Final Project - Instacart Market Basket Analysis

Golden Gate University MSBA326

Predictive Analytics & Machine Learning

Instructor: Jahan Ghofraniha

Hanqing (Jamie) Huang
Jing (DJ) Ding
Xiaoting (Theresa) Liu
Xiangwen (Jessica) Meng

Table of Content

Executive Summary

The project comes from Instacart market basket analysis from Kaggle's feature prediction competition. Here is a short description from the competition: whether you shop from meticulously planned grocery lists or let whimsy guide your grazing, our unique food rituals define who we are. Due to the serious competition among grocery delivery application, we are going to explore the data to gain deeper insights on this application by finding out what are the problems it should solve, identifying the most profitable segments and proposing better business strategies or suggestions to maximize reorder quantity.

Introduction

Background

Instacart, a grocery ordering and delivery application, aims to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them. After selecting products through the Instacart app, personal shoppers review your order and do the in-store shopping and delivery for you.

Resource

The project contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. There are seven datasets with the following column names:

- aisles.csv (aisle_id, aisle)
- departments.csv (department_id, department)
- order_products_*.csv (order_id, product_id, add_to_cart_order, reordered)
- orders.csv (order_id, user_id, eval_set, order_number, order_dow, order_hour_of_day, days_since_prior_order)
- products.csv (product_id, product_name, aisle_id, department_id)
- sample_submission.csv (order_id, products)

Goals

Our goals for this project is to identify the most profitable reorder products, optimizing product segments to maximize profit and selecting important features to improve reorder accuracy.

Literature Review

Current Situation

It is interesting that Amazon is offering a similar service starting from 2018. Amazon Prime members can access online grocery-shopping and delivery from Whole Foods. The problems that Amazon faced including wrong orders, inaccurate stock information, Whole Foods employee distraction from Amazon workers (Dow Jones Institutional News, 2019). The publication (Dow Jones Institutional News, 2019) indicates that most of the companies lack of technology that can readily track inventory in real-time. Instacart, as the largest third-party online order delivery company, has problem with incomplete orders that raise customer satisfaction. Not only Amazon but also other online order delivery companies are facing the same challenge. Besides the facts that some people complain about grocery-delivery service, there are others that are addicted to this service. Haddon (2019) indicates that younger people make more frequent purchases. Young professionals are more relying on the delivery services as they have no time to buy food or cook.

Advantages

There is an interesting article talks about the factors that affect user's behavioural e-loyalty. This article introduces a study that aims to examine various variables that can best explain customer satisfaction and behavioural e-loyalty ragards online local food shopping (Lopez, Virto and San-Martin, 2018), which means that these companies have a great advantage to improve customer loyalty because they can access much more available data from the system. When we open the Instacart website, we instantly notice that our address and zip code is asked as a starter. Scrolling down the page, there are 6 major features are identified by Instacart: products you like, same-day delivery, save time and money, deals that delight, browse products and fresh produce picked perfectly (Instacart.com). These are the competitive advantages Instacart has compared to other grocery delivery service.

Approach Methodology

In this paper, we are going to use Python to provide multi-dimensional analysis. In order to identify the most possible reordered products, we will start with descriptive analysis to provide an overview on the data. Then we are going to provide predictive analysis by manipulating the data to see which products are highly likely to be reordered based on different features. Finally, we will build a classification model as our prescriptive analysis to find out purchase patterns and propose a suggestion for the grocery in order to maximum reorders.

Descriptive Analysis

Preprocessing Data

After loading the dataset, there is only one column in dataframe order has 206,209 null values. However, they are reasonable to be as null values, so here we will not drop them. To better review the results, we loop the datasets with their names to get the output in one time (Appendix 1).

After previewing datasets, we decide to visualize some key variables based on time-series, distribution, boxplot. Moreover, we are going to merge some datasets for further analysis (Appendix 2).

We can see the statistical information for each dataset. There are a total of 21 departments, 134 aisles and 49,688 products present in the Instacart shopping application (Appendix 3).

