Instacart Project Report

Supply Chain Artificial Intelligence

Instructors: Xiaowei Xu; Zhibin Zhang

Group members: Jiaxin Li; Chi Zhang

Contribution: Jiaxin Li 50%
Chi Zhang 50%

2020/05/10

INTRODUCTION, PROBLEM, PURPOSE

We are the data scientists from Instacart, an American technology company that operates as a same-day grocery delivery and pick-up service in the U.S. and Canada. We are the grocery shopping lovers and aim to transform and improve how people buy groceries everyday. By building the data pipeline to analyze the customers' orders, we can clearly see what specific products are more likely to be ordered online and make sure the hottest products are always available in store and make the suggestion to the company based on reorder.

Data Source

Instacart shopping basket analysis:

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

Instacart, a grocery ordering and delivery app, is one of our favorite tech platforms. As a graduate student with a busy academic schedule, I simply don't have the time to do grocery shopping. Instacart aims to make it easy to do the shopping online with the touch of a button. After the customers select the products through the Instacart app, personal shoppers review the order and do the in-store shopping and delivery for customers. You can think of Instacart as the Uber for grocery shopping.

Back in 2017, the company announced its first public dataset release, which is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. The goal is then to predict which previously purchased products will be in a user's next order. This is a classification problem because we need to predict whether each pair of user and product is a reorder or not. This is indicated by the value of the reordered variable, i.e. reordered=1 or reordered=0

1.aisle dataset: contains 134 kinds of product categories to be selected (From the supply chain perspective, this dataset provides the information about the item stored in which aisle in the warehouse).

2.departments dataset: 134 product catalogs are divided into 21 departments, each department has its own department_id.

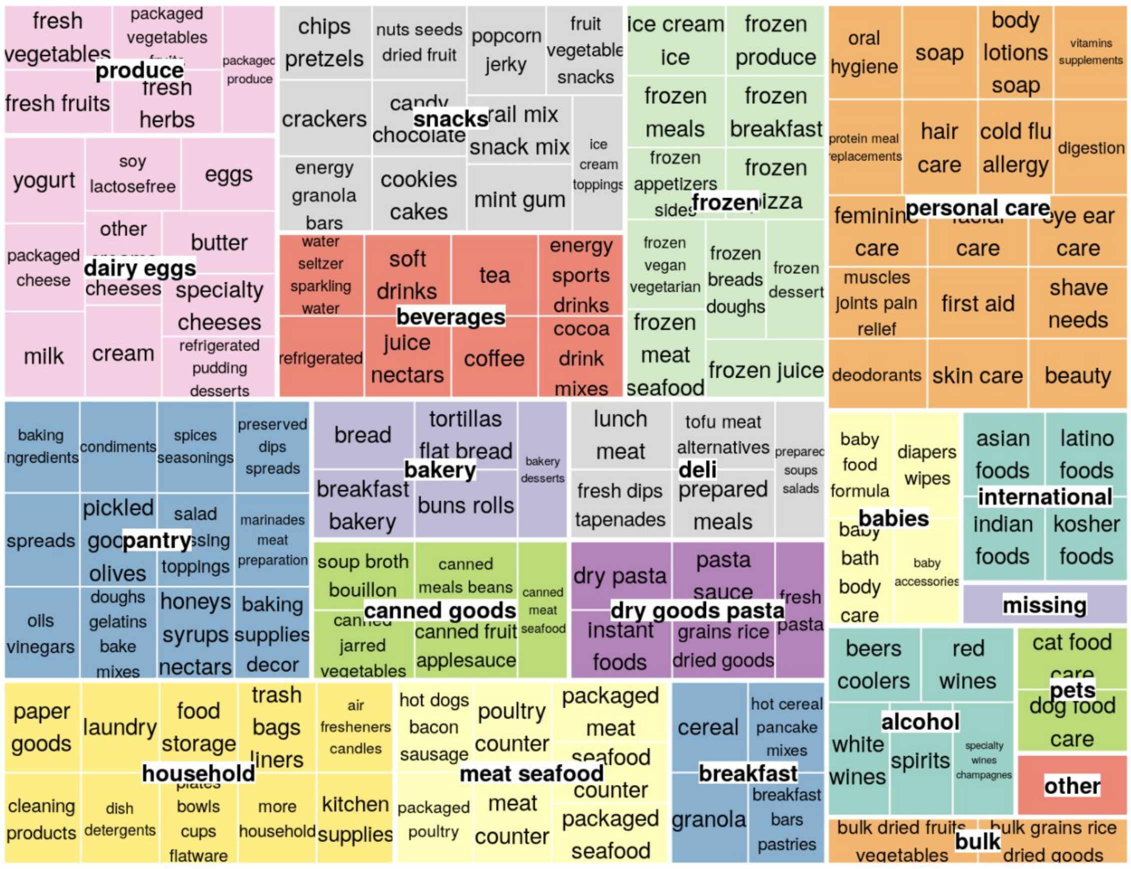
3.order_products__prior & order_products_train dataset: These datasets specify which products were purchased in each order. It contains previous order contents for all customers.

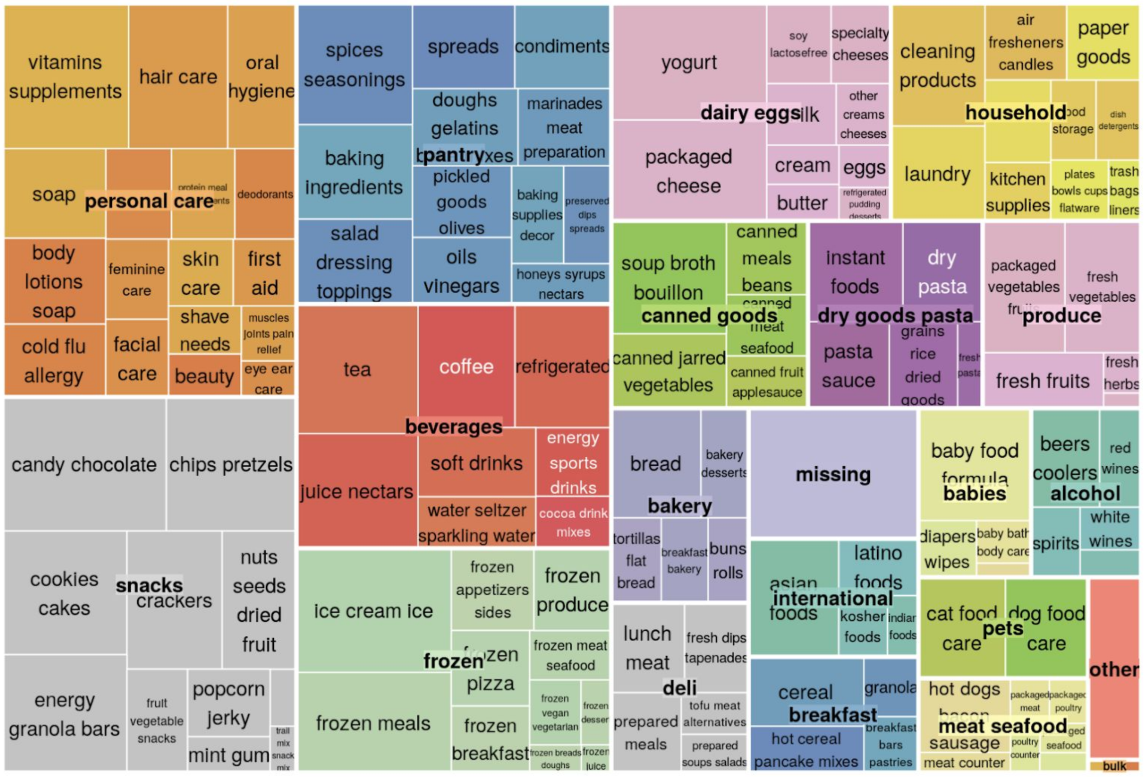
'reordered' indicates that the customer has a previous order that contains the product. Note that some orders will have no reordered items. We may predict an explicit 'None' value for orders with no reordered items.

4.orders dataset: This dataset tells to which set (prior, train, test) an order belongs. We are predicting reordered items only for the test set orders.Information about historical orders has been collected for customers with different user_id, "order_hour_of_day" records the time of placing an order, and “days_since_prior_order” shows the number of days before the last order, and “order_dow” is the day of week.

5. products dataset: It contains 49688 kinds of products, and each product has a product_id, which can represent their names.Each product has its own aisle and department to distinguish.

6. data cleaning: when we merge or process the datasets, we have used the is.na function to check out whether there is missing data.


produce
snacks
frozen
personal care
dairy eggs
beverages
pantry
bakery
deli
babies
international
canned goods
dry goods pasta
missing
household
meat seafood
breakfast
alcohol
pets
other
bulk


personal care
pantry
dairy eggs
household
canned goods
dry goods pasta
produce
beverages
bakery
missing
babies
alcohol
snacks
international
pets
other
frozen
deli
breakfast
meat seafood
bulk

