Curious about the food Americans eat? Ordering food supplies online is a new way of restocking groceries and other essential items. Once order is placed, what happens when you forget few items while adding items to the cart or want to get better suggestions on your items? To deal with such situations, users are provided with suggestions based on their past orders or user preferences.

Instacart, a grocery order and delivery app with over 500 Million plus products and 40000 plus stores serves across U.S & Canada. Instacart opened up a challenge for Kaggle community to use the anonymized data on customer order over time to predict which previously purchased products will be in a user's next order. This was exciting challenge to-do as data set gives immense opportunity for features building and fine tuning the Machine Learning model.

Business Objectives and Constraints:

The objective is to predict/recommend which products will be in a user's next order based on past history. The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, Instacart provided between 4 and 100 of their orders, along with the sequence in which products were placed in the cart. Additionally, if user is new then top products ordered for that given hour will be displayed.

Data:

Data can be broadly divided into 3 parts.

- Prior data : Order history of every user . This data contains nearly 3–100 past orders per user
- Train data : Current order data of every user . This data contains only 1 order per user
- Test data : Future order data of every user . This data will not contain any product information (We need to predict that)

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)

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

deptartments (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

where SET is one of the four following evaluation sets (eval_set in orders):

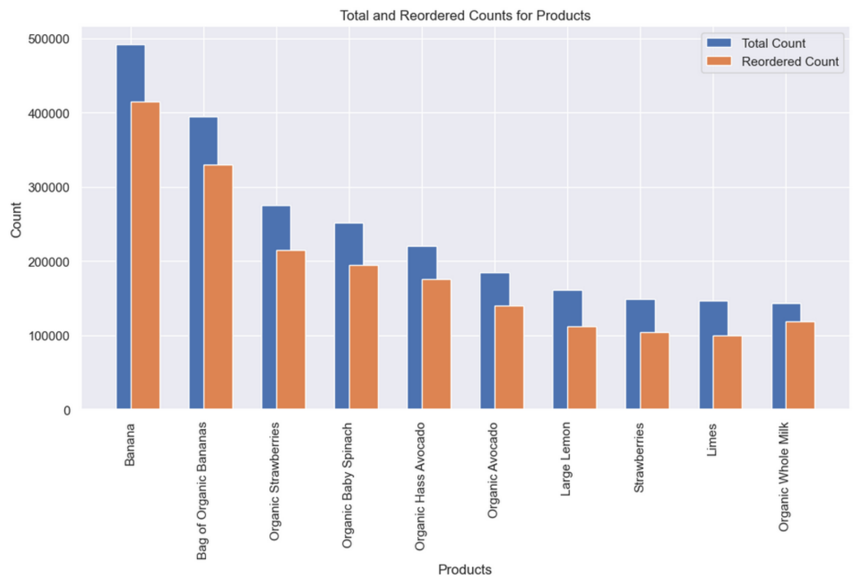
- "prior": orders prior to that users most recent order (~3.2m orders)
- "train": training data supplied to participants (~131k orders)
- "test": test data reserved for machine learning competitions (~75k orders)

