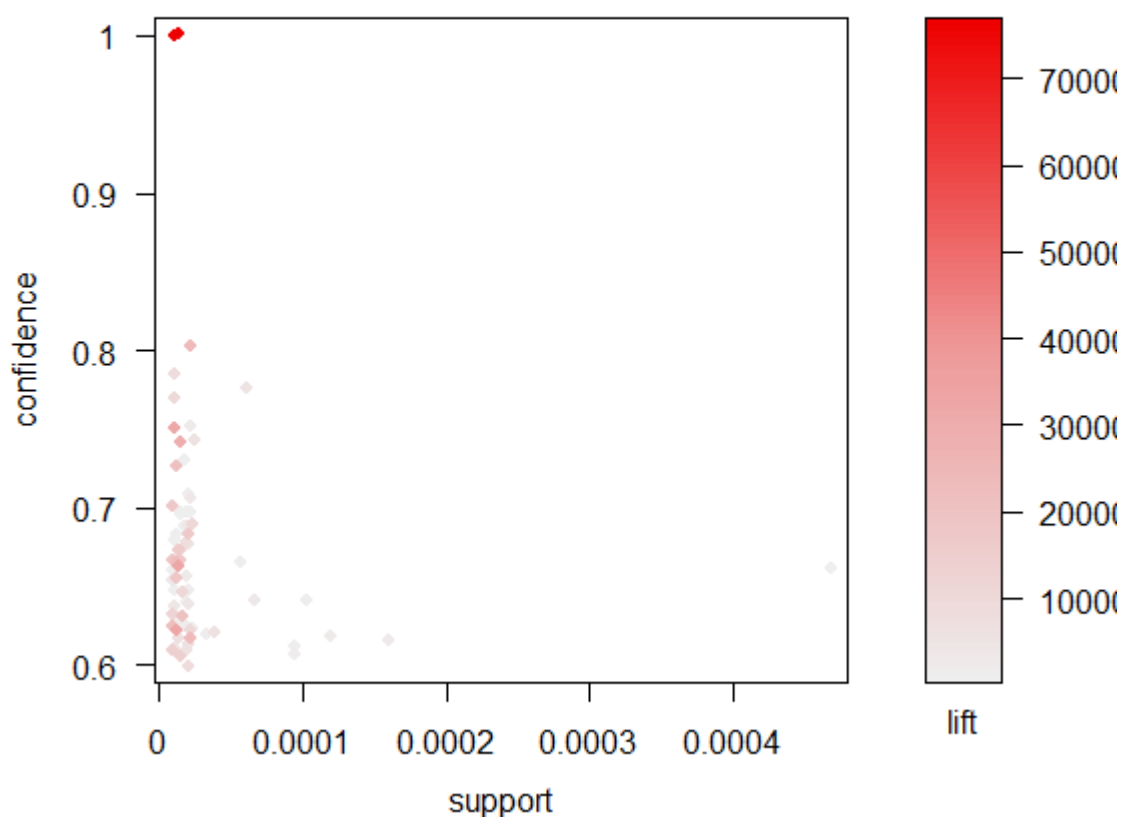
**Step 3:** Lets mine some association rules. First, we use a low support threshold and a high confidence to generate strong rules even for items that are less frequent.

**support = 0.00001, confidence = 0.6**

```
> summary(quality(rules1))
```

support	confidence	lift	count
Min. :0.00001026	Min. :0.6000	Min. : 4.28	Min. : 33.00
1st Qu.:0.00001275	1st Qu.:0.6234	1st Qu.: 408.02	1st Qu.: 41.00
Median :0.00001944	Median :0.6605	Median : 3938.58	Median : 62.50
Mean :0.00003041	Mean :0.6725	Mean : 8981.96	Mean : 97.78
3rd Qu.:0.00002146	3rd Qu.:0.6909	3rd Qu.:12403.96	3rd Qu.: 69.00
Max. :0.00046720	Max. :1.0000	Max. :76544.62	Max. :1502.00

**Scatter plot for 80 rules**



There are some rules with a large value of lift indicating a strong association between the items. Let's further investigate those critical rules.