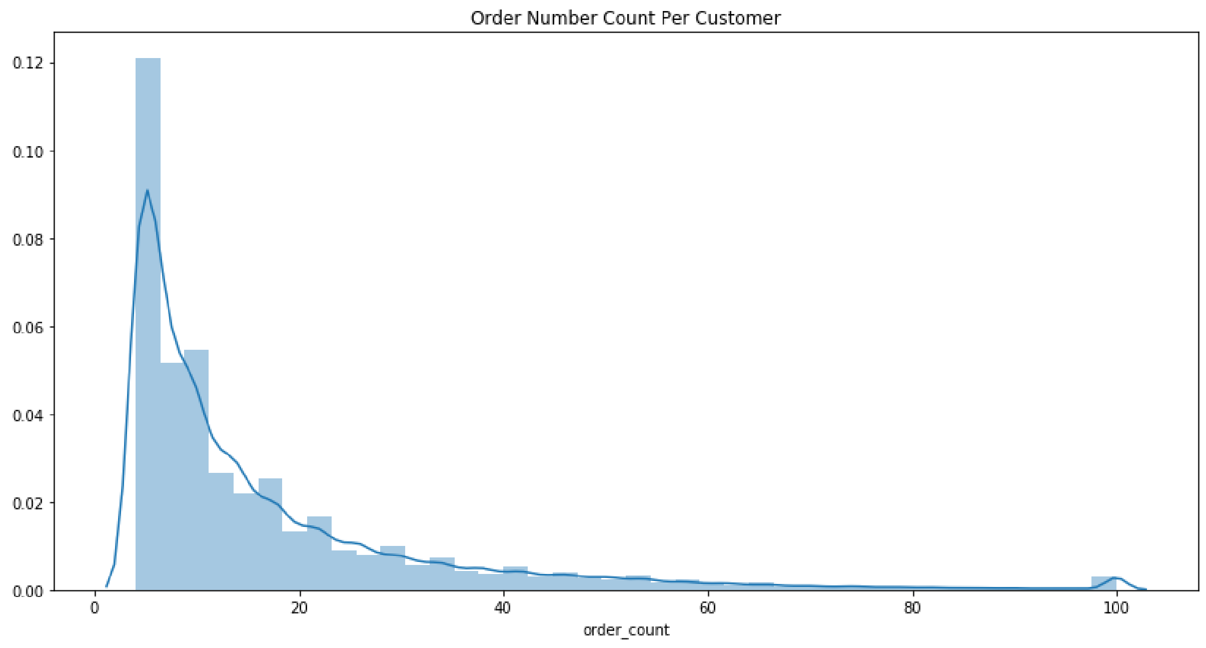
*There are 49,685 products made of 3346,083 orders.

FINDINGS

First of all, we will analyze the five datasets, which can help us understand the customer shopping behavior on the Instacart platform in detail. Knowing which products are most frequently purchased is the first step in Instacart's optimization of software products and recommendation of products to customers. Secondly, we will use the findings in the EDA analysis to select variables that can be applied to construct the model to predict what product will the customer purchase in the next basket.

1.EDA

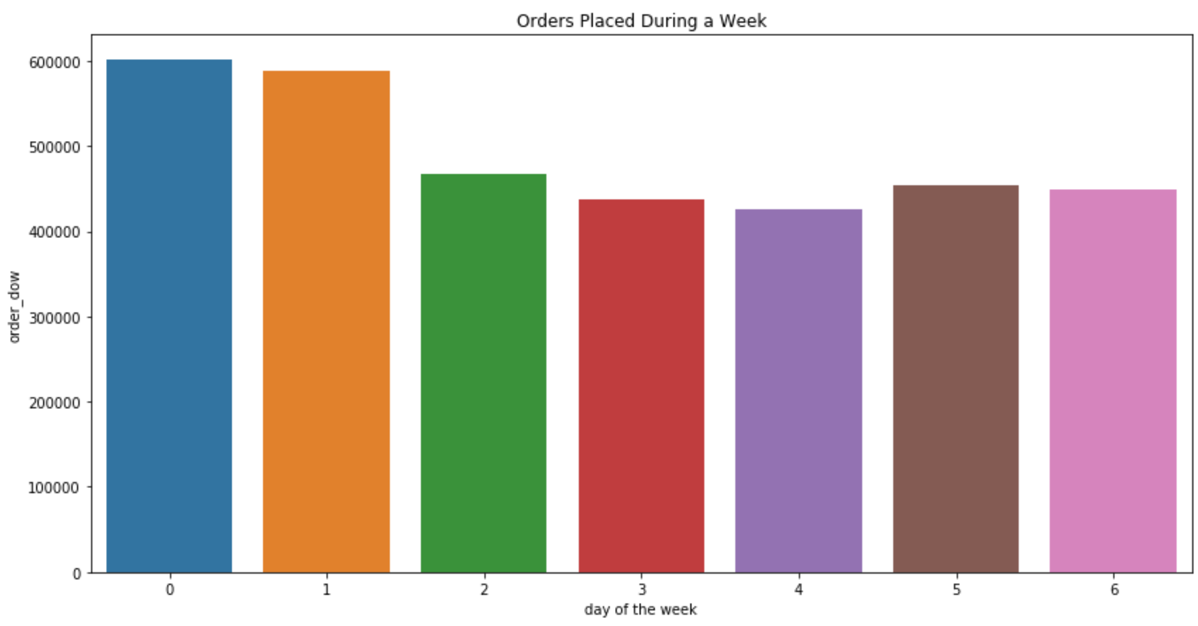
1.1How many customers (observations) and how customers' order number distributed



There are 206209 transactions in our dataset with 131209 treated as training data and 75000 as test dataset.

The average number a customer ordered from Instacart is 17 times with the minimal order number 4 and the largest number order 100(Instacart Aholic :D). Most customers' order number is in the range between 4-10.

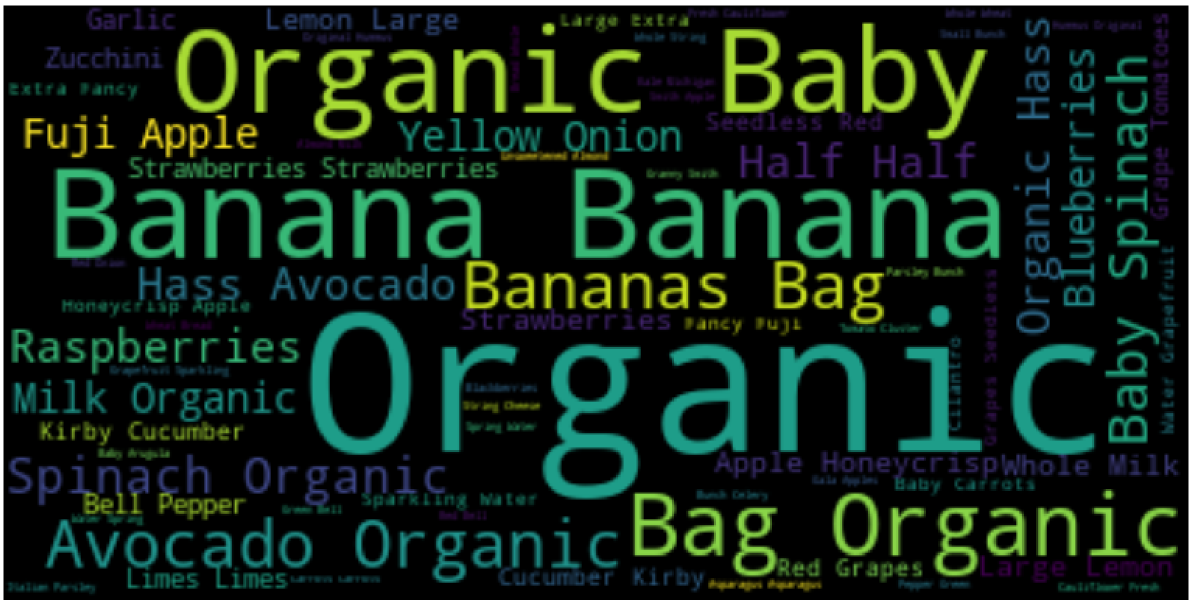
1.2In which day during a week and in which hour during a day people are more likely to do the grocery shopping from Instacart?



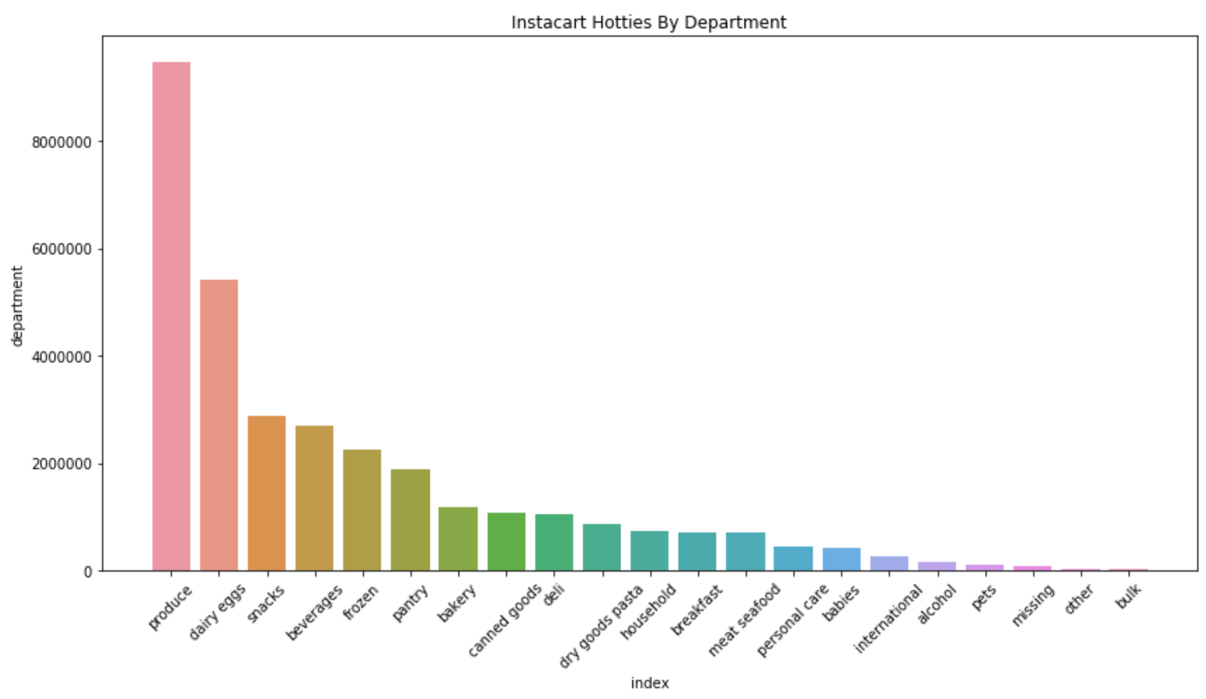
From the barchart,obviously, people like to do online grocery shopping on Sunday and Monday. From orders placed during a day, the hottest time is around 10 am. Therefore, the best time to send email and inform Instacart subscribers to order stuff from Instacart is Sunday 10 AM.