Exploratory Data Analysis

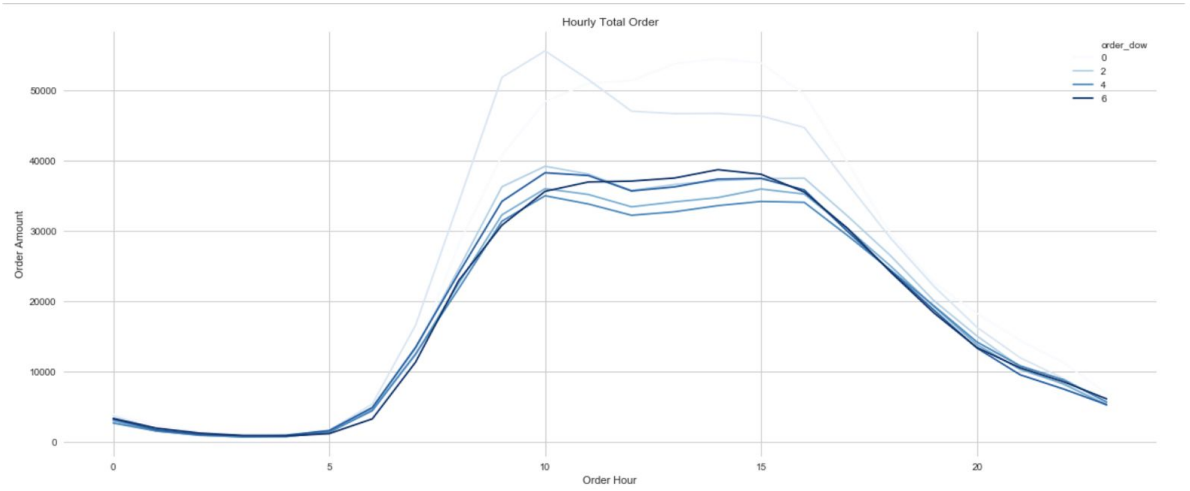

Figure 1. Time-series Overview on Days of The Week & Hours

With the result above, The lowest order amount is mainly from midnight to 5am in the morning while the highest amount is around 7am to 4pm (index 16). Sunday (order_dow on 0) and Monday (order_dow on 1) reach the top 2 highest order amount in a week. Order amount reaches the highest point over 50,000 at 3pm (index 15) Sunday (order_dow on 0) and 10am on Monday (order_dow on 1) respectively.

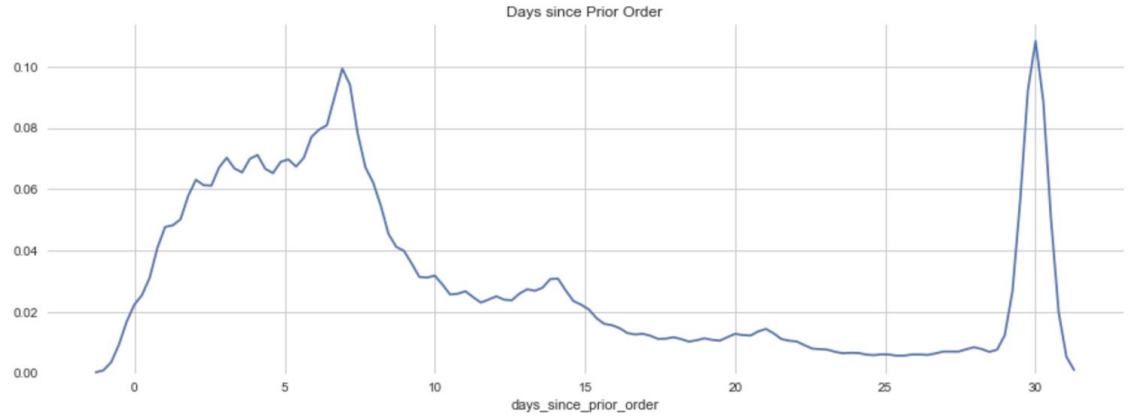

Figure 2. Daily Trend of Days since Prior Order

Generally, it is not the normal distribution on days since prior order, which indicates that there is less relationship between orders and days. On the other hand, there is highly

likely that customers will purchase again after 3 and 7 days or 30 day, based on our Time-series overview on days since prior order figure.

After the basic overview on the data, we decide to merge some datasets into further analysis (Appendix 4). First of all, we merge two splitted datasets into one large dataset named order_sum: train and prior. Then we merge aisle, depart and product name into order_sum. The order_sum consists of over 3000,000 records, which takes a really long time to run statistical analysis or build classification model. Due to extremely large dataset, we decide to randomly select 100,000 from the merged dataset.

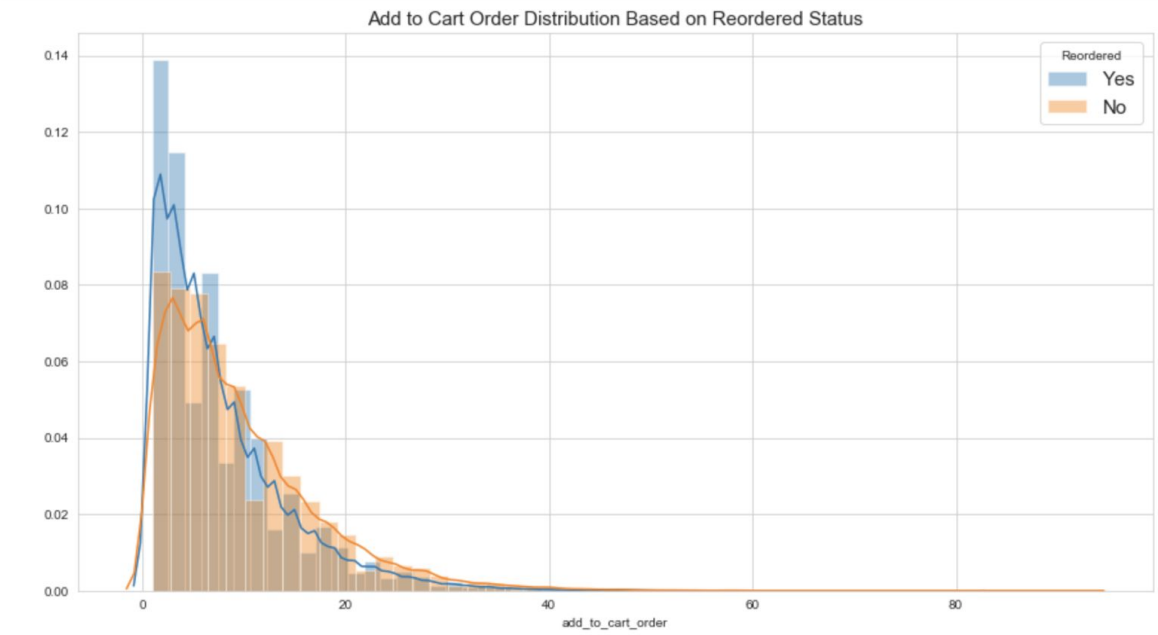

Figure 3. Distribution on Add to Cart Order (Reorder Status Oriented)