S. No.	RULES		support	confidence	lift	count
1	{Moisturizing Facial Wash}	=> {Moisturizing Non-Drying	0.000	1.000	76544.620	42
2	{Moisturizing Non-Drying Facial	=> {Moisturizing Facial Wash}	0.000	1.000	76544.620	42
3	{Prepared Meals Simmered Beef Entree Dog Food}	=> {Prepared Meals Beef & Chicken Medley Dog Food}	0.000	0.621	32211.590	41
4	{Prepared Meals Beef & Chicken Medley Dog Food}	=> {Prepared Meals Simmered Beef Entree Dog Food}	0.000	0.661	32211.590	41
5	{Ocean Whitefish}	=> {Premium Classic Chicken	0.000	0.750	32148.740	33
6	{Ancient Grains Apricot Blended Low-Fat Greek Yogurt}	=> {Oats Ancient Grain Blend with Mixed Berry Low-Fat	0.000	0.742	29088.160	46
7	{Thousand Island Salad Snax}	=> {Raspberry Vinaigrette Salad	0.000	0.616	23030.140	69
8	{Raspberry Vinaigrette Salad	=> {Thousand Island Salad Snax}	0.000	0.802	23030.140	69
9	{Mighty Veggie Carrot Pear Pomegranate & Oats Vegetable & Fruit Smoothie}	=> {Organic Yogurt Baby Food}	0.000	0.667	22095.350	36
10	{Organic Baby Food Fruit Mashup Strawberry Patch 9+ Mo}	=> {Mashup Mama Bear Blueberry 7+ Mo}	0.000	0.623	21081.140	38

**Figure 3; Top 10 Rules by Lift**

S. No.	RULES		support	confidence	lift	count
1	{Moisturizing Facial Wash}	=> {Moisturizing Non-Drying	0.000	1.000	76544.619	42
2	{Moisturizing Non-Drying Facial	=> {Moisturizing Facial Wash}	0.000	1.000	76544.619	42
3	{Raspberry Vinaigrette Salad Snax}	=> {Thousand Island Salad Snax}	0.000	0.802	23030.140	69
4	{Extra Virgin Olive Oil Spray}	=> {All-Purpose Unbleached	0.000	0.784	8055.814	40
5	{2nd Foods Turkey Meat}	=> {2nd Foods Chicken & Gravy}	0.000	0.777	4887.886	202
6	{Apple Strawberry Banana Squeezable Fruit}	=> {Graduates Grabbers Fruit & Yogurt Strawberry Banana}	0.000	0.767	9908.550	33
7	{Chocolate Love Bar}	=> {Ultra-Purified Water}	0.000	0.752	1624.656	76
8	{Ocean Whitefish}	=> {Premium Classic Chicken	0.000	0.750	32148.740	33
9	{Chocolate Love Bar}	=> {Lite Energy Drink}	0.000	0.743	5953.323	75
10	{Ancient Grains Apricot Blended Low-Fat Greek Yogurt}	=> {with Mixed Berry Low-Fat Greek Yogurt}	0.000	0.742	29088.160	46

**Figure 4: Top 10 rules by Confidence**

**Step 4:** Next, we increase the support and decrease confidence to get rules of some more frequent items but with less confidence.