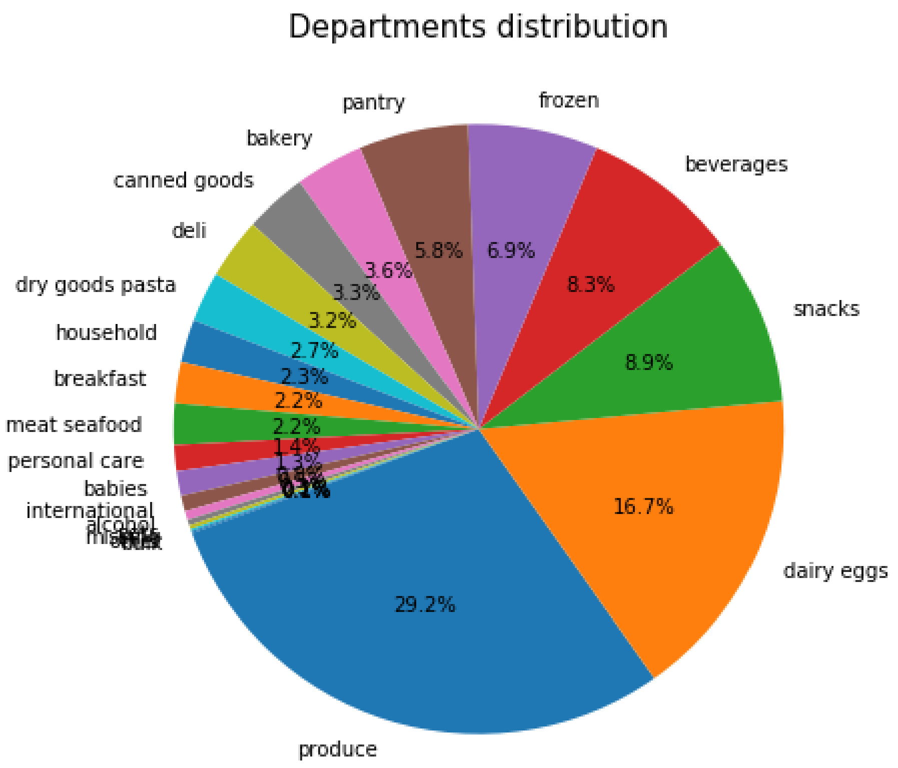
1.3Hottest Product & Department

What products are more popular in Instacart? By knowing that we can inform the suppliers to keep that product in a high level in store and make sure the products are always in stock.



As the word cloud showed above about the top 50 ordered products shows the following characteristics or stuff related: Organic, Banana, Baby, Avocado...The most common words in the those hottest products are Organic, which we can say that people tend to buy more organic foods online than other and food suppliers should prepare and source more organic type of food in order to fulfill the demand.

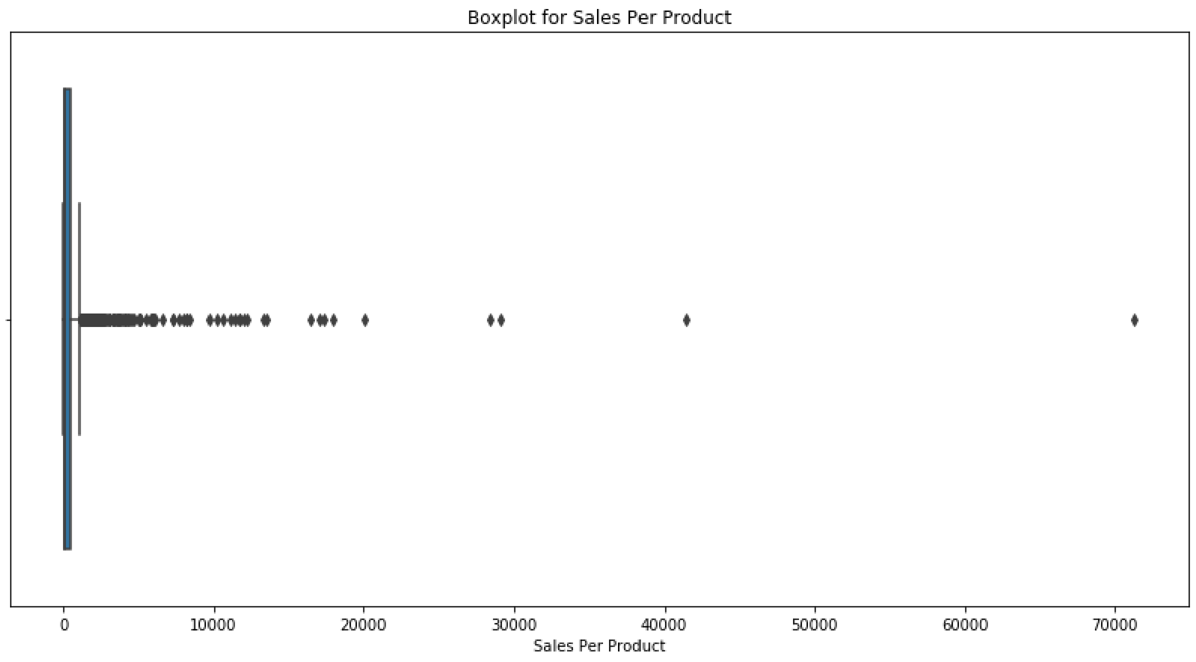


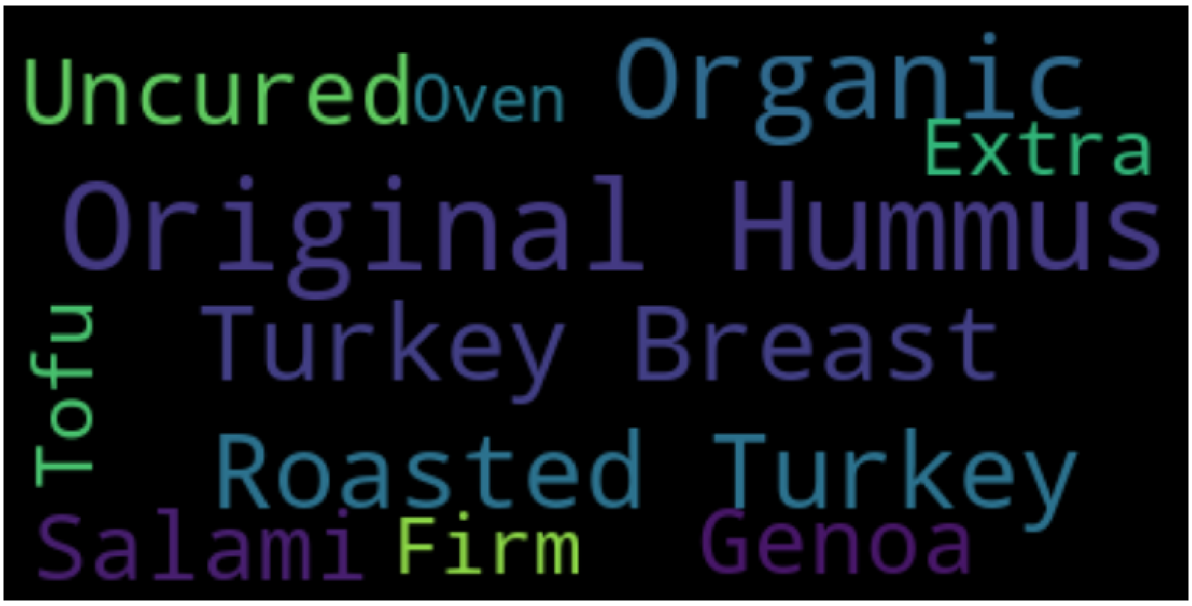


The most frequently ordered products from the ‘produce’ and ‘dairy eggs’ department. Also these two departments have the most products. Let’s dive into the two hottest departments ‘produce’ and ‘dairy & eggs’ to see what specific products contribute to their popularity.

1.4Produce

There are 1321 unique values in the Produce category. The median value of transaction in the Produce department is 115. The mean value of transaction in the Produce department is 795.19. The more than 20000 ratio rate in Produce is 0.3% and the sales contribution rate is 18%.From the word plot we can see that, the most popular products in the ‘produce’ category are: Original Hummus, Turkey Breast, Roasted Turkey, Organic, Salami, Tofu...

