**Milestone Report2 for Capstone 2 project**

**Background/Context:**

Online grocers like Instacart provide a certain set of benefits to their end customers. For customers that are willing to pay a small fee, online grocers are able to shop for them based on their grocery requirements and deliver the groceries to them within a reasonable timeframe.  End customers save the time spent doing groceries themselves.

Online grocers offer a software platform (both on mobile and desktops) that customers can use to browse various products/grocery items offered by different grocery stores that the online grocer has partnered with. Once the customer submits the order via the software platform, online grocers take care of that order via shoppers and drivers that they have employed or contracted with. For the service they provide, they take a fee from the customers and their partner grocery stores.

End clients that I have in mind for capstone 2 project are online grocers who value recommending relevant products to their customers. End customers would benefit by being aware of products potentially relevant to them and as a result, online grocers would benefit by potential upside to their revenue if the recommended products are purchased by customers.

**Problem to be solved and beneficiary of solution:**

Specific problem I will be addressing for online grocers is that of making product recommendations to customers based on their similarity to other customers. The primary beneficiary of solution to this problem would be marketing/advertising managers at online grocers who can potentially use this to increase revenue from targeted customers.  From customer’s perspective, they would have a better experience with the brand of online grocer since they get recommended products relevant to them vs being bombarded with products not relevant to them.

**Methodology used:**

Following steps were performed for this project:

1. Data gathering
2. Data cleaning/wrangling
3. Data processing
4. Data exploration
5. Features selection and Clustering
6. Product recommendations to users

Following sections describe the steps that were performed and results obtained for this project.

**Data gathering:**

I used data provided at Kaggle:

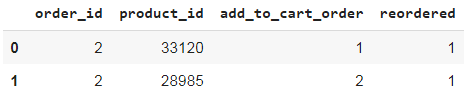
<https://www.kaggle.com/c/instacart-market-basket-analysis/data>

The data was comprised of 6 files/dataframes:

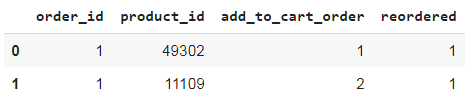
1. df\_orders (about 3M rows, contains information such as orders, users, day of week order was made, hour of day order was made and days since prior order)



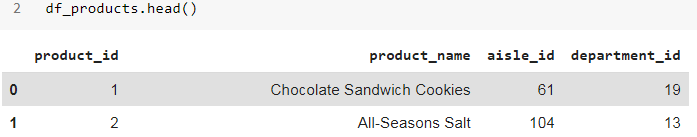
1. df\_order\_products\_prior (about 30M rows, contains detailed information regarding orders such as products associated with orders, add to cart order for products and whether product was reordered or not)



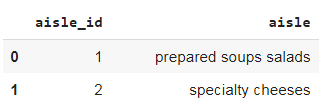
1. df\_order\_products\_train (about 1M rows, contains information regarding orders to be used for training purposes)



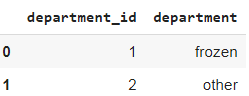
1. df\_products (about 50K rows, contains information regarding products being offered)



1. df\_aisles (about 100 rows, contains information regarding aisles that products belong to)



1. df\_departments (about 20 rows, contains information regarding departments that products belong to)



Instacart had provided the above dataset at Kaggle for the purpose of predicting which products would be reordered by each user in their next order given their past purchase history with Instacart. For this project, instead of using the dataset to identify solution to problem Instacart posed, what I did was take the dataset and used it to recommend products to users based on their similarity to other users as identified by certain behavioral traits. For the purpose of this project, I made use of the following dataframes:

1. df\_orders
2. df\_order\_products\_prior
3. df\_products
4. df\_aisles
5. df\_departments

df\_order\_products\_train was not necessary since there was no training involved for this project . Summarizing the data contained in the dataframes:

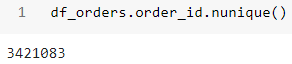
1. There were about 3.4 million orders
2. There were about 200K users
3. There were about 50K products
4. There were 134 aisles
5. There were 21 departments

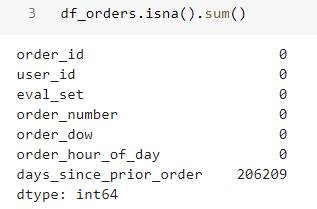
**Data wrangling/Cleaning:**

Since the data was obtained from Kaggle, data was relatively clean:

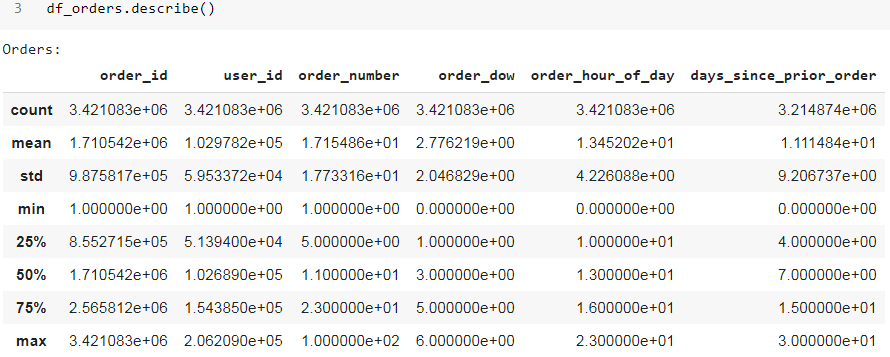
1. Treatment of null/NaN values

Of the 5 dataframes used, only df\_orders had null or NaN values. It had about 200K NaN values for one of the columns out of a total of about 3.4M rows. The rows corresponding to NaN values were dropped with minimal impact (since they were less than 10% of total number of rows)



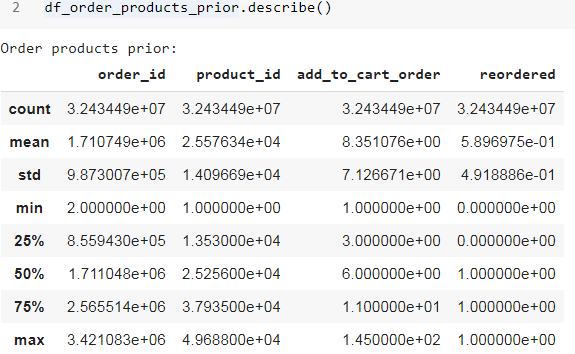


1. Treatment of outliers or bad/incorrect data points:



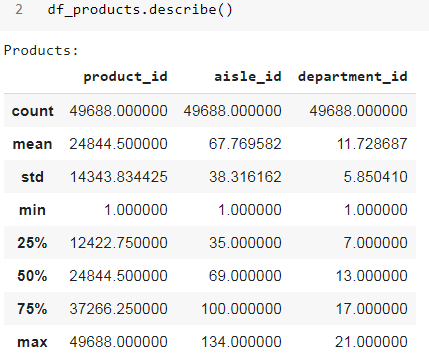
From results above, min and max values for all the columns of df\_orders seemed reasonable. There were no outliers or bad/incorrect data points that need to be treated.

1. order\_id : goes from 1 to about 3.4M
2. user id : goes from 1 to about 200K
3. order\_number : goes from 1 to about 100 which seems reasonable (a user can have made upto 100 orders to instacart in his/her history)
4. order\_dow : goes from 0 to 6 which is resonable ( day of the week for order can have any of the 7 possible values)
5. order\_hour\_of\_day : goes from 0 to 23 which is reasonable (hour of the day for order can have any of the 24 possible values)
6. days\_since\_prior\_order : goes from 0 to 30 which is reasonable (days since user last made the order can have any of the 31 possible values)



From results above, min and max values for all columns of df\_order\_products\_prior seemed reasonable. There were no outliers or bad/incorrect data points that need to be treated:

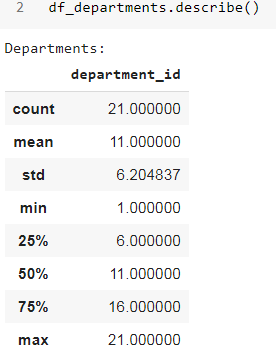
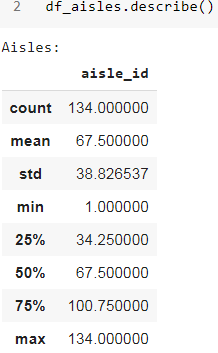
1. product\_id : goes from 1 to about 49K which is reasonable (it is possible to have product offerings of upto 50K given all the grocery stores tha Instacart has partnered with)
2. add\_to\_cart\_order : goes from 1 to 145 which is reasonable (it is possible for a user to have upto 145 products in a given order)
3. reordered : goes from 0 to 1 which is reasonable ( this indicates whether a given product item in an order was reordered or not)



From results above, min and max values for all columns of df\_products seemed reasonable. There were no outliers or bad/incorrect data points that need to be treated:

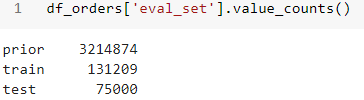
1. aisle\_id : goes from 1 to 134 which is reasonable (it is possible for a grocery store to have upto 134 aisles)
2. department\_id: goes from 1 to 21 which is reasonable ( it is possible for a grocery store to have upto 21 departments)

Simiarly for dataframes df\_aisles and df\_departments below, all columns were reasonable.