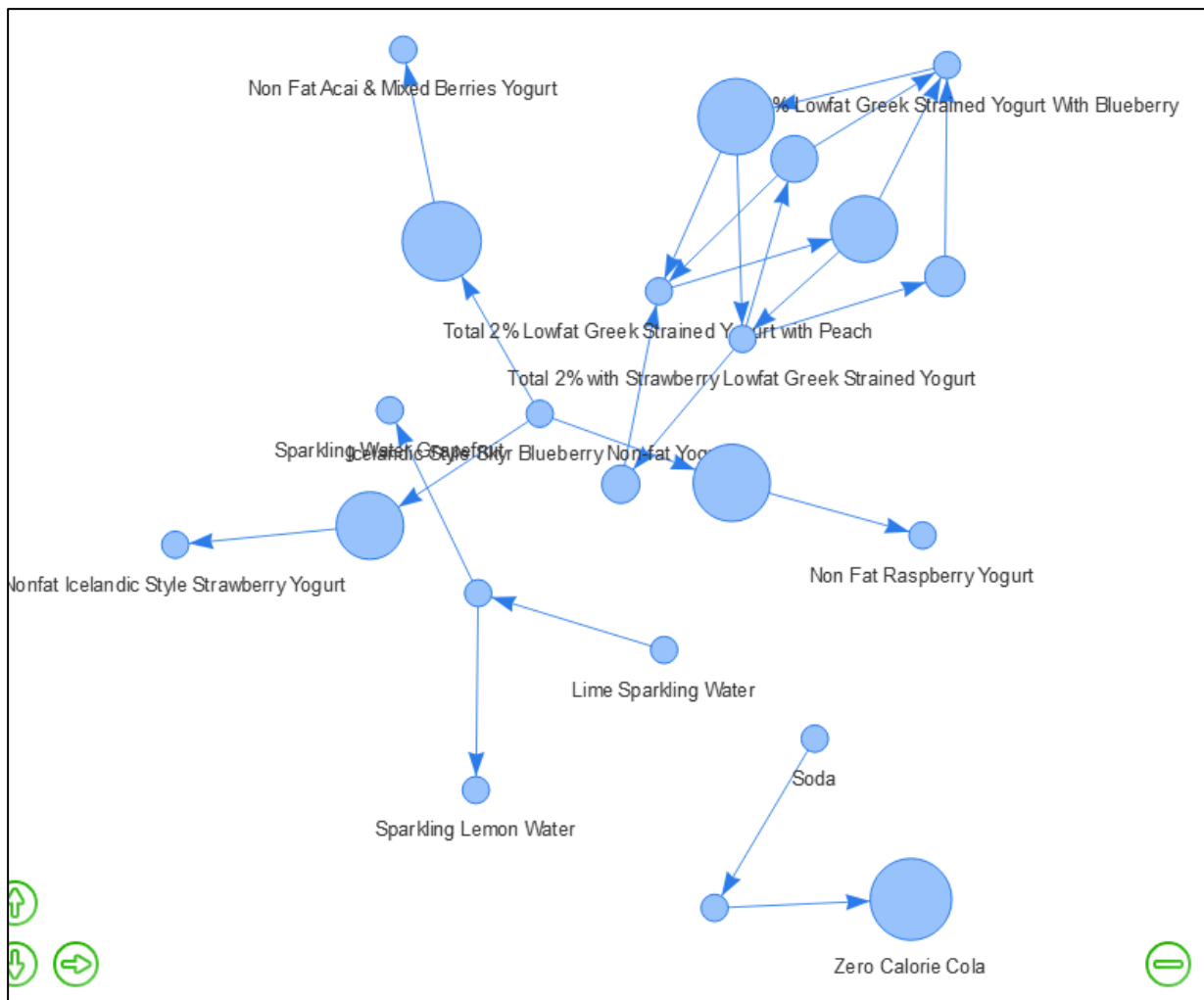
**support = 0.001, confidence = 0.4**

```
> summary(quality(rules2))
```

support	confidence	lift	count
Min. :0.001006	Min. :0.4001	Min. : 2.779	Min. : 3235
1st Qu.:0.001161	1st Qu.:0.4167	1st Qu.: 3.028	1st Qu.: 3733
Median :0.001280	Median :0.4411	Median : 3.683	Median : 4116
Mean :0.001483	Mean :0.4436	Mean :19.220	Mean : 4769
3rd Qu.:0.001541	3rd Qu.:0.4536	3rd Qu.:33.202	3rd Qu.: 4955
Max. :0.003549	Max. :0.5942	Max. :75.648	Max. :11409

S. No.	RULES		support	confidence	lift	count
1	{Non Fat Acai & Mixed Berries Yogurt}	=> {Icelandic Style Skyr Blueberry Non-fat Yogurt}	0.001	0.453	75.648	3925
2	{Non Fat Raspberry Yogurt}	=> {Icelandic Style Skyr	0.002	0.441	73.641	7224
3	{Total 2% Lowfat Greek Strained Yogurt with Peach,	=> {Total 2% Lowfat Greek Strained Yogurt With Blueberry}	0.001	0.466	72.142	3733
4	Total 2% with Strawberry Lowfat Greek Strained Yogurt}	=> {Icelandic Style Skyr Blueberry Non-fat Yogurt}	0.001	0.428	71.378	4559
5	{Nonfat Icelandic Style Strawberry Yogurt}	=> {Total 2% Lowfat Greek Strained Yogurt with Peach}	0.001	0.400	64.605	3733
6	{Total 2% Lowfat Greek Strained Yogurt With Blueberry,	=> {Total 2% with Strawberry Lowfat Greek Strained Yogurt}	0.001	0.594	63.908	3733
7	Total 2% with Strawberry Lowfat Greek Strained Yogurt}	=> {Total 2% with Strawberry Lowfat Greek Strained Yogurt}	0.003	0.450	48.343	9331
8	{Total 2% Lowfat Greek Strained	=> {Total 2% with Strawberry	0.002	0.403	43.293	8014
9	Total 2% Lowfat Greek Strained	=> {Soda}	0.0012349	0.4638934	41.66855	3970
10	{Total 2% Lowfat Greek Strained Yogurt With Blueberry}	=> {Lime Sparkling Water}	0.0013792	0.4807025	33.20152	4434