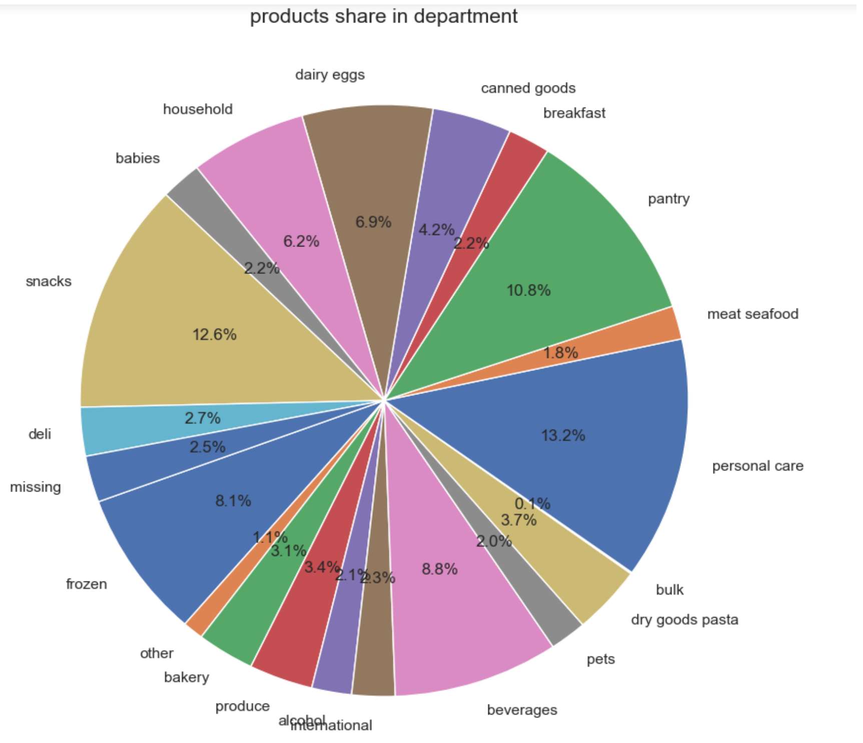
<u>Goal :</u>

For the user to get product recommendations based on his past N orders, we need to observe patterns and generate rules which will give recommendations with high probability. Since we have over 3 Million data points, we need to automate the learning process and using Machine Learning we can achieve probabilistic prediction. Since they are many products (49688 products), rather than Multi-Label Classification, I opted for binary classification problem to check whether product will be reordered or not with probability of an items being ordered in future order. Here I picked general threshold values of P(X) > 0.5 and system will recommend top 5 products to the users.

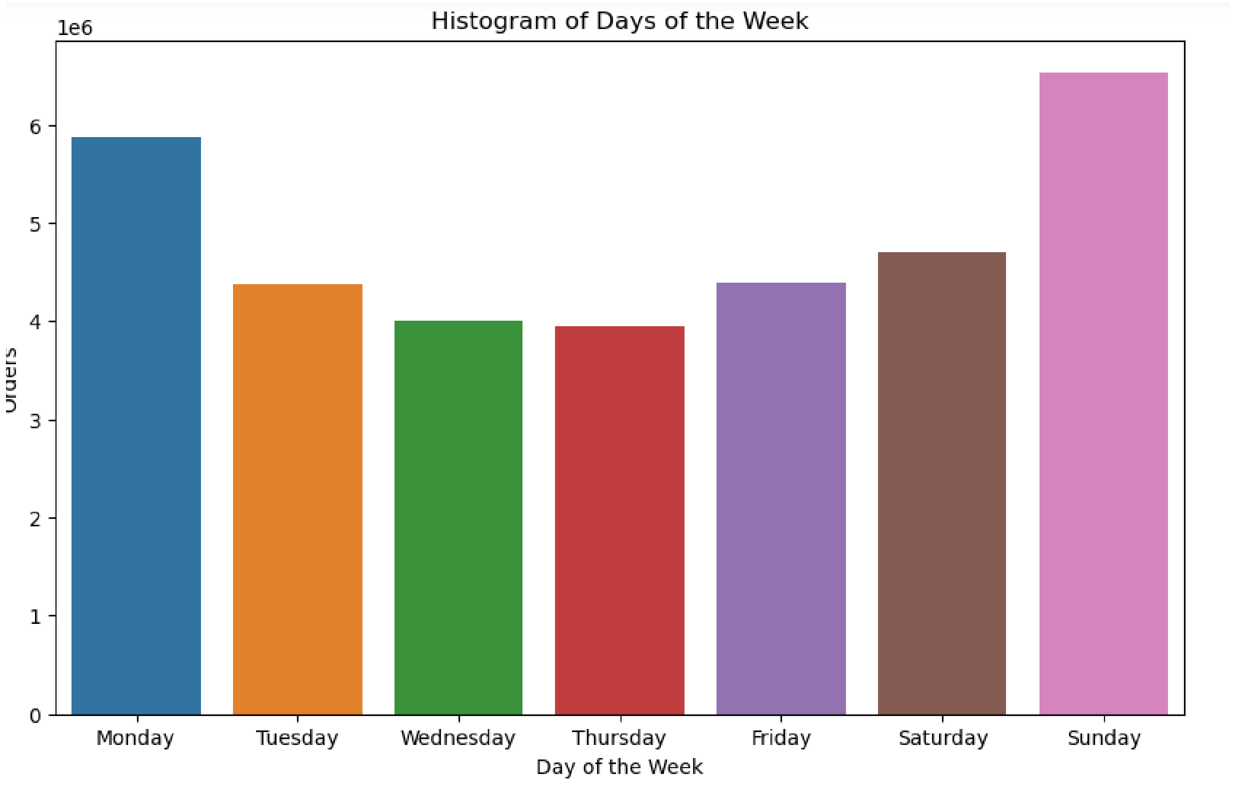
Exploratory Data Analysis:
EDA is important as it helps to answer many questions about the data which further leads to better feature engineering. Given data set is clean with no missing values except for days_since_prior_order column (first order has Nan values). I imputed this value to 0.