The distribution indicates that among the customers who added product to cart, there is a higher chance to make the reorder transaction than those who don't complete the reorder transaction (which refers to 'non-reorder' afterward in this paper). Also, both distribution is right-skewed below 20 add to cart order. The highest order number is over 90 on customers who add products to cart without reorder while the number of reordered customers only reaches around 40 per order.

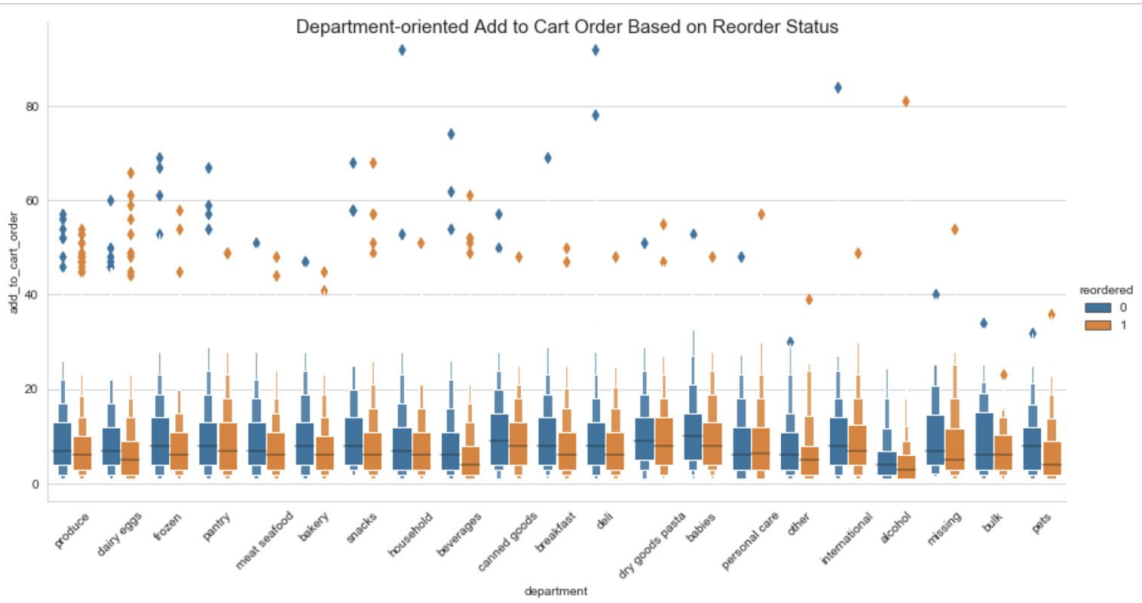

Figure 4. Add to Cart Order Number (Reorder Status & Department-oriented)

Next, we explore deeper by grouping the number based on department. The figure indicates that the add to cart order is normally below 25. To be more specific, the highest outliers for non-reordered are 'deli', 'household' and 'international'. The highest outliers for reordered are 'alcohol', 'daily eggs' and 'snacks'. 'Bables' has relatively higher order number for non-reordered customers.