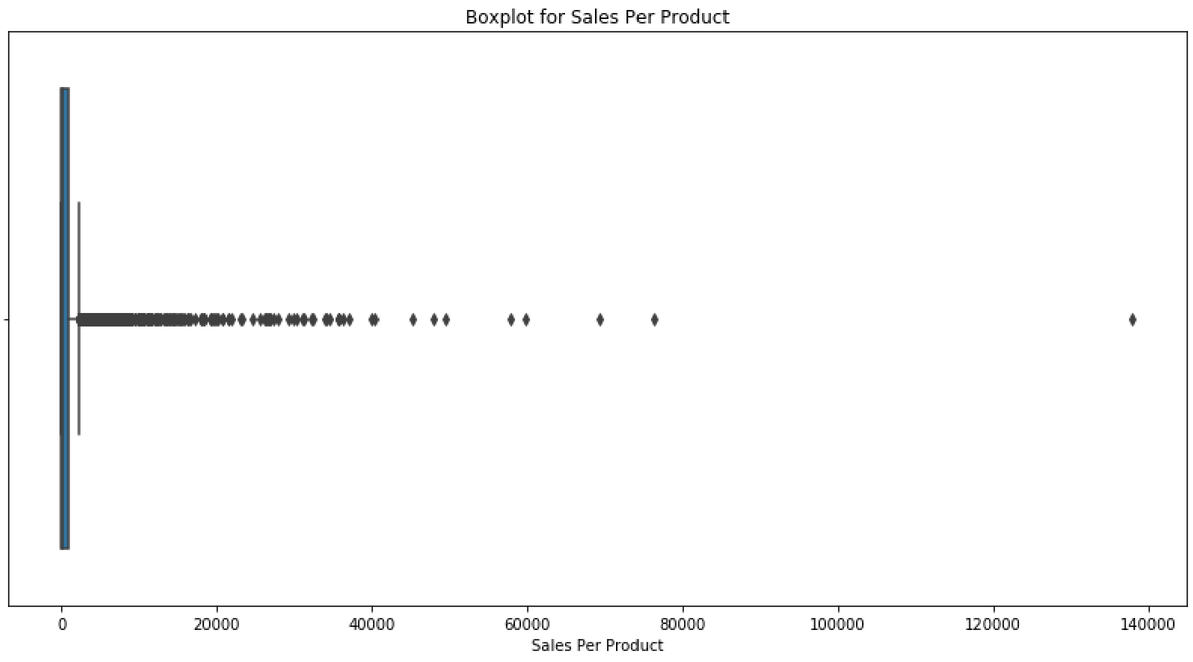




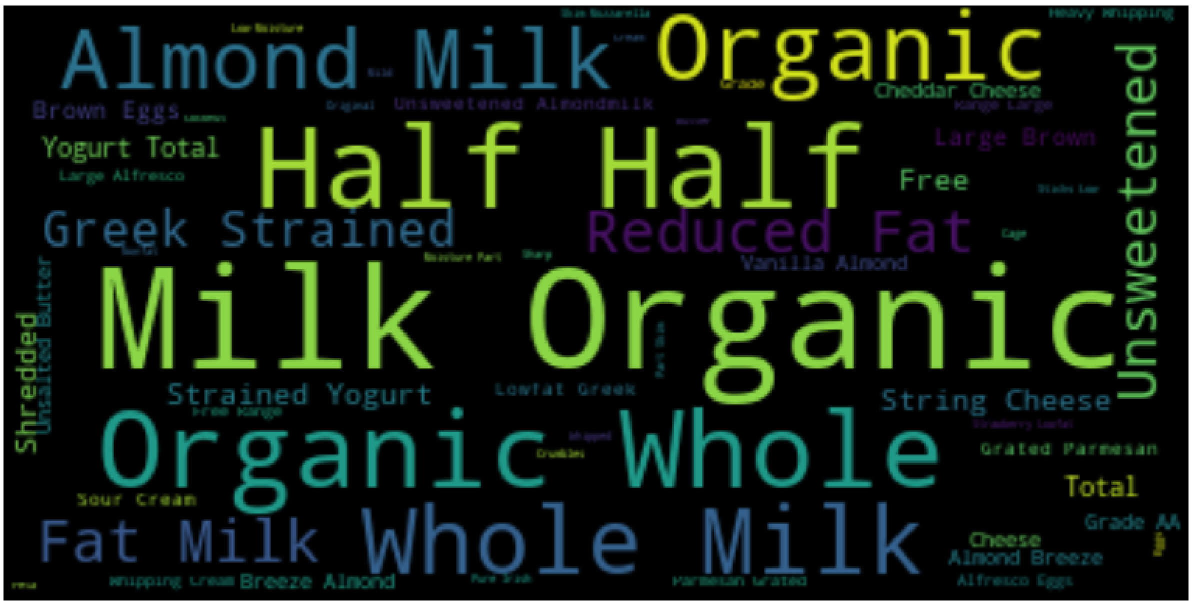
From the word plot we can see that, the most popular products in the 'produce' category are:

Original Hummus, Turkey Breast, Roasted Turkey, Organic, Salami, Tofu...

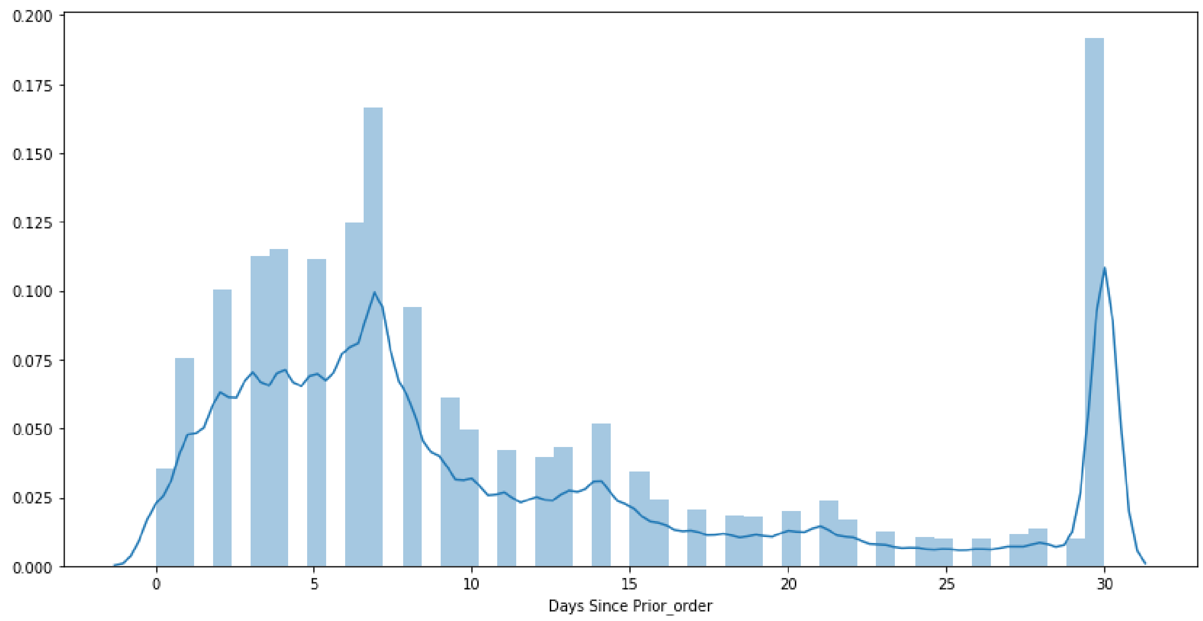
Egg & Dairy:



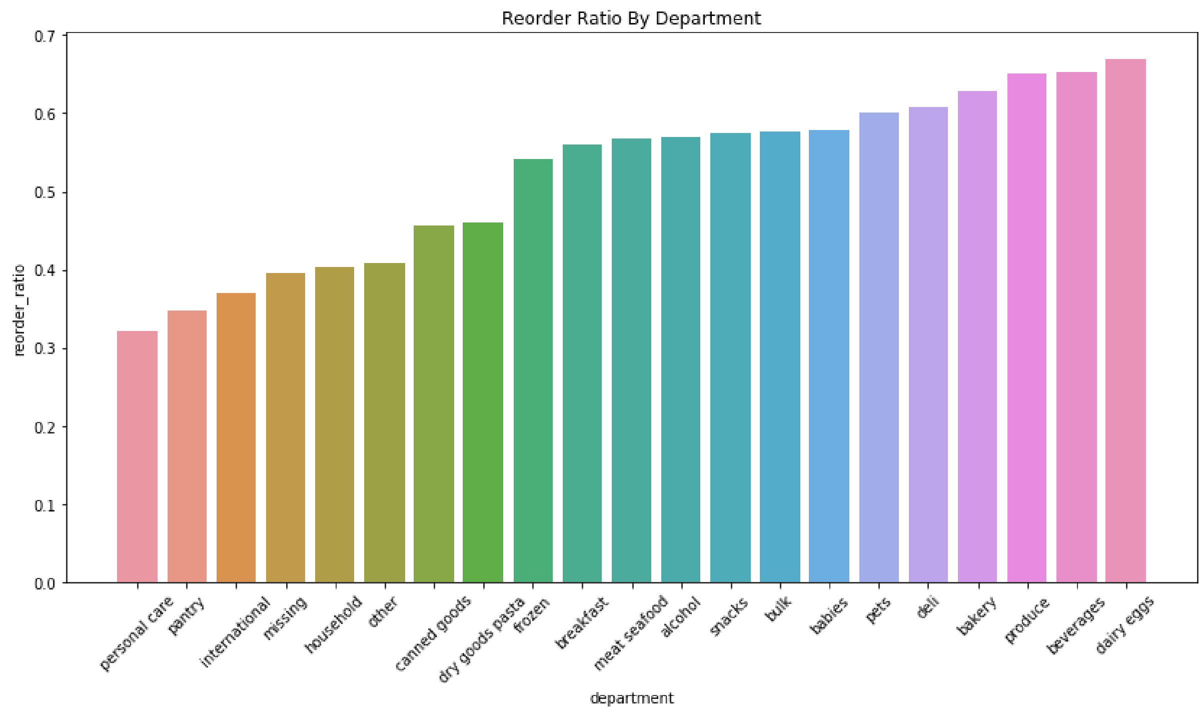
There are 3448 unique products in the dairy and eggs category, the mean value of the transaction is 1570 and median value of transaction is 206. As we can see here, there are a couple of products in the dairy and eggs category that are extremely popular that contribute the most sales. There are 42 kinds of products that have more than 20000 sales, they take up 1.2% of all products in the dairy and eggs category and their sales contribute to 29.32%. Let's figure out what are those hot products that contribute the most sales. From the word cloud of the products that sales are more than 20000, we can conclude that the popular products have the following labels: Milk, organic, half & half, Almond Milk…