**Figure 5: Top 10 rules by Lift**



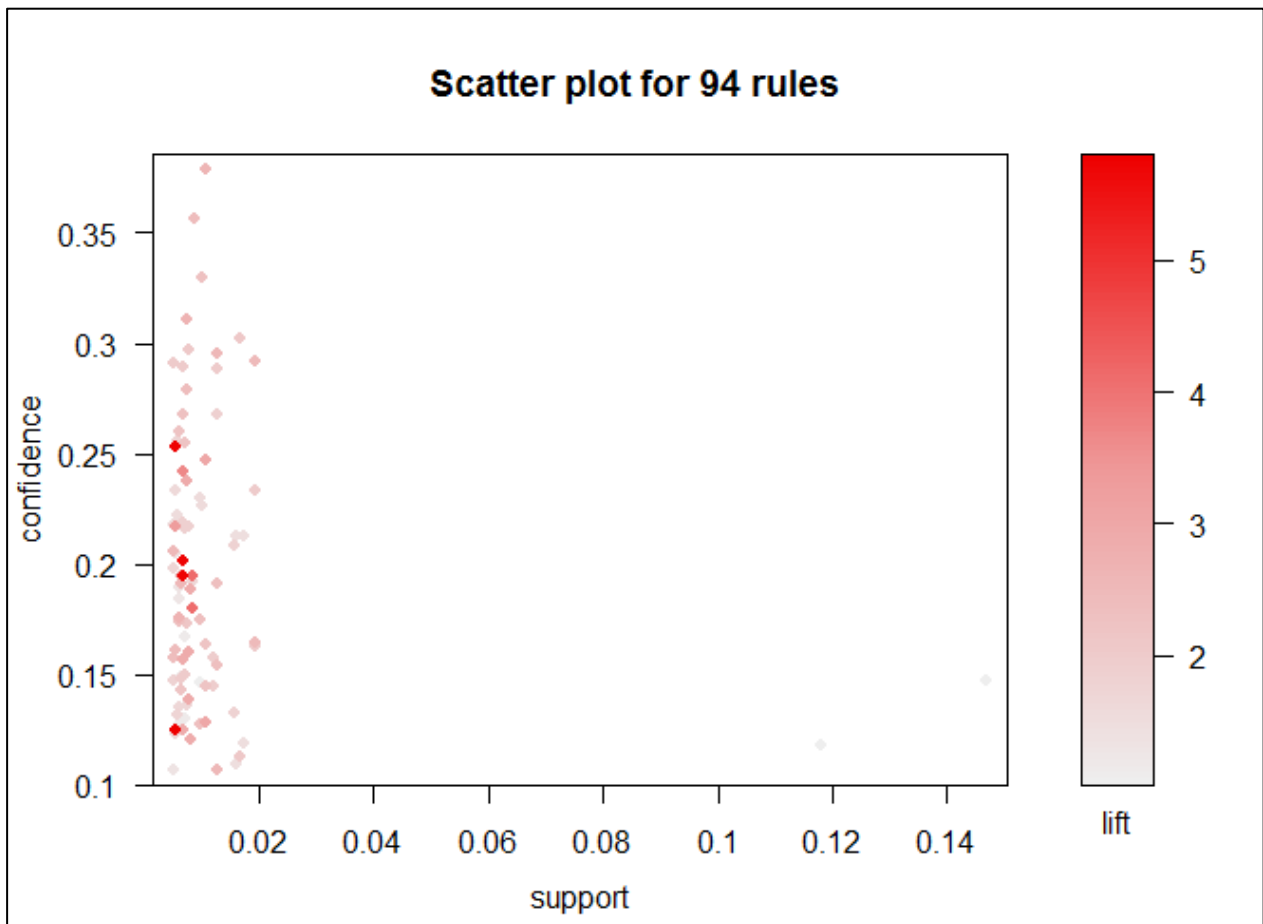
**Figure 6: Network visualization of rules**