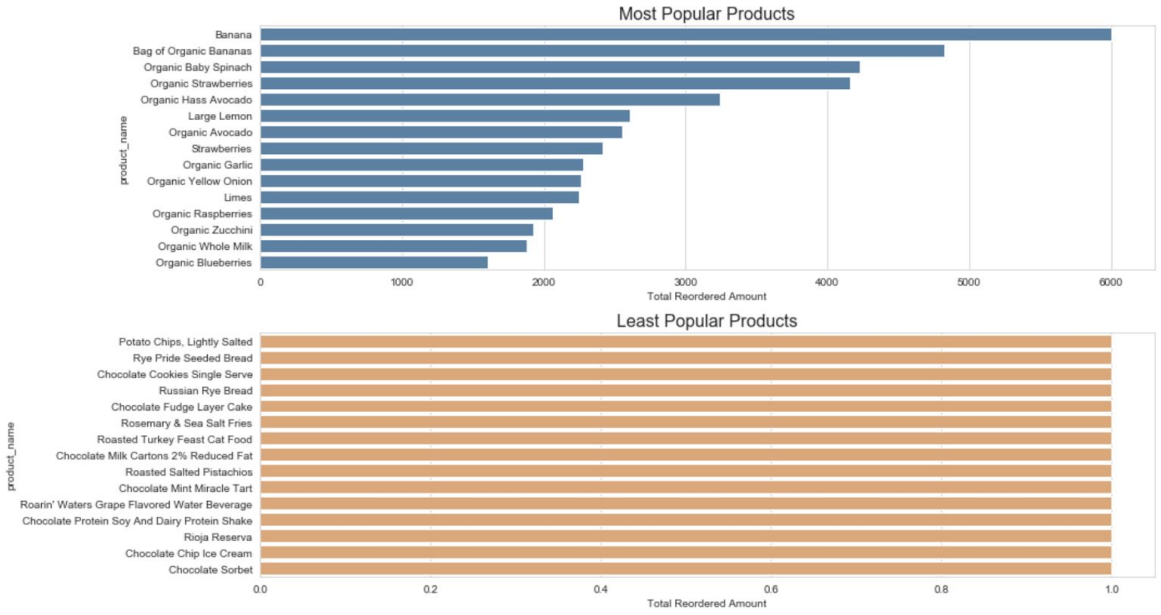

Figure5. Top 10 & Bottom 10 Reordered Products based on Add to Cart Order Number

To get further insight on reorder product, we filter the dataframe only contains reordered products (reordered status equals 1 in the dataframe). Here are the most and least popular reordered products based on add to cart order number. we can see that products with 'Organic' , fruits such as 'Banana' as well as 'Organic Strawberries', and vegetables such as 'Organic Hass Avocado' or 'Organic Baby Spinach' are the most popular for reordered customers while products with high calories such as 'Potato Chips, Lightly Salted', 'Chocolate Chip Ice Cream' and 'Rosemary & Sea Salt Fries' are less likely to be reordered for customers.

Predictive Analysis

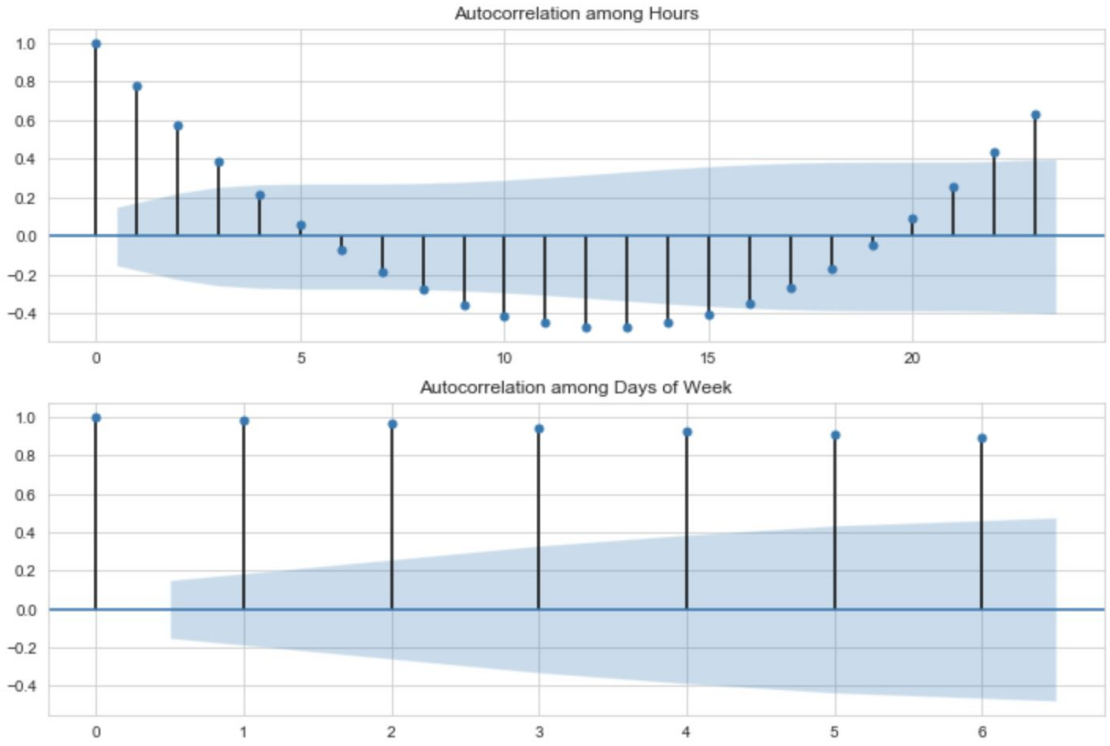

Figure 6. Time-series ACF on Hours & Days of The Week

In this part, we are going to figure out the relationship among the features on time series. Here we are going to plot Autocorrelation Function (ACF) to visualize the auto-correlationship on themselves. ACF plots are heavily used in time series analysis and forecasting.These are plots that graphically summarize the strength of a relationship with an observation in a time series with observations at prior time steps (Brownlee, 2017). We can see the first plot on ACF among Hours that order number is less correlated with each other and there is no statistically significant relationship with a lagged version of itself. However, the second plot on ACF among Days of Week indicates that order number is highly

correlated with each other and there is a statistically significant relationship with a lagged version of itself. Therefore, there is a significant trend pattern on Days of Week, which is a valuable reference to make daily order prediction.