The most ordered as well as reordered items as well as product share in the department:

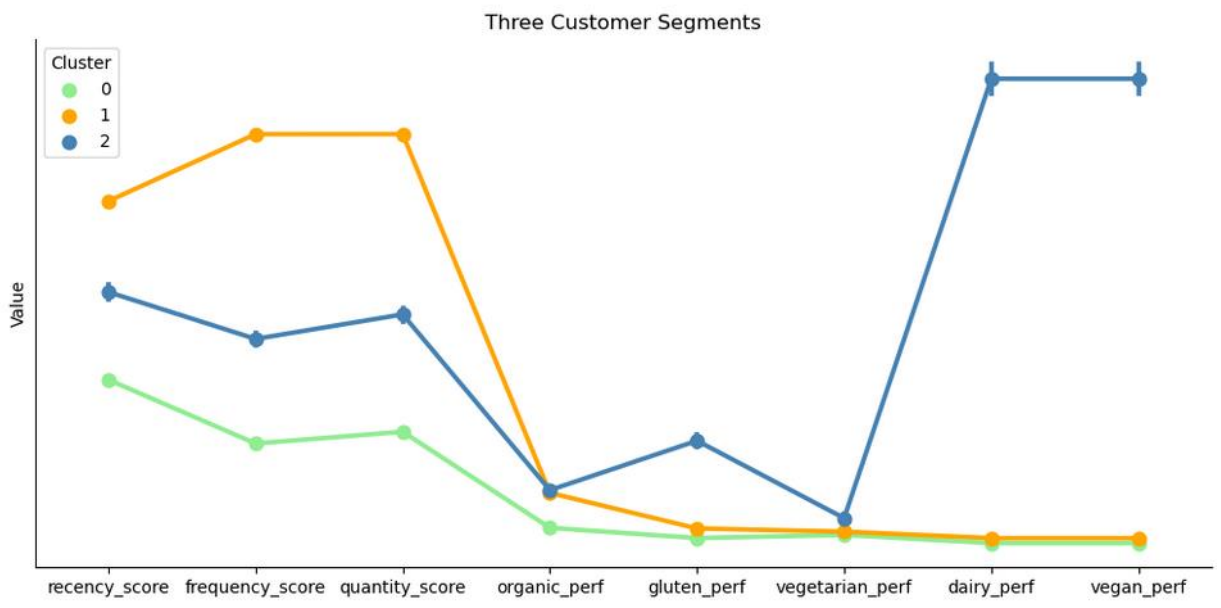




With given days of order (0 as sunday) Customers order the most in the beginning of the week. Good way to start the week with grocery items purchase !!



Users clustered into different segments based on product type, product purchase history and order history. Interestingly users are moving towards vegan and organic purchases.



Feature Engineering

Features were divided largely into these three categories:

Product Features

- What is product reorder rate
- What is product average position in the cart
- What is product reorder rate from aisle to which it belongs
- What is product reorder rate from the department to which it belongs
- For a given time of hour what is the most ordered product