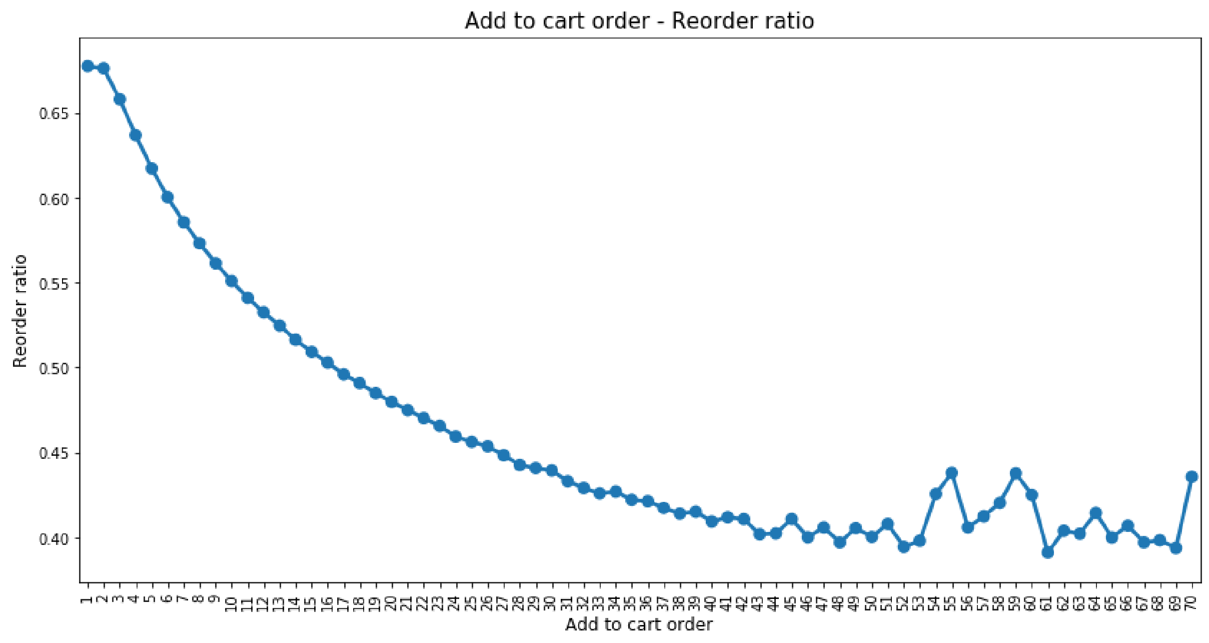


1.5Reorder Exploration



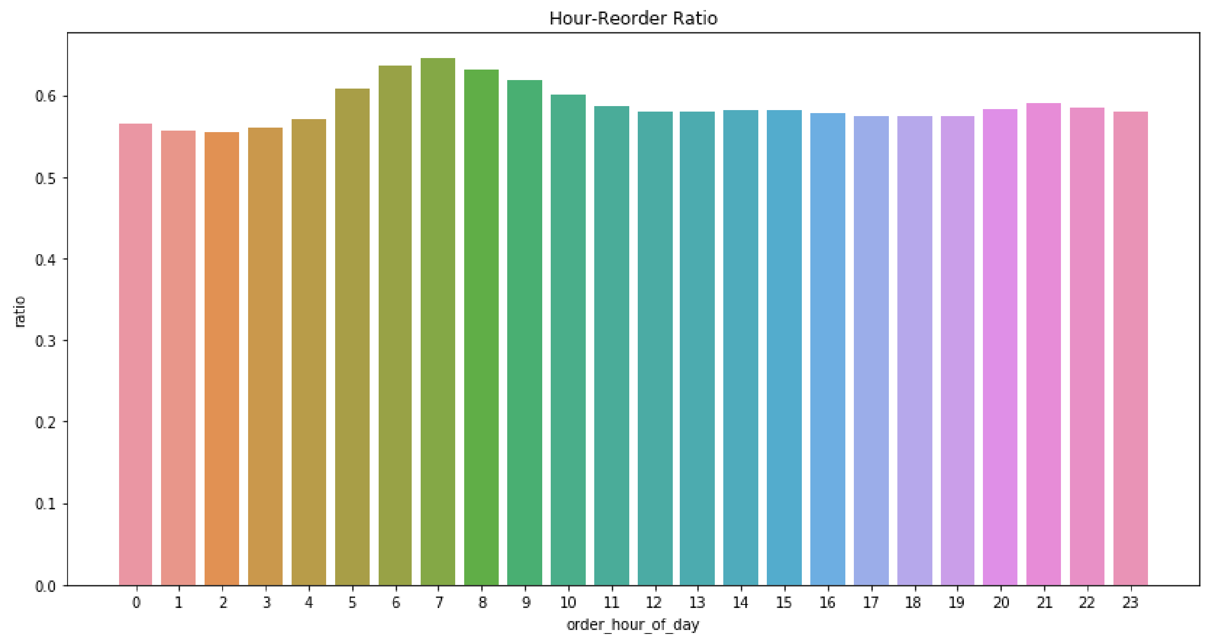
As we can see from this chart there are two reorder peaks: the 7th day and the 30th day. We can conclude that the majority of people like to reorder either on a monthly basis or a weekly basis.

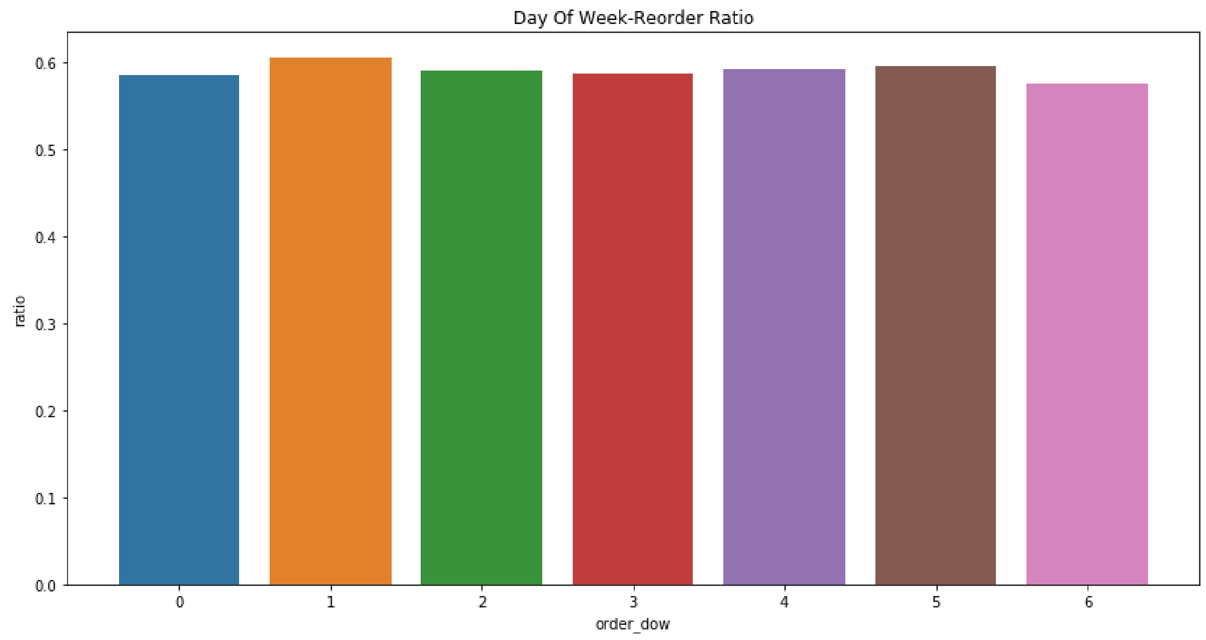




There are 145 add_to_cart values from 1-145, which means some products are put into customers' shopping cart 1 time and some are put into cart 145 times.

Within the 50 add_to_cart value, it shows a linear negative correlationship. The more times the stuff added into the shopping cart, they are less likely to be reordered.





There are no significant differences in reorder among days within a week. The reorder mostly happened at 6-8 pm in the day.

1.6EDA Findings

1. People like to go grocery shopping from Instacart on Sunday & Monday and from 9AM to 3PM.

2. People love buying organic stuff from Instacart. Produce and dairy & eggs are the two departments that have the most order and product diversity.

3. Reorder happened the most in the 7th days and 30th days interval.

4. Reorder is highly related to product-add-to-cart: The more times the stuff added into the shopping more, they are less likely to be ordered.

5. In the morning(6am-8am), products are more likely to be reordered.

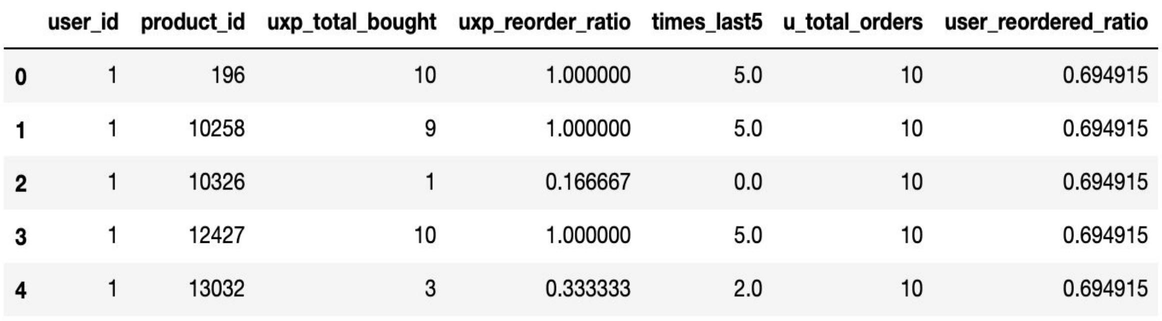
2.Feature Engineering

From the EDA, we clearly know whether re-order or not is highly related to: a. customer behavior(eg: number of items per transaction); b. product characteristics(eg: product category); c. customer - product relationship (reorder times for a specific product). So, we are now ready to identify and calculate predictor variables based on the previous findings. We can create various types of predictors such as:

- **User predictors** describing the behavior of a user e.g. total number of orders of a user.
- **Product predictors** describing characteristics of a product e.g. total number of times a product has been purchased.
- **User & product predictors** describing the behavior of a user towards a

specific product e.g. total times a user ordered a specific product.