Prescriptive Analysis

Categorical Analysis

Figure 7. WordCloud Comparison on Products (Reorder Status Oriented)

We build word cloud comparison based on reorder status to see the difference between popular products. We can see 'Organic Baby', 'Organic Whole' and 'Milk Organic' are really popular in both segments, which indicates that customers pay more attention on organic products. Other products such as 'Baby Spinach, 'Avocado Organic', 'Bag Organic' are likely to reorder while products such as 'Ice Cream', 'Cheddar Cheese' and 'Organic Red' are only add to cart but not reorder, which indicates that customers prefer healthy diet products rather than high-calorie products.

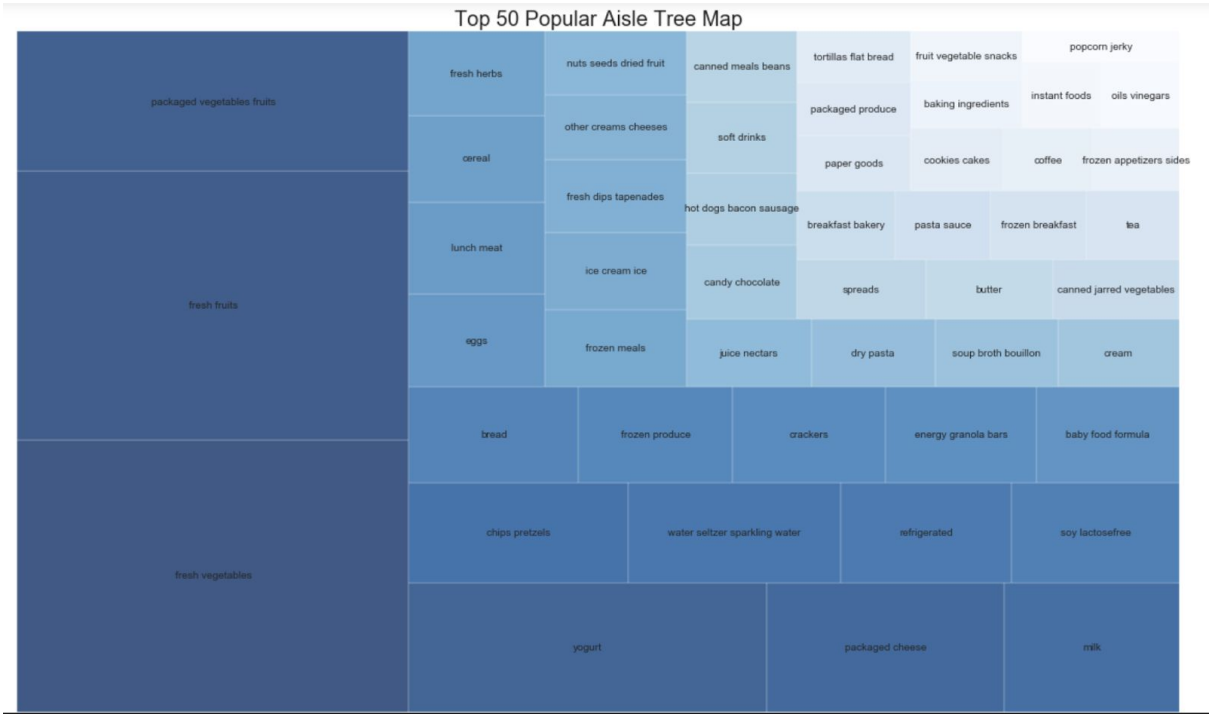

Figure 8. Tree Map on Top 50 Reorder Aisles Based on Add to Cart Order Amount

The above tree map shows that the top 3 reorder aisles based on add to cart order amount are ' fresh vegetable', 'fresh fruits' and ' packaged vegetables fruits'. To be more specific, we are going to explore these three kinds of aisles with the highest order amount.

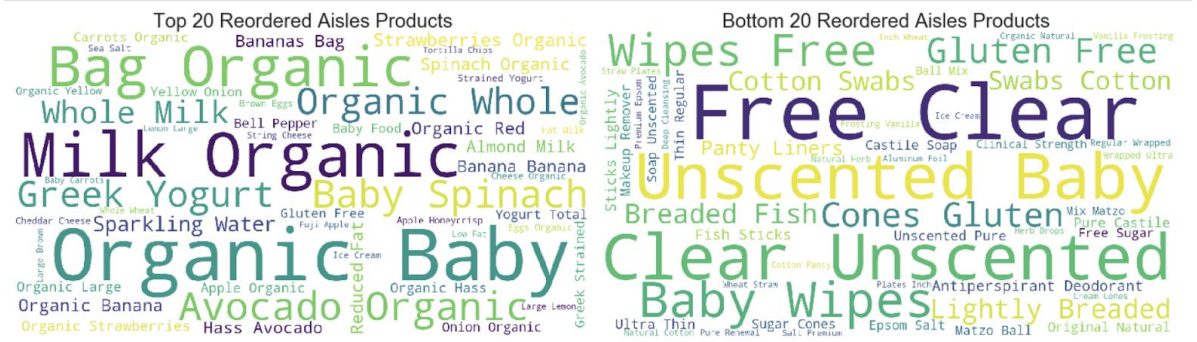

Figure 9. WordCloud Reordered Products Comparison between Top & Bottom 20 Aisles

The first WordCloud shows the top 20 reordered aisles products. We can see that ‘Organic Baby’, ‘Organic Whole’ and ‘Milk Organic’ are the best selling products in popular aisles. The second WordCloud shows the bottom 20 reordered aisles products, and the business strategy should be changed to increase performance for products such as ‘Free Clear’, ‘Unscented Baby’ and ‘Baby Wipes’.