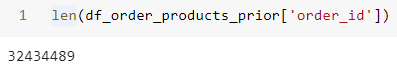


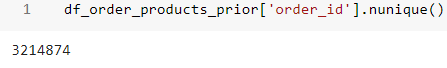
**Data processing**

The dataframe df\_orders has about 3.4M rows, all of which correspond to unique orders. Of these, about 3.2M rows correspond to orders when eval\_set = prior:



Dataframe df\_order\_products\_prior has about 32.4M rows. Each row in df\_order\_products\_prior corresponds to one product item in an order for a user. Hence there are lot more rows in df\_order\_products\_prior than in df\_orders. However, number of unique orders in df\_order\_products\_prior is the same as that in df\_orders (for the case eval\_set = prior) as is seen in results below:

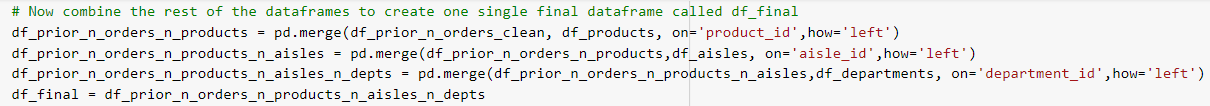


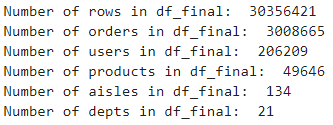


What this means is that when we combine the two dataframes as follows, orders corresponding to cases when eval\_set = train or test in df\_orders are dropped since the only common orders between the two dataframes are for the case when eval\_set = prior :



In order to understand data better, it is useful to combine the 5 dataframes into a single dataframe as follows which can then be explored for insights into the data :