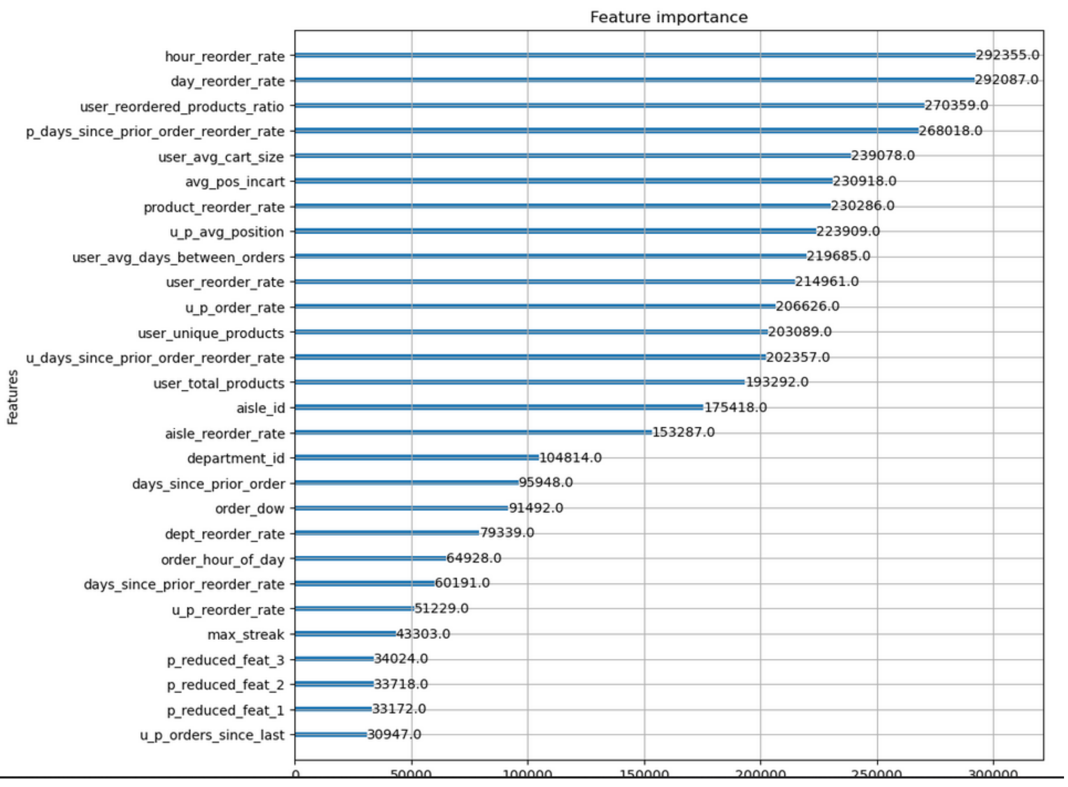
User Features:

- What is user re-order rate
- What are the unique products ordered by given user
- What is the average cart size of user
- What is the average days between orders
- What is the user and product re-order ratio

User and Product interaction Features:

- How frequently user has ordered given product
- How frequently user has re-ordered a given product
- What is Average position of product in the cart

Out of all these features which were the most useful one?



Top 5 Features:

Hour reorder rate: for a given hours of the day what is the rate product is reordered. This is features derived from products and its re-order history for all users.

Day reorder rate : For given day what product was reordered the most? Did you know Mint Crème Filled Dark chocolate is reordered most on weekend ?

User_reordered_products: For a given user and product what is the reorder rate for the product

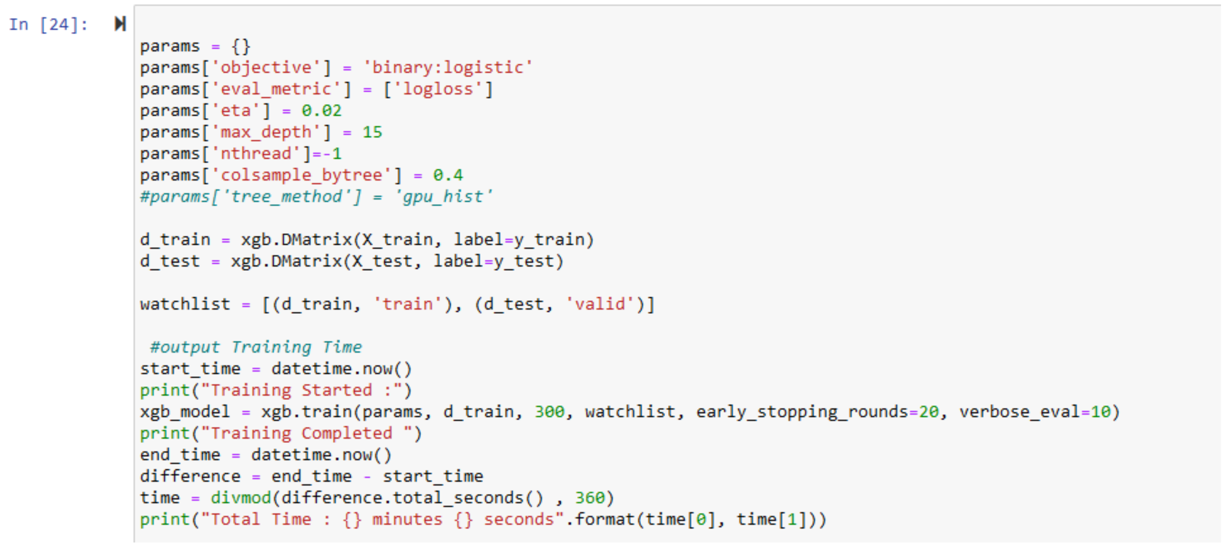
p_days_since_prior_order_reorder_rate : How frequently a product was reordered given that difference between 2 orders containing product, in days