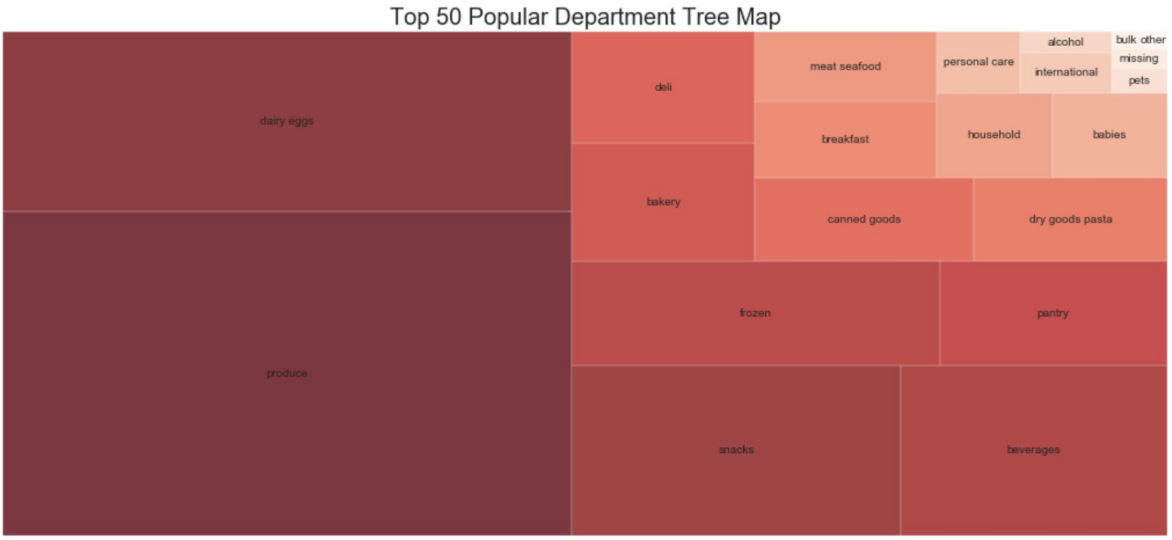

Figure 10. Tree Map on Department Ranking Based on Add to Cart Order Amount

Here the tree map indicates the department ranking based on add to cart order amount. Among others, we can see that 'produce' and 'dairy eggs' are the most popular departments with the highest reorder amount .

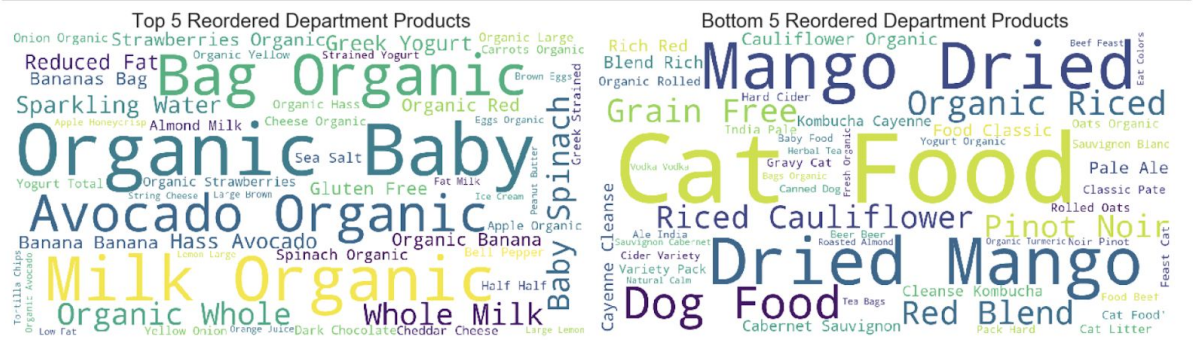

Figure 11. WordCloud Reordered Products Comparison between Top & Bottom 5 Departments

The first WordCloud shows that 'Organic Baby', 'Organic Whole', ‘Organic Whole’ and ‘Avocado Organic’ are the best seller products in popular departments. Products such as ‘Dried Mango’, ‘Cat Food’ and ‘Dog Food’ need to have some special inspection to further promote their sales.

Model Selection

The competition is to predict which products Instacart consumers will buy again based on their customers' historical purchase records, so that when customers need the

products, the supply will be sufficient. Some questions we have to think about relate to the inherent order and timeliness are interesting. For example, how do we measure the time since the customer last bought the item? What are the specific buying patterns related to each customer? Does the purchase day and time important?