**Step 5:** Finally, lets further increase support and decrease confidence.

**support = 0.005, confidence = 0.1**

```
> summary(quality(rules3))
```

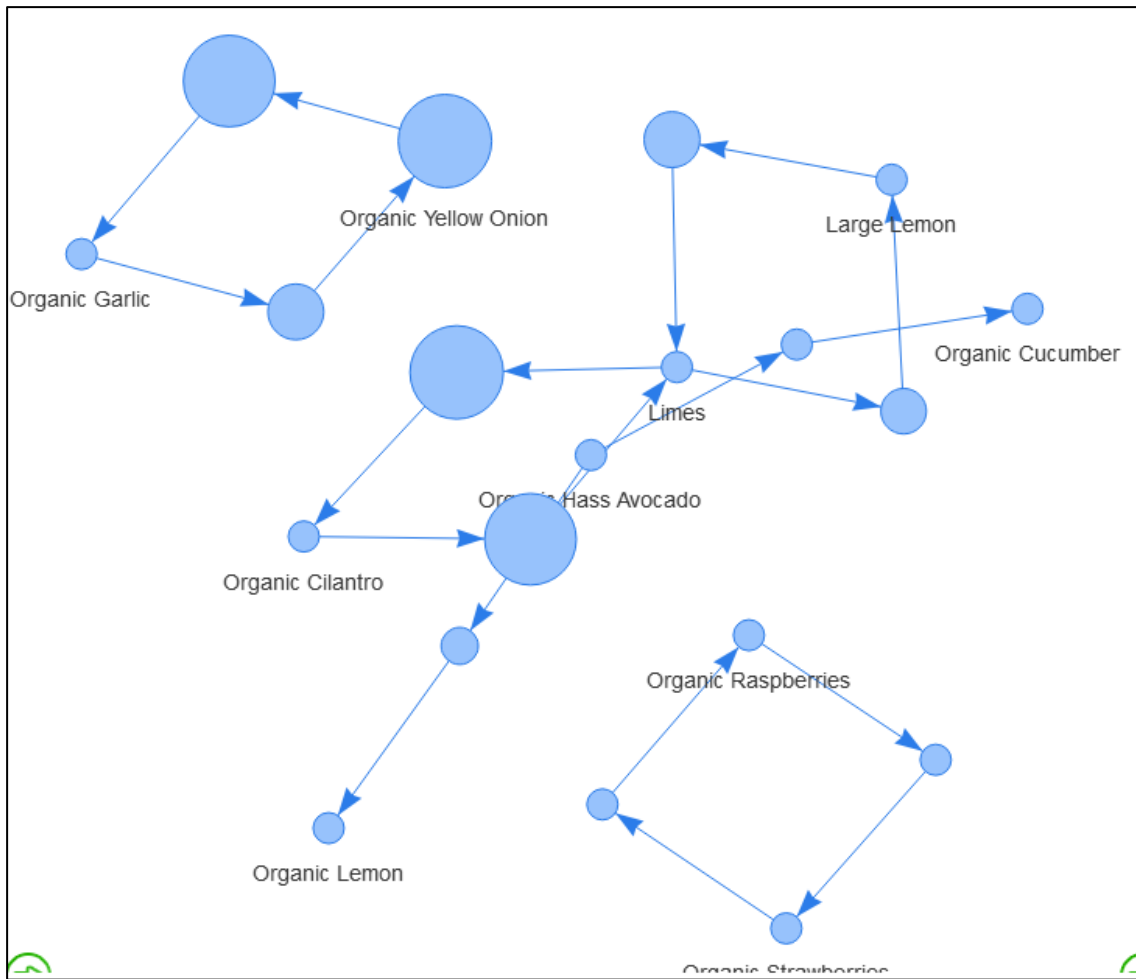
support		confidence		lift		count	
Min.	:0.005066	Min.	:0.1067	Min.	:0.9945	Min.	: 16286
1st Qu.	:0.006016	1st Qu.	:0.1463	1st Qu.	:1.6284	1st Qu.	: 19340
Median	:0.007319	Median	:0.1862	Median	:2.0044	Median	: 23530
Mean	:0.011497	Mean	:0.1933	Mean	:2.2069	Mean	: 36962
3rd Qu.	:0.010782	3rd Qu.	:0.2286	3rd Qu.	:2.5031	3rd Qu.	: 34662
Max.	:0.146993	Max.	:0.3787	Max.	:5.7758	Max.	:472565



S. No.	RULES		support	confidence	lift	count
1	{Organic Cilantro}	=> {Limes}	0.005	0.253	5.776	17565
2	{Limes}	=> {Organic Cilantro}	0.005	0.125	5.776	17565
3	{Organic Garlic}	=> {Organic Yellow Onion}	0.007	0.201	5.699	22073
4	{Organic Yellow Onion}	=> {Organic Garlic}	0.007	0.195	5.699	22073
5	{Limes}	=> {Large Lemon}	0.009	0.195	4.104	27403
6	{Large Lemon}	=> {Limes}	0.009	0.180	4.104	27403
7	{Organic Lemon}	=> {Organic Hass Avocado}	0.007	0.242	3.645	21246
8	{Organic Cucumber}	=> {Organic Hass Avocado}	0.005	0.217	3.268	17456
9	{Organic Raspberries}	=> {Organic Strawberries}	0.0105332	0.2470724	3.000973	33863
10	{Organic Strawberries}	=> {Organic Raspberries}	0.0105332	0.1279379	3.000973	33863