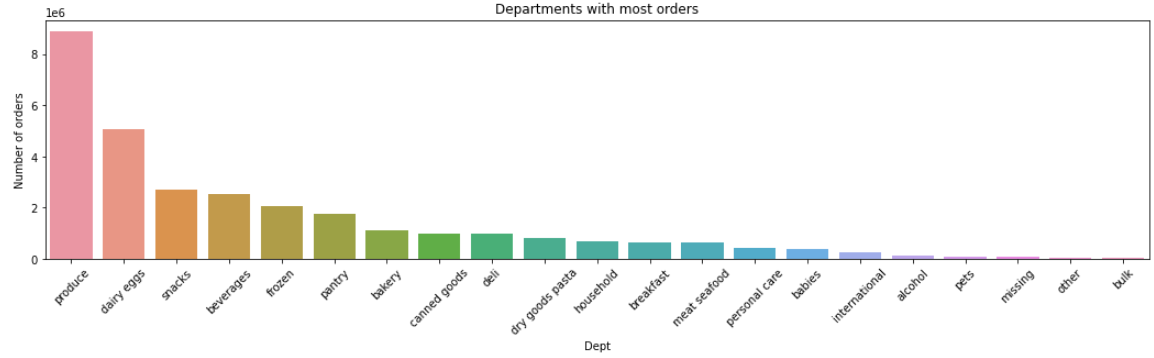




**Data exploration**

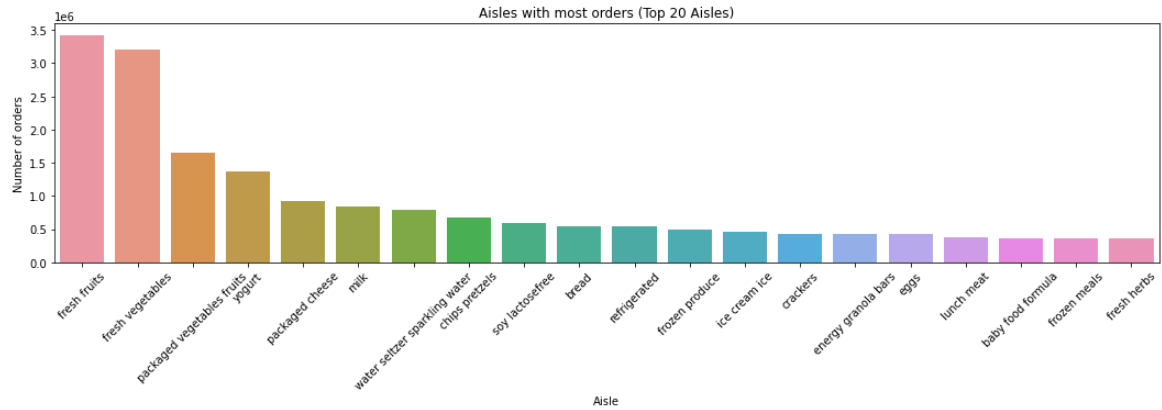
Ipython notebook covers all the insights uncovered for this project. This report goes into the salient ones only:

1. Produce, Dairy eggs and Snacks were top 3 departments in terms of demand

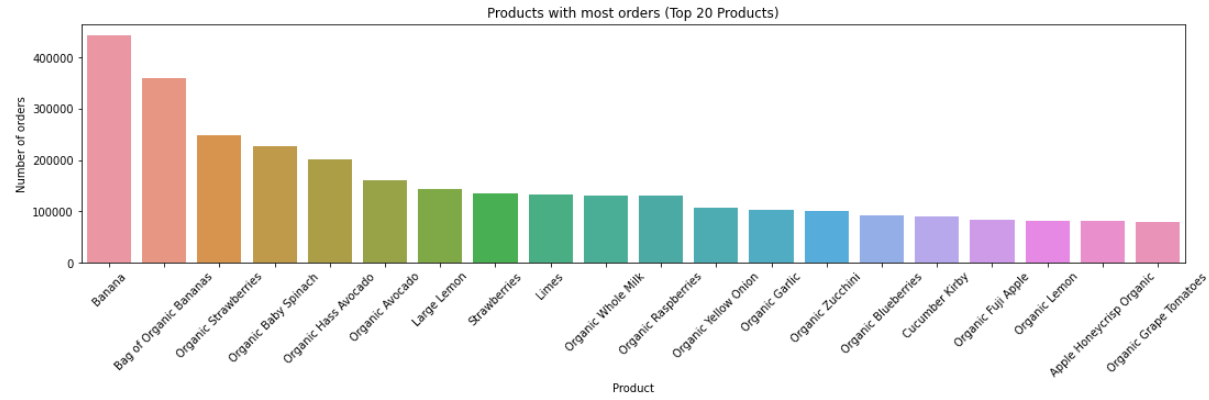


1. Fresh fruits, Fresh vegetables and Packaged Vegetables Fruits were top 3

aisles in terms of demand



1. Bananas, Bag of Organic Bananas and Organic Strawberries were top 3 products in terms of demand



1. Days 0 and 1 (presumably Saturday and Sunday) get more orders than days 2,3,4,5 and 6



1. Orders start to pick up around 6am in the morning, hit the peak around 10am, stays around the peak till about 4pm after which it starts to go down.

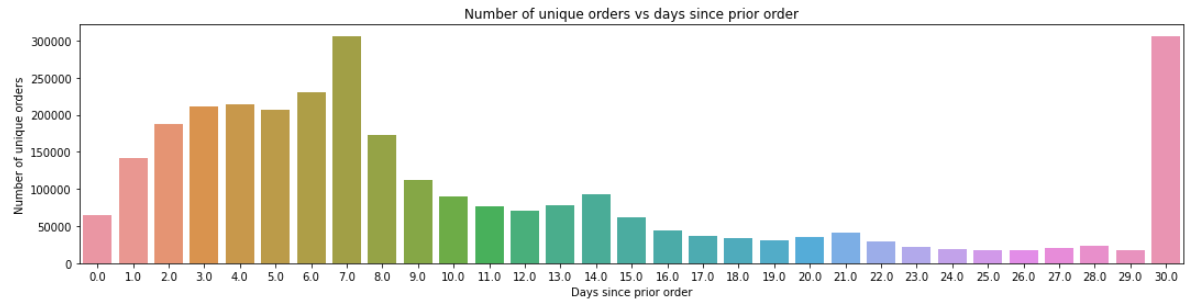


1. There is one large segment of customers who order every week and there is another large segment of customers who order every month as evident by 2 major peaks at 7 and 30 for days since prior order.