user_avg_cart_size : What is the average users cart size ?

Time to build and train the model so that model can predict product purchases for user's future orders:

Here I tried different classification models like Decision tree, Random Forest classifier and XGBoost Classifier. All algorithm has similar accuracy but in terms of time and CPU efficiency XGBoost won !!

XGBoost Algorithm was built using logloss eval metric , and max_depth of 15 . This model had accuracy of 91 % which is really good.



In [24]:

```
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = ['logloss']
params['eta'] = 0.02
params['max_depth'] = 15
params['nthread']=-1
params['colsample_bytree'] = 0.4
#params['tree_method'] = 'gpu_hist'

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

 #output Training Time
start_time = datetime.now()
print("Training Started :")
xgb_model = xgb.train(params, d_train, 300, watchlist, early_stopping_rounds=20, verbose_eval=10)
print("Training Completed ")
end_time = datetime.now()
difference = end_time - start_time
time = divmod(difference.total_seconds() , 360)
print("Total Time : {} minutes {} seconds".format(time[0], time[1]))
```

```
[230]	train-logloss:0.21123	valid-logloss:0.24441
[240]	train-logloss:0.20999	valid-logloss:0.24408
[250]	train-logloss:0.20878	valid-logloss:0.24378
[260]	train-logloss:0.20763	valid-logloss:0.24358
[270]	train-logloss:0.20658	valid-logloss:0.24342
[280]	train-logloss:0.20557	valid-logloss:0.24328
[290]	train-logloss:0.20456	valid-logloss:0.24312
[299]	train-logloss:0.20359	valid-logloss:0.24292
Training Completed
Total Time : 2.0 minutes 209.223026 seconds
```

```
C:\Users\Pallavi\anaconda3\Lib\site-packages\sklearn\metrics\_classification.
rameter is deprecated and will be removed in 1.5. Instead eps will always hav
e).eps`.
  warnings.warn(
```

```
The test log loss is: 0.24292359567035127
```

Let’s see what products are recommended by the model:

When is tested for a new user, then most ordered product list for given hour is recommended whereas for existing users, prior order history is analyzed before the product recommendation.