To answer these questions, we have to generate some more features that we believe are important to understand what product will purchase again. There are 4 types of features we created: user related features, order related features, product related features and user & product related features. Final data frame that we used for model selection and feature engineering is as follows:

order_id	int32
product_id	int32
user_total_orders	int16
user_total_items	int16
total_distinct_items	int16
user_average_days_between_orders	float32
user_average_basket	float32
order_hour_of_day	int8
days_since_prior_order	float32
days_since_ratio	float32
aisle_id	uint8
department_id	uint8
product_orders	int32
product_reorders	float32
product_reorder_rate	float32
UP_orders	int16
UP_orders_ratio	float32
UP_average_pos_in_cart	float32
UP_reorder_rate	float32
UP_orders_since_last	int16
UP_delta_hour_vs_last	int8

Table 1. Additional Features

Light GBM is mainly used as the model for future prediction. According to Microsoft (2016), LightGBM is a new gradient boosting tree framework, which is highly efficient and scalable and can support many different algorithms including GBDT, GBRT, GBM, and MART. LightGBM is evidenced to be several times faster than existing implementations of gradient boosting trees, due to its fully greedy tree-growth method and histogram-based memory and computation optimization. Before the next step, let's evaluate how is the performance using this model for our dataset. We generated a classification report to measure the quality of predictions from a classification algorithm. The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are

calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes.

The accuracy score of this model reached 0.98 which is very high. AUC score reached 0.61 which represents the probability that a random positive reorder rate is positioned to the right of a random negative reorder rates.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	2529347
1	0.15	0.23	0.18	13038
micro avg	0.99	0.99	0.99	2542385
macro avg	0.57	0.61	0.59	2542385
weighted avg	0.99	0.99	0.99	2542385

Table 2. Confusion Report

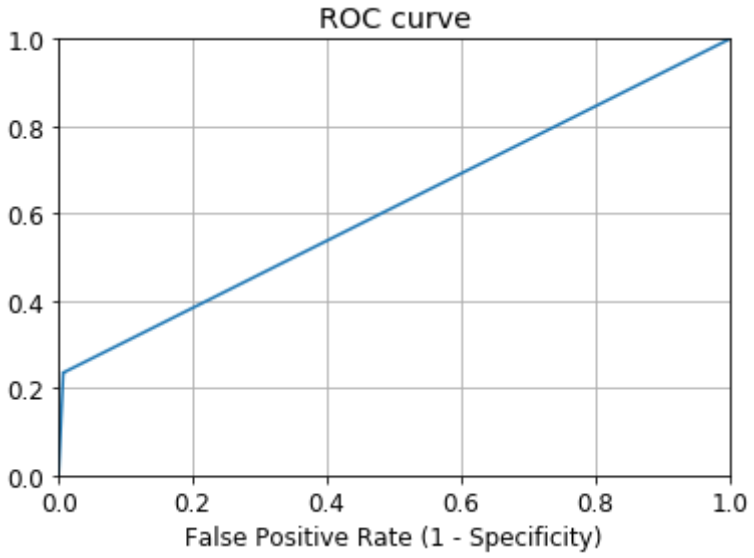

Figure 12. ROC Curve

Feature Engineering

From the confusion matrix, we are able to correctly predict no-reorder rate of 2,511,889 transactions and 3,062 reorder transactions.