There is also large segment of customers who order every 1, 2,3,4,5 and 6 days as well.

There is moderate segment of customers who order every 2 weeks (day 14) and every 3 weeks (day 21) as well.

There is also a sizable segment of customers who order on the same day (day 0)



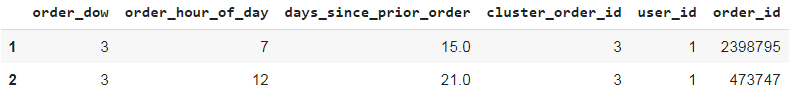
**Features selection and Clustering**

Users can differ from each other in a number of ways:

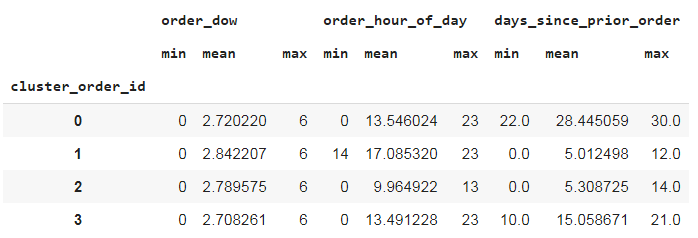
1. Product preferences
2. Behavioral attributes

For this project, three features (order\_dow, order\_hour\_of\_day, days\_since\_prior\_order) were selected to be used as features for clustering the orders since they give insight into the behavioral attributes of the users (i.e. day of the week order was placed, hour of the day order was placed, days since prior order by the user)

K-Means algorithm was used to cluster the orders using above 3 features after selecting elbow point for K-Means algorithm (which came out to be 4). The result would look something as follows:



The table below attempts to show how the 4 clusters differ:



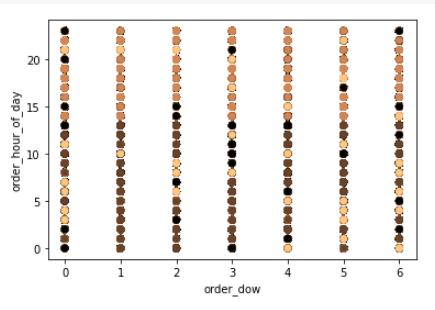
From table above, the clusters differ from each other in the following ways:

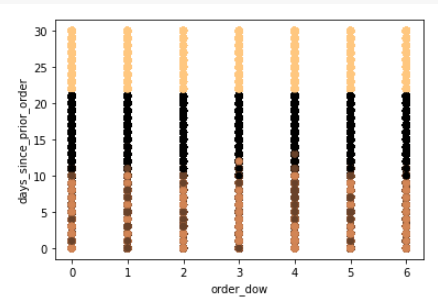
i) **All clusters** have **similar mean** values for **order\_dow**

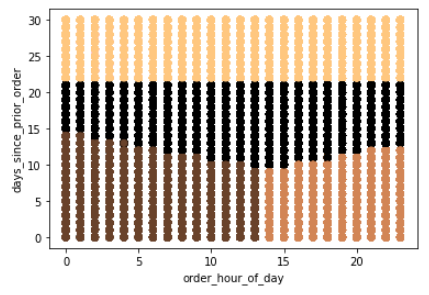
ii) Two clusters have **similar mean** values for **order\_hour\_of\_day** (around 13.5) but differ in **different mean** values for **days\_since\_prior\_order**(one around 28, other around 15)

iii) Other two clusters have **similar mean** values for **days\_since\_prior\_order** (around 5) but differ in **different mean** values for **order\_hour\_of\_day** (one around 17, other around 9.9)

Below are 3 plots of how cluster labels vary for each two of the 3 selected features:







From above 3 plots, we can see that only the plot of days\_since\_prior\_order vs order\_hour\_of\_day does a clear separation of the 4 clusters. Other two plots don’t do as good a separation of the 4 clusters. This implies that the two features days\_since\_prior\_order and order\_hour\_of\_day are the most useful features for the purpose of segmenting orders into 4 different clusters.

Now, so far, the orders have been clustered into 4 different clusters. A given user has multiple orders, hence there are multiple cluster labels for a user. **The dominant/most frequent cluster label for the user was chosen and that dominant cluster label was assigned as the cluster label for the user.**