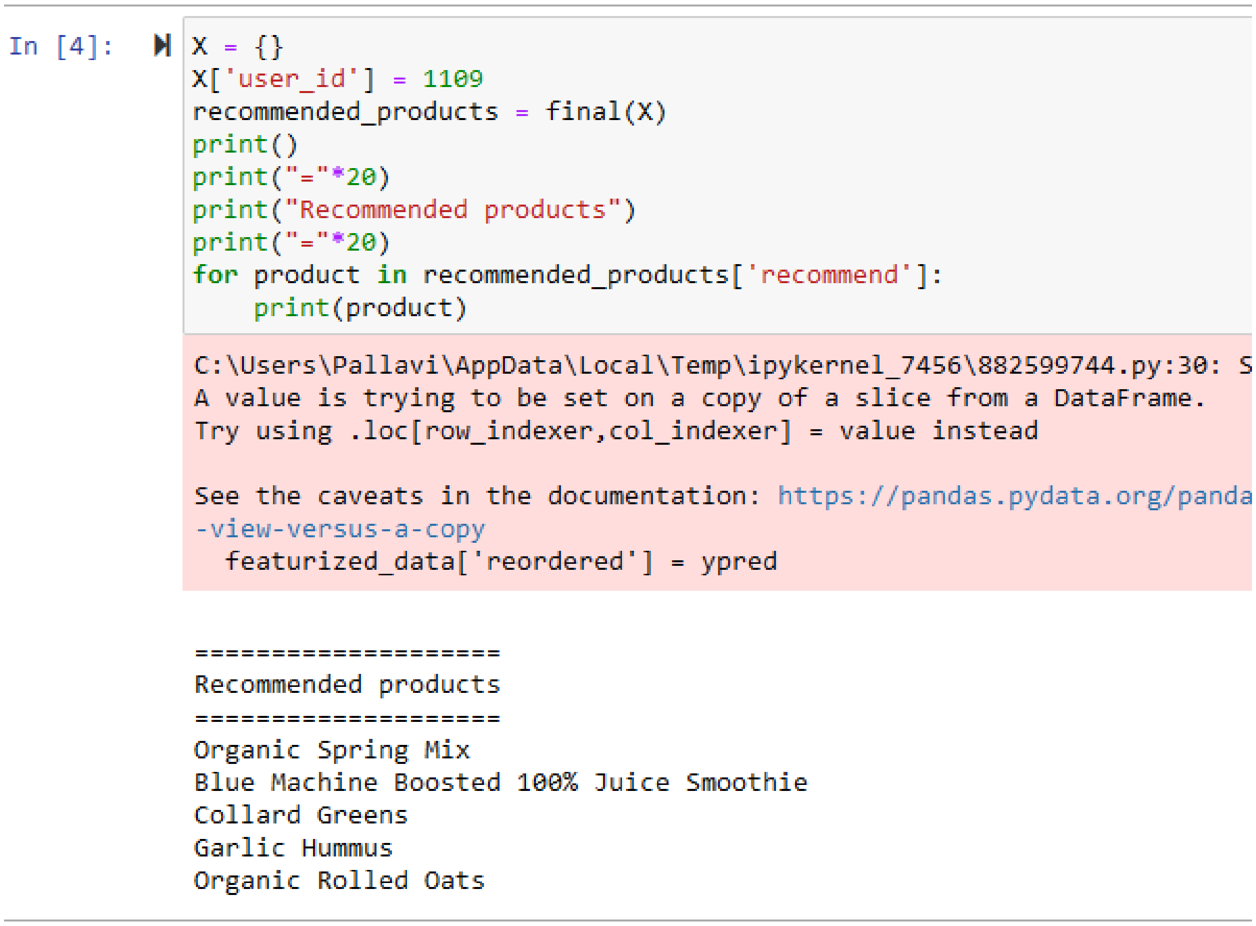
```python
X = {}
X['user_id'] = 119909
recommended_products = final(X)
print()
print("="*20)
print("Recommended products")
print("="*20)
for product in recommended_products['recommend']:
    print(product)
```

```
C:\Users\Pallavi\AppData\Local\Temp\ipykernel_13140\882599744.py:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
-view-versus-a-copy
  featurized_data['reordered'] = ypred
```

```

====================
Recommended products
====================
Bag of Organic Bananas
Organic Cucumber
Organic Avocado
Organic Raspberries
Organic Grape Tomatoes
```



```
In [4]:  X = {}
         X['user_id'] = 1109
         recommended_products = final(X)
         print()
         print("="*20)
         print("Recommended products")
         print("="*20)
         for product in recommended_products['recommend']:
             print(product)
```

```
C:\Users\Pallavi\AppData\Local\Temp\ipykernel_7456\882599744.py:30: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
-view-versus-a-copy
  featurized_data['reordered'] = ypred
```

```

====================
Recommended products
====================
Organic Spring Mix
Blue Machine Boosted 100% Juice Smoothie
Collard Greens
Garlic Hummus
Organic Rolled Oats
```

Citation :

https://towardsdatascience.com/make-working-with-large-dataframes-easier-at-least-for-your-memory-6f52b5f4b5c4

https://kaggle.com/competitions/instacart-market-basket-analysis