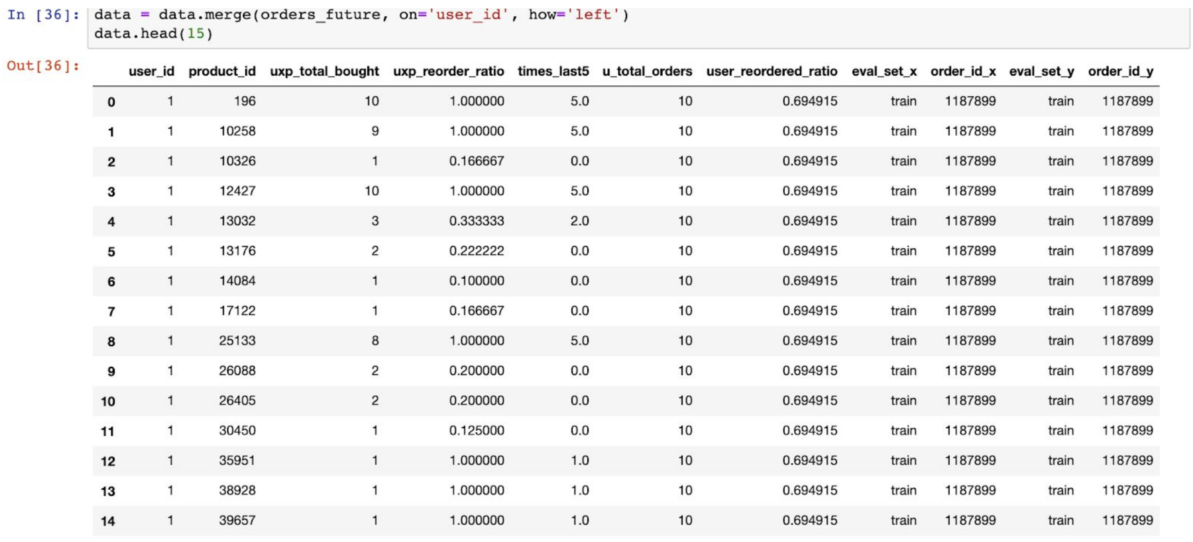
Based on what we mentioned above, we'll use the following as predictors:1. Numbers of orders per customer; 2. How frequent a customer has reordered products; 3. Number of purchases for each product; 4. What's the probability for a product to be reordered ; 5. How many times a user bought a product; 6. How frequently a customer bought a product after its first purchase; 7. How many times a customer bought a product on its last 5 orders. We generated the test evaluation dataset as below:



	user_id	product_id	uxp_total_bought	uxp_reorder_ratio	times_last5	u_total_orders	user_reordered_ratio
0	1	196	10	1.000000	5.0	10	0.694915
1	1	10258	9	1.000000	5.0	10	0.694915
2	1	10326	1	0.166667	0.0	10	0.694915
3	1	12427	10	1.000000	5.0	10	0.694915
4	1	13032	3	0.333333	2.0	10	0.694915

For this dataset it means: for user 1, she has a total of 10 orders at instacart and she purchased product 196 in her order_id 1187899 and 69% of her purchases are reordered. Here uxp_reorder_ratio is used to describe how many times a user bought a product out of how many times she had the chance to buy it (starting from her first purchase of the product). For example, for product 196 she had 1 exp)reorder_ratio, that means if she bought it this product on her 3 third order(10 in total), the 4,5,6,7,8,9,10 the rest of her order all has this product. Also, product 196 is in her last 5 order 5 times.

Then, we split the data into two parts: training data and test data:



```
In [36]: data = data.merge(orders_future, on='user_id', how='left')
         data.head(15)
```

Out[36]:

	user_id	product_id	uxp_total_bought	uxp_reorder_ratio	times_last5	u_total_orders	user_reordered_ratio	eval_set_x	order_id_x	eval_set_y	order_id_y
0	1	196	10	1.000000	5.0	10	0.694915	train	1187899	train	1187899
1	1	10258	9	1.000000	5.0	10	0.694915	train	1187899	train	1187899
2	1	10326	1	0.166667	0.0	10	0.694915	train	1187899	train	1187899
3	1	12427	10	1.000000	5.0	10	0.694915	train	1187899	train	1187899
4	1	13032	3	0.333333	2.0	10	0.694915	train	1187899	train	1187899
5	1	13176	2	0.222222	0.0	10	0.694915	train	1187899	train	1187899
6	1	14084	1	0.100000	0.0	10	0.694915	train	1187899	train	1187899
7	1	17122	1	0.166667	0.0	10	0.694915	train	1187899	train	1187899
8	1	25133	8	1.000000	5.0	10	0.694915	train	1187899	train	1187899
9	1	26088	2	0.200000	0.0	10	0.694915	train	1187899	train	1187899
10	1	26405	2	0.200000	0.0	10	0.694915	train	1187899	train	1187899
11	1	30450	1	0.125000	0.0	10	0.694915	train	1187899	train	1187899
12	1	35951	1	1.000000	1.0	10	0.694915	train	1187899	train	1187899
13	1	38928	1	1.000000	1.0	10	0.694915	train	1187899	train	1187899
14	1	39657	1	1.000000	1.0	10	0.694915	train	1187899	train	1187899

3.Predictive Analysis

3.1Gerneral Process

To create the predictive model we:

- Import and reshape data(pd.merge)
- Feature engineering: This step includes identifying and calculating features from the initial datasets provided by Instacart.
- Create train and test DataFrames
- Fit the predictive model: In this step we train a predictive model through the train dataset
- Test accuracy(cross-validation/F1)

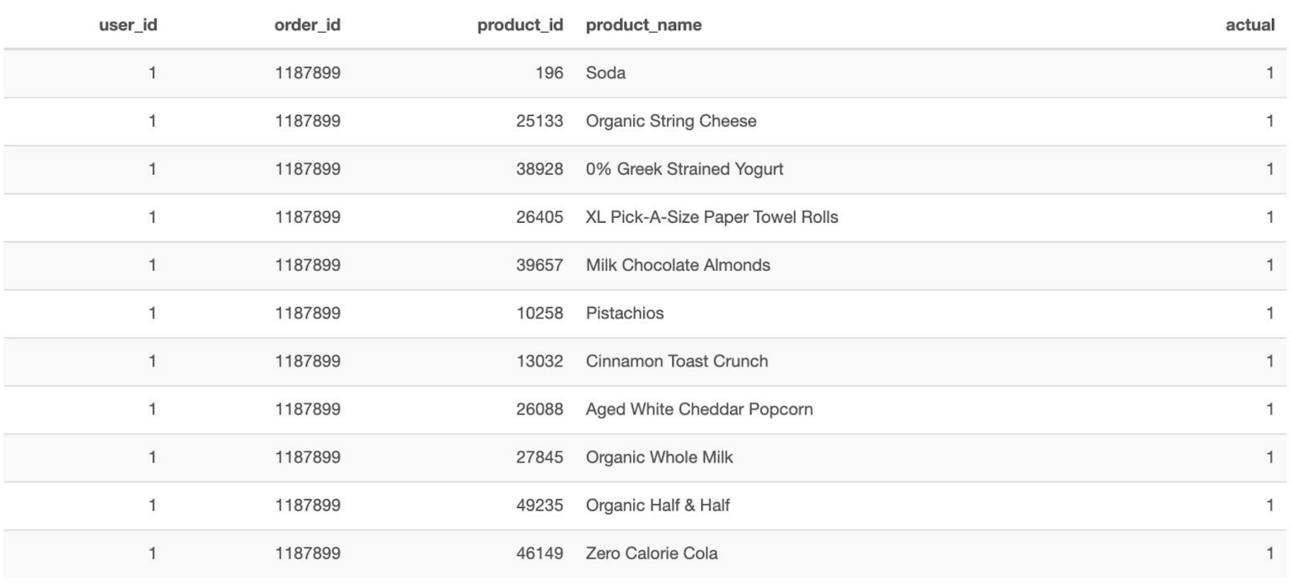
3.2Naive Prediction & Baseline Prediction

In this part, we will use the RStudio to generate two simple models, one is the Naive Prediction Model and another is the Baseline Prediction Model.

Predicting what product will the customer purchase in the next basket means estimation of probably if each product being purchased before will be purchased again. This is a classification problem, as well as a regression of probability of repurchases. The output of the prediction is to be evaluated against the training evaluation set with below table columns:

- <user_id product_id>: as key
- label : 1 or 0 , this is the target label

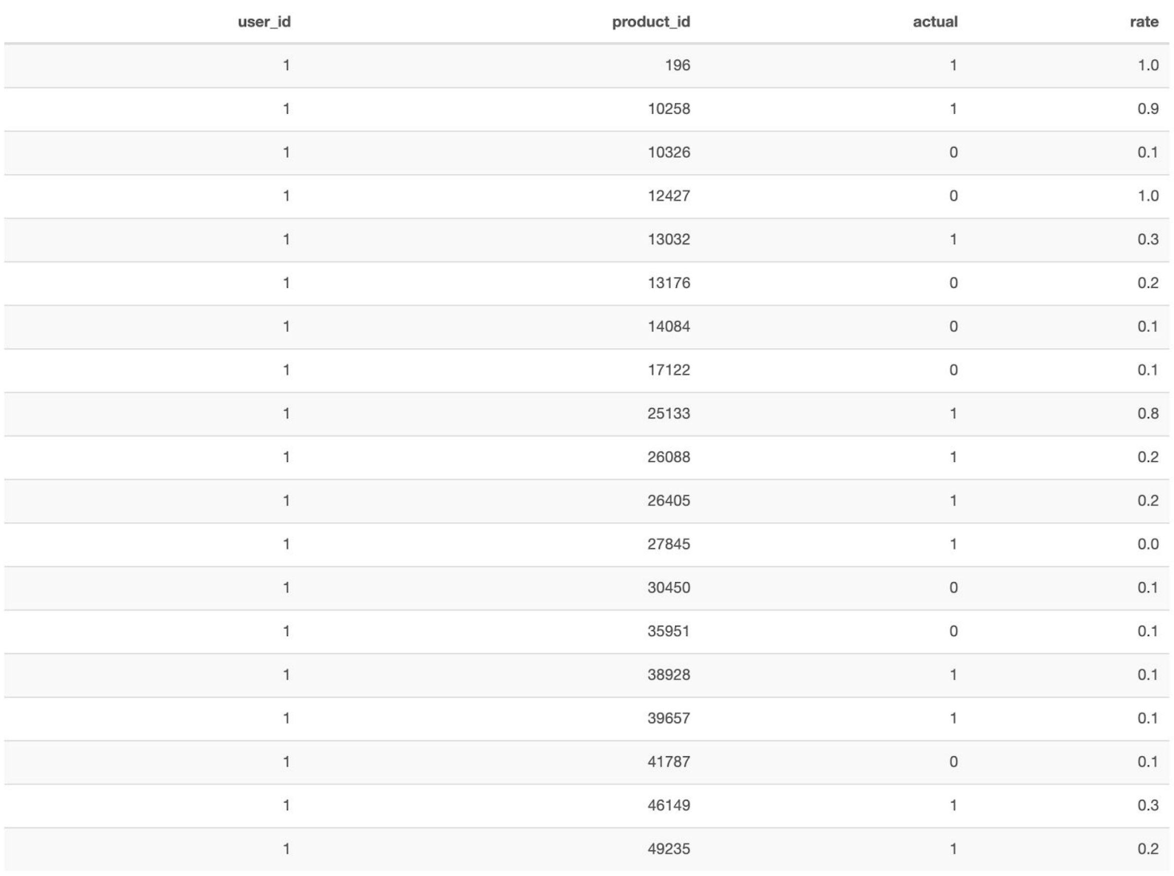
*We created the training label, which is the value or either 1 or 0 to indicate the actual basket content. All prediction will be evaluated against this training labe



user_id	order_id	product_id	product_name	actual
1	1187899	196	Soda	1
1	1187899	25133	Organic String Cheese	1
1	1187899	38928	0% Greek Strained Yogurt	1
1	1187899	26405	XL Pick-A-Size Paper Towel Rolls	1
1	1187899	39657	Milk Chocolate Almonds	1
1	1187899	10258	Pistachios	1
1	1187899	13032	Cinnamon Toast Crunch	1
1	1187899	26088	Aged White Cheddar Popcorn	1
1	1187899	27845	Organic Whole Milk	1
1	1187899	49235	Organic Half & Half	1
1	1187899	46149	Zero Calorie Cola	1

3.2.1Naive Prediction

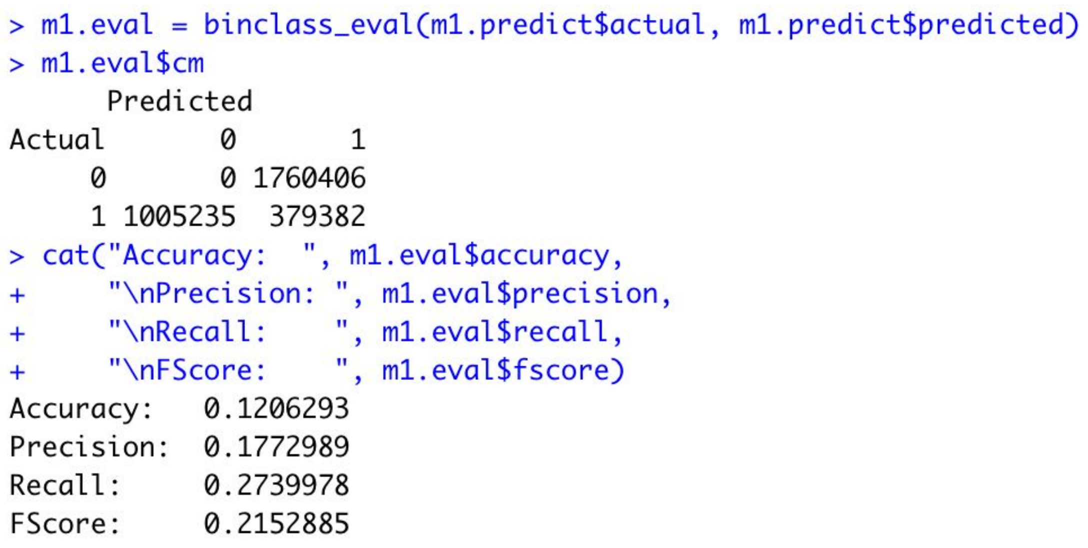
In this model, We can simply predict the basket based on the user's last order. Below is the sample of output.



user_id	product_id	actual	rate
1	196	1	1.0
1	10258	1	0.9
1	10326	0	0.1
1	12427	0	1.0
1	13032	1	0.3
1	13176	0	0.2
1	14084	0	0.1
1	17122	0	0.1
1	25133	1	0.8
1	26088	1	0.2
1	26405	1	0.2
1	27845	1	0.0
1	30450	0	0.1
1	35951	0	0.1
1	38928	1	0.1
1	39657	1	0.1
1	41787	0	0.1
1	46149	1	0.3
1	49235	1	0.2

Due to the high imbalanced dataset, We use the F1 Score to measure the model's performance, instead of accuracy, we also had created a custom function to build a confusion matrix and derive other binary classification metrics.

Performance Evaluation:



```
> m1.eval = binclass_eval(m1.predict$actual, m1.predict$predicted)
> m1.eval$cm
      Predicted
Actual       0       1
     0       0 1760406
     1 1005235  379382
> cat("Accuracy:  ", m1.eval$accuracy,
+     "\nPrecision: ", m1.eval$precision,
+     "\nRecall:    ", m1.eval$recall,
+     "\nFScore:    ", m1.eval$fscore)
Accuracy:   0.1206293
Precision:  0.1772989
Recall:     0.2739978
FScore:     0.2152885
```

3.2.2Baseline Prediction

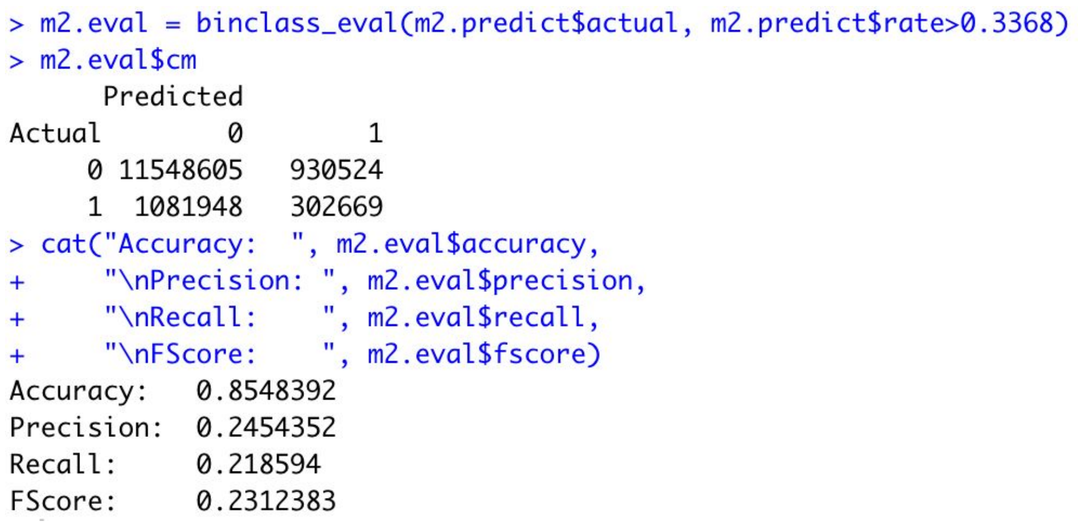
In this model, we predict products in the basket by estimating their frequency of repurchased. This way we get a ratio to indicate probability of re-purchases. We use the ROCR package to predict the best cutoff point (at which above this cutoff we shall predict for re-order) that gives us the optimum F1 score. Below is the sample of output.

user_id	product_id	actual	rate
1	196	1	1.0
1	10258	1	0.9
1	10326	0	0.1
1	12427	0	1.0
1	13032	1	0.3
1	13176	0	0.2
1	14084	0	0.1
1	17122	0	0.1
1	25133	1	0.8
1	26088	1	0.2
1	26405	1	0.2
1	27845	1	0.0
1	30450	0	0.1
1	35951	0	0.1
1	38928	1	0.1
1	39657	1	0.1
1	41787	0	0.1
1	46149	1	0.3
1	49235	1	0.2

In order to maximize F1 Score, we need to set the cutoff threshold to 0.3368, which is the next step. That's because the prediction is based on probability, we need to build a function to discover cutoffs that optimize various performance metrics.

	max	cutoff
best.accuracy	0.9001268	Inf
best.ppv	0.4345635	0.8588235
best.recall	1.0000000	0.0000000
best.fscore	0.2312383	0.3368421
best.tpr_fpr	6.9266631	0.8588235

Performance Evaluation



```
> m2.eval = binclass_eval(m2.predict$actual, m2.predict$rate>0.3368)
> m2.eval$cm
      Predicted
Actual        0        1
     0 11548605   930524
     1  1081948   302669
> cat("Accuracy:  ", m2.eval$accuracy,
+     "\nPrecision: ", m2.eval$precision,
+     "\nRecall:    ", m2.eval$recall,
+     "\nFScore:    ", m2.eval$fscore)
Accuracy:   0.8548392
Precision:  0.2454352
Recall:     0.218594
FScore:     0.2312383
```

From the confusion matrix and the table of model performance, we can figure out that we are getting slightly better F1 Score (0.2312) in the Baseline Prediction Model compared to the Naive Prediction Model.