**Figure 7: Top 10 rules by Lift**





S. No.	RULES		support	confidence	lift	count
1	{Organic Fuji Apple}	=> {Banana}	0.011	0.379	2.576	33943
2	{Honeycrisp Apple}	=> {Banana}	0.009	0.356	2.423	28408
3	{Cucumber Kirby}	=> {Banana}	0.010	0.330	2.244	32097
4	{Organic Large Extra Fancy Fuji Apple}	=> {Bag of Organic Bananas}	0.007	0.311	2.634	23364
5	{Organic Avocado}	=> {Banana}	0.017	0.302	2.054	53395
6	{Seedless Red Grapes}	=> {Banana}	0.008	0.297	2.023	24594
7	{Organic Raspberries}	=> {Bag of Organic Bananas}	0.013	0.296	2.504	40503
8	{Organic Hass Avocado}	=> {Bag of Organic Bananas}	0.019	0.292	2.473	62341
9	{Blueberries}	=> {Banana}	0.0050658	0.2911021	1.980377	16286
10	{Yellow Onions}	=> {Banana}	0.0065844	0.2893108	1.96819	21168

**Figure 8: Top 10 rules by Confidence**

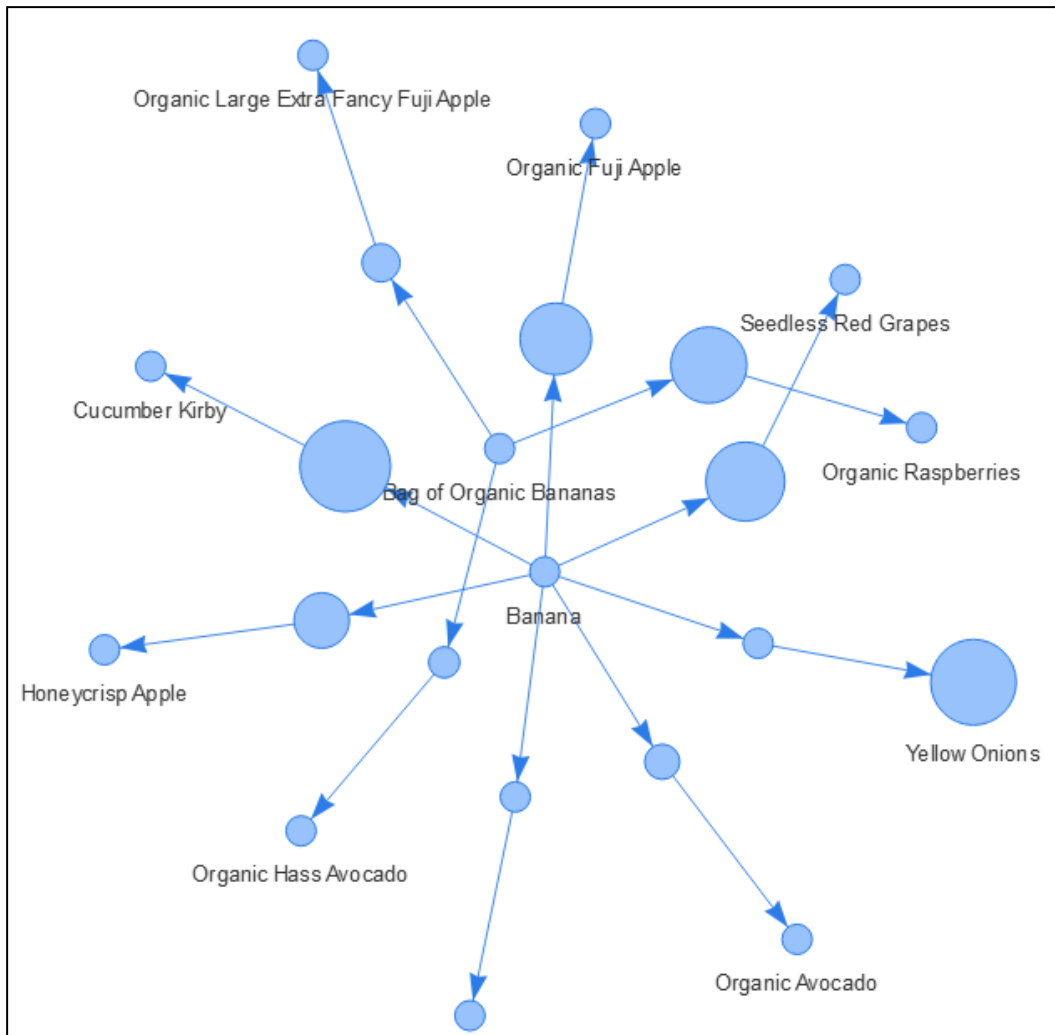


Figure 9: Network Visualization of top 10 rules by confidence

## Conclusions

- Bananas and Strawberries stand first in all charts. People simply love these!
- Fresh Fruits and vegetables are the best sellers, as also seen by exploratory data analysis.
- Some rules, though their support is quite less, have very high Lift values. Since this is such a huge dataset, even a support of 0.0002 can prove to be helpful in case of Association analysis.
- The listed rules can be used for Recommendation feature in Instacart app. For eg. Someone ordering *Moisturizing face wash* can be recommended to also look at *moisturizing non-drying face wash*; as depicted by a huge Lift score.

# Data Modelling

## CHOICE OF ALGORITHM

Our Choice of algorithm is LSTM due to huge training data with limitations of High processing systems. Our main objective was to implement the algorithm, which is easy to train and should be fast enough and give us close by accuracy as any of the other algorithms (XGBOOST, RandomForest).

## WHAT IS LIGHT GBM?

Light GBM is a gradient boosting framework that uses tree based learning algorithm

### **How it differs from another tree based algorithm?**

Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while another algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

### **Why Light GBM is gaining extreme popularity?**

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as 'Light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

### **What about its implementation?**

Implementation of Light GBM is easy, the only complicated thing is parameter tuning. Light GBM covers more than 100 parameters.

## Control Parameters

- **max\_depth**

It describes the maximum depth of tree. This parameter is used to handle model overfitting. Any time you feel that your model is overfitted, my first advice will be to lower max\_depth.

- **min\_data\_in\_leaf**

It is the minimum number of the records a leaf may have. The default value is 20, optimum value. It is also used to deal over fitting