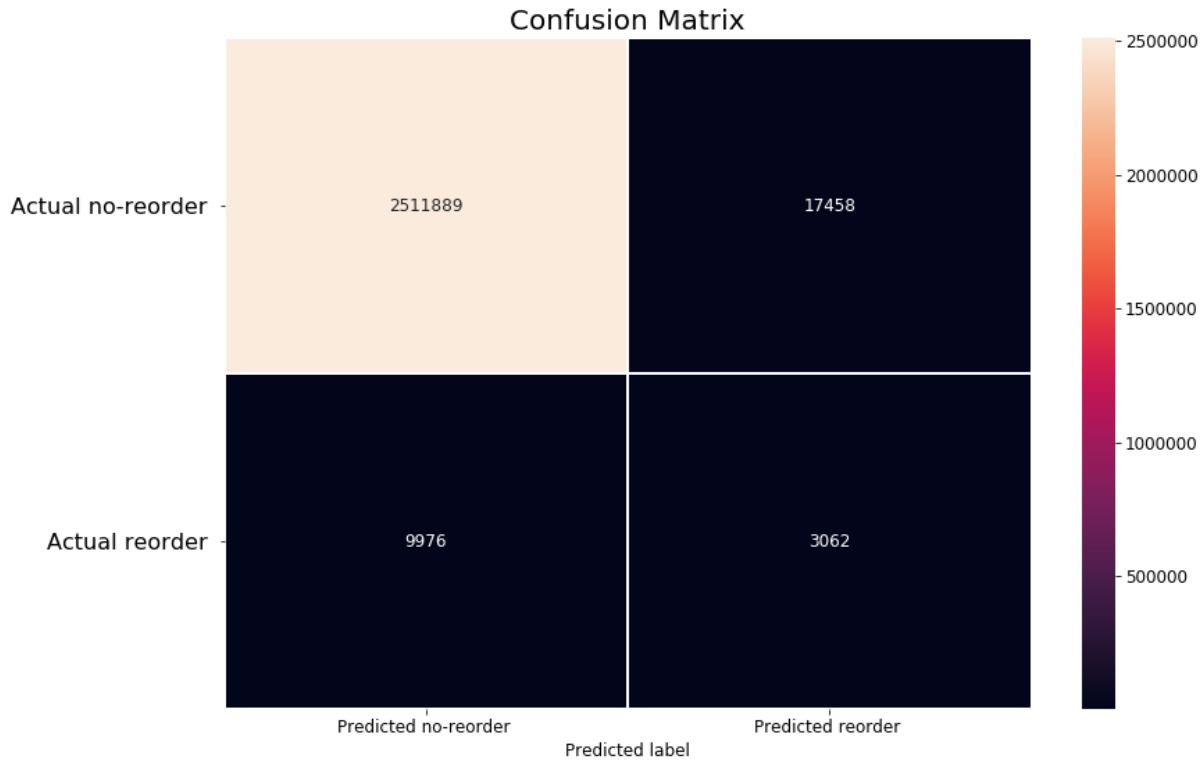

Figure 13. Confusion Matrix

Important features are selected by the Light GBM model as follows. The Top 5 important features are product_id, User & Product orders ratio, order_id, user total orders and user total item. Refer to above EDA analysis, we can make some suggestions to enhance prediction in the future.

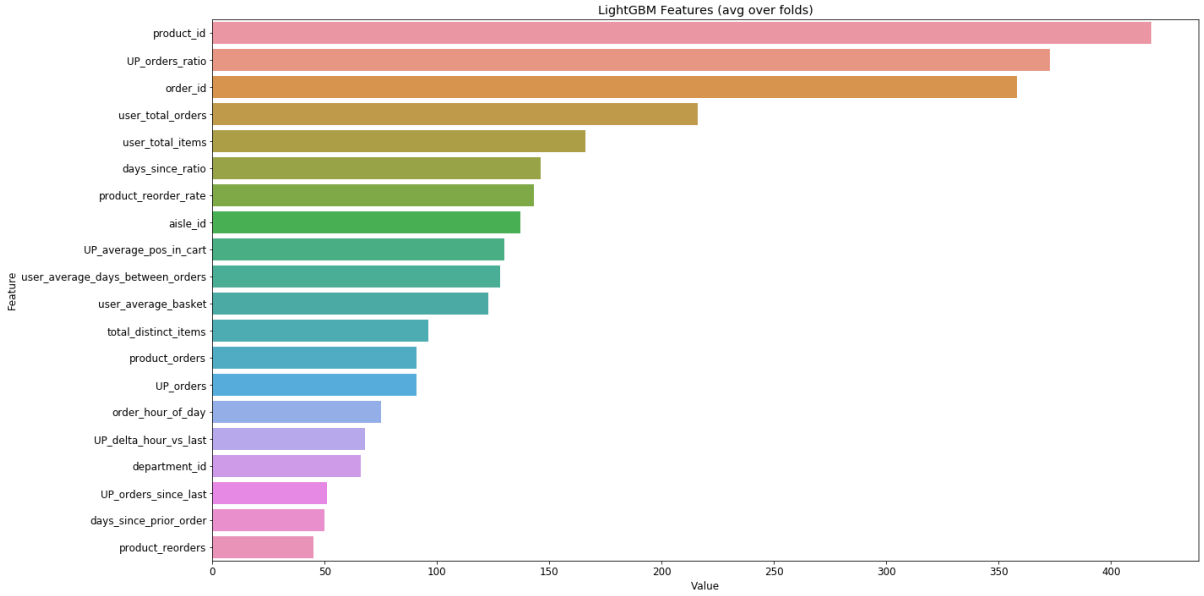

Figure 14. Feature Selection

Conclusion

This paper provides a deep analysis of various factors that can answer our question: which products will an Instacart consumer purchase again. We examined the whole dataset and did various analysis to better understand the situation. We also analyzed the top best sales product and bottom worst sale product based on department, aisles, add to cart frequency and recorded rate. Besides that basic features, we also added other features that are related to our prediction. We decided to go with Light GBM model since it is highly efficient and scalable and can support many different algorithms. The features are ranked from most important to least important to our prediction. We were also able to leverage past customer behavior to make prediction on the next reorder transaction. This can give us a better perspective on which product will be ordered next time. The inventory and supply system can thus take action to take informed decision on inventory planning.

Reference

Blasco Lopez, M. F., Recuero Virto, N., & San-Martín, S. (2018). Local Food Shopping: Factors Affecting Users' Behavioural E-Loyalty. Administrative Sciences (2076-3387), 8(3), 47. Retrieved from https://doi.org/10.3390/admsci8030047

Brownlee, J. (2017, February 6). A Gentle Introduction to Autocorrelation and Partial Autocorrelation. Retrieved from https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

Dow Jones Institutional News. (2019, March 24). Amazon to whole foods online delivery customers: We're out of celery, how's kale? Retrieved from https://search.proquest.com/docview/2196329058?accountid=25283

Haddon, H. (2019, March 11). 'I'm Addicted'; Why Food-Delivery Companies Want to Create Superusers. Wall Street Journal - Online Edition, p. 1. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=bth&AN=135192068&site=ehost-live&scope=site

Instacart.com. Retrieved from https://www.instacart.com/

Microsoft. (2016, October 1). LightGBM. Retrieved from https://www.microsoft.com/en-us/research/project/lightgbm/

Appendix

Appendix 