3.3XGBoost & Logistics Regression

3.3.1 XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

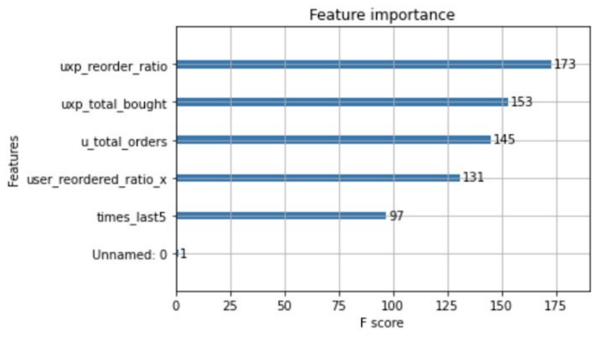
Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

Fit the predictive model:

```
In [15]: X_train, y_train = data_train.drop('reordered', axis=1), data_train.reordered
         parameters = {'eval_metric':'logloss',
                       'max_depth':'5',
                       'colsample_bytree':'0.4',
                       'subsample':'0.75'
                      }
         xgbc = xgb.XGBClassifier(objective='binary:logistic', parameters=parameters, num_boost_round=10)
         model = xgbc.fit(X_train, y_train)
         xgb.plot_importance(model)

         [01:14:26] WARNING: /workspace/src/learner.cc:686: Tree method is automatically selected to be 'approx' for faster sp
         eed. To use old behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f43b09524e0>
```



From the XGBoost result we can see the user reorder ratio dominates the other features, clearly standing out as the most important predictor of reorder or not.

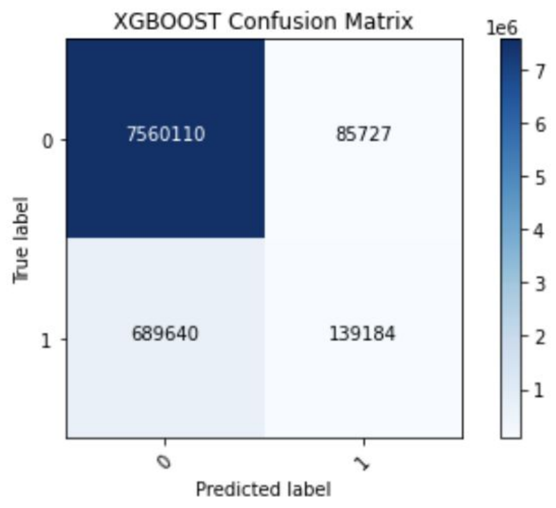
Performance Evaluation

- Confusion Matrix

```
#plot confusion matrix
cnf_matrix = confusion_matrix(y_train, train_pred)
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    import itertools
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['0', '1'],
                      title='XGBOOST Confusion Matrix')
```



After few hours struggling, we finally successfully run the cross validation and got a pretty decent accuracy 90.95%, although we clearly classify most of cased, when we have a closer look at the CM, the false negative has way more numbers than false positive, which means most false classification happens to when this A product from A customer are not reordered but our model tells they are. But in general this is a pretty decent one, capturing more than 90% right classification.

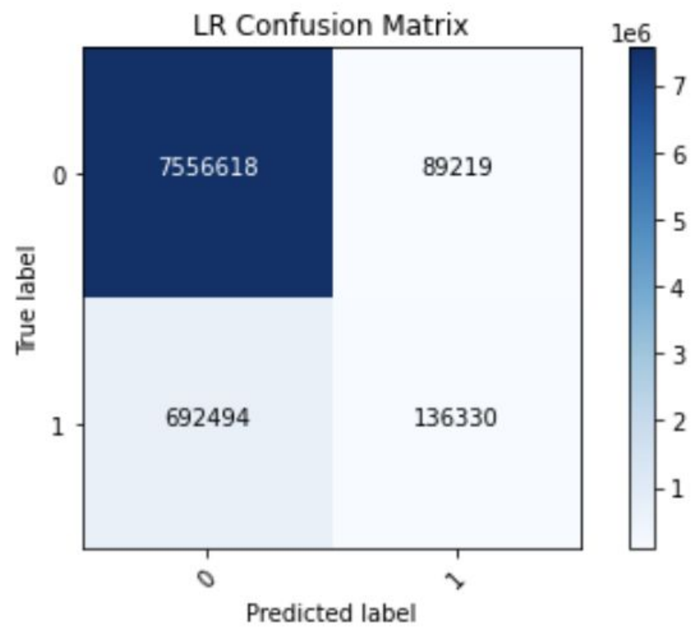
3.2.2 Logistics Regression

Logistics Regression is the model used to model the probability of a certain class.

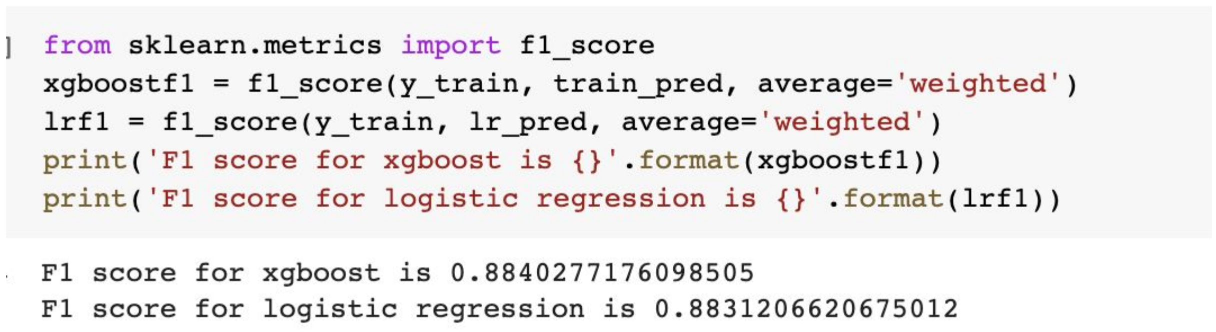
Performance Evaluation

- Confusion Matrix

```
lr_pred = logreg.predict(X_train).astype(int)
cmlr_matrix = confusion_matrix(y_train, lr_pred)
plt.figure()
plot_confusion_matrix(cmlr_matrix, classes=['0', '1'],
                      title='LR Confusion Matrix')
```



Performance Evaluation(weighted recall & precision)



```python
from sklearn.metrics import f1_score
xgboostf1 = f1_score(y_train, train_pred, average='weighted')
lrf1 = f1_score(y_train, lr_pred, average='weighted')
print('F1 score for xgboost is {}'.format(xgboostf1))
print('F1 score for logistic regression is {}'.format(lrf1))
```

```
F1 score for xgboost is 0.8840277176098505
F1 score for logistic regression is 0.8831206620675012
```

After drawing the confusion matrix, we found it’s very imbalanced type 1 and type 2 error, to weigh recall and precision in the leverage level, we will use F1 score instead. F1 considers both the precision *p* and the recall *r* of the test to compute the score. As the result shows, both XGBoost and logistics regression reach a high F1 score. XGBoost has a better performance in accuracy and F1 score than other algorithms.

Conclusion

Based on the Project:

1. XGBoost is the best performance model in our case with more than 90.85% accuracy and weighted recall & precision rate 0.884. In this case, we generated 4 kinds of models: Naive Prediction Model, Baseline Prediction Model, XGBoost Model and Logistic Regression model. The first two prediction models have a limited number of influencing factors for repurchases, so when comparing the performance of the models, we will find that their prediction accuracy and F1 score are relatively low. This enlightens us that when using Naive PredictionModel and Baseline Prediction Model to predict data, the number of hidden dependent variables involved cannot be too large, otherwise it will affect the performance of the prediction model and fail to achieve more effective prediction results.
2. The most important factors to influence if a customer will reorder or not is uxp_reorder_ratio and uxp_total_bought. We can learn from: if a customer gets a chance to buy the product, the person will be more likely to reorder. The more times the customer buys this product, the more likely she will reorder. As the data scientists at Instacart, we recommend the company to expand the products exposure in different categories and invest money on marketing where products are less exposed. Also, the company could reach the large scale of benefits by bundling products together, like 0the shoes + socks products combo or instant noodles with coke combo.
3. Make sure hot products are always available: Instacart takes full advantage of the convenience and quality of customers to provide groceries to thousands of customers

in 375 cities in the United States. With a lean business model, Instacart integrates Twilio 's SMS and notification technology to help them handle end-to-end customer service Shoppers issue alerts and notifications, and manage their growing remote workforce. Instacart relies on Twilio's SMS notifications to manage their order confirmation, delivery ETA, customer satisfaction surveys, and employee shift alerts. Whether you can have an efficient, accurate and stable forecast model is very important for instacart. Meeting the customer's purchasing needs is the only criterion for measuring the quality of the forecast model.

4. For the company:According to an effective prediction model, Instacart can expand the exposure of frequently purchased products in the user-selected interface, or bundle best-selling products for sale. In this way, it is convenient for customers to purchase products and save time, and more importantly, it can cater to the preferences of customers, thereby establishing customer loyalty and achieving greater profitability.

What we learned:

1. Data science workflow: This project gave us a clearer picture of what a complete data science workflow looks like and how to use AI knowledge combined with programming practice to solve a supply chain problem. The most valuable knowledge we learnt is how to conduct a data science project from scratch and how to solve the problem using data. The whole workflow is like: Come up a business problem --->find the appropriate dataset that can be used to solve the problem ---> data

cleansing ---> data exploration(discover relationships,information detection) --->feature engineering(select/create new features) ---> modeling --->testing/evaluation --->improve model

2. Method to deal with huge dataset: we can use library gc, and free up memory by deleting unnecessary dataset and gc.collect()
3. Feature engineering techniques: in this project, we spent most of our time in feature engineering. What we did to make a ideal feature is: Transform the character variables into category, identify useful info by looking at the relationship between features and target variable and calculate predictor variables based on original dataset.

Reference

[1] **Instacart XGboost GridSearch Notebook**

https://www.kaggle.com/kokovidis/ml-instacart-xgboost-gridsearch-notebook/notebook
#5.-Apply-predictive-model-(predict)