- **feature\_fraction**

Used when your boosting (discussed later) is random forest. 0.8 feature fraction means LightGBM will select 80% of parameters randomly in each iteration for building trees.

- **bagging\_fraction**

Specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.

- **early\_stopping\_round**

This parameter can help you speed up your analysis. Model will stop training if one metric of one validation data doesn't improve in last early\_stopping\_round rounds. This will reduce excessive iterations.

- **Lambda**

lambda specifies regularization. Typical value ranges from 0 to 1.

- **min\_gain\_to\_split**

This parameter will describe the minimum gain to make a split. It can have used to control number of useful splits in tree.

- **max\_cat\_group**

When the number of category is large, finding the split point on it is easily over-fitting. So LightGBM merges them into 'max\_cat\_group' groups, and finds the split points on the group boundaries, default:64

## **TUNING PARAMETERS**

Following set of practices can be used to improve our model efficiency.

### **1. num\_leaves**

This is the main parameter to control the complexity of the tree model. Ideally, the value of num\_leaves should be less than or equal to  $2^{(\text{max\_depth})}$ . Value more than this will result in overfitting.

### **2. min\_data\_in\_leaf**

Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting. In practice, setting it to hundreds or thousands is enough for a large dataset.

### **3. max\_depth**

You also can use max\_depth to limit the tree depth explicitly.

## **For faster speed**

- Use bagging by setting bagging\_fraction and bagging\_freq
- Use feature sub-sampling by setting feature\_fraction
- Use small max\_bin
- Use save\_binary to speed up data loading in future learning
- Use parallel learning, refer to parallel learning guide.

## **For better accuracy**

- Use large max\_bin (may be slower)
- Use small learning\_rate with large num\_iterations
- Use large num\_leaves (may cause over-fitting)
- Use bigger training data
- Try dart
- Try to use categorical feature directly

## **To deal with over-fitting**

- Use small max\_bin
- Use small num\_leaves
- Use min\_data\_in\_leaf and min\_sum\_hessian\_in\_leaf
- Use bagging by set bagging\_fraction and bagging\_freq
- Use feature sub-sampling by set feature\_fraction
- Use bigger training data
- Try lambda\_l1, lambda\_l2 and min\_gain\_to\_split to regularization
- Try max\_depth to avoid growing deep tree

## FEATURE EXTRACTION

In machine learning, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

We developed a set of features by combining existing variables, to be fed into the Light GBM model. Here is a brief description of the same:

Dataset	Feature Name	Description	Group
Prior	Orders	No. of orders for each product	Product features
Prior	reorders	Count of reorders for each product	Product features
Prior	reorder_rate	$\text{reorders/orders}$	Product features
orders	average_days_between_orders	Average of days between orders for each user	User features
orders	nb_orders	Total no. of orders for each user	User features
Prior	total_items	Total items ordered by a user	User features
Prior	all_products	Set of all products ordered by a user	User features
users	total_distinct_items	Count of all products ordered	User features
users	average_basket	$\text{total\_items/nb\_orders}$	User features
prior	user_product	$\text{priors.product\_id} + \text{priors.user\_id} * 100000$	Priors features
userXproduct	order_number,order_id	Combining User and Product Features	User and Product Combined
userXproduct	add_to_cart_order		User and Product Combined
userXproduct	nb_orders		User and Product Combined
userXproduct	last_order_id		User and Product Combined
userXproduct	sum_pos_in_cart		User and Product Combined

Figure 3: Intermediate features based on given data

Feature Name	Description	Group
user_id	$\text{df.order\_id.map}(\text{orders.user\_id})$	User related features
user_total_orders	$\text{df.user\_id.map}(\text{users.nb\_orders})$	User related features
user_total_items	$\text{df.user\_id.map}(\text{users.total\_items})$	User related features
total_distinct_items	$\text{df.user\_id.map}(\text{users.total\_distinct\_items})$	User related features
user_average_days_between_orders	$\text{df.user\_id.map}(\text{users.average\_days\_between\_ord})$	User related features
user_average_basket	$\text{df.user\_id.map}(\text{users.average\_basket})$	User related features
order_hour_of_day	$\text{df.order\_id.map}(\text{orders.order\_hour\_of\_day})$	Order related features
days_since_prior_order	$\text{df.order\_id.map}(\text{orders.days\_since\_prior\_order})$	Order related features
days_since_ratio	$\text{df.days\_since\_prior\_order} / \text{df.user\_average\_days}$	Order related features
aisle_id	$\text{df.product\_id.map}(\text{products.aisle\_id})$	Product related features
department_id	$\text{df.product\_id.map}(\text{products.department\_id})$	Product related features
product_orders	$\text{df.product\_id.map}(\text{products.orders})$	Product related features
product_reorders	$\text{df.product\_id.map}(\text{products.reorders})$	Product related features
product_reorder_rate	$\text{df.product\_id.map}(\text{products.reorder\_rate})$	Product related features
UP_orders	$\text{userXproduct.nb\_orders}$	User and Product related features
UP_orders_ratio	$\text{df.UP\_orders} / \text{df.user\_total\_orders}$	User and Product related features
UP_average_pos_in_cart	$\text{userXproduct.sum\_pos\_in\_cart} / \text{df.UP\_orders}$	User and Product related features
UP_reorder_rate	$\text{df.UP\_orders} / \text{df.user\_total\_orders}$	User and Product related features
UP_orders_since_last	$\text{df.user\_total\_orders} - \text{df.UP\_last\_order\_id.map}(\text{or})$	User and Product related features
UP_delta_hour_vs_last	$\text{abs}(\text{df.order\_hour\_of\_day} - \text{df.UP\_last\_order\_id.map}(\text{orders.order\_hour\_of\_d ay})).\text{map}(\text{lambda x: min}(x, 24-x))$	User and Product related features

Figure 4: Final set of features given as input to LightGBM algorithm

## MODEL VALIDATION (MEAN F1 SCORE)

The traditional F-measure or balanced F-score (**F<sub>1</sub> score**) is the **harmonic mean** of precision and recall:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy.






It considers both the precision  $p$  and the recall  $r$  of the test to compute the score:

- $p$  is the number of correct positive results divided by the number of all positive results returned by the classifier, and
- $r$  is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

## Results

Result with maximum F1 score of **0.4001449**

<div><div>In the money</div><div>Gold</div><div>Silver</div><div>Bronze</div></div>								
#	Δpub	Team Name	Kernel	Team Members	Score ?	Entries	Last	
1	—	胡萝卜			0.4091449	62	7mo	
2	—	===== KEEP OUT 🖐 =====			0.4082039	138	7mo	
3	—	sjv			0.4081041	76	7mo	
4	—	Let's overFit			0.4074450	115	7mo	
5	▲ 9	Stanislav Semenov			0.4073023	35	7mo	

We scored **0.3762561**

6 submissions for <a href="#">vrana95</a>				Sort by	Most recent	▼
All	Successful	Selected				
Submission and Description			Private Score	Public Score	Use for Final Score	
<a href="#">sub.csv</a> 3 hours ago by <a href="#">vrana95</a> <a href="#">add submission details</a>			0.3765282	0.3762561	<input type="checkbox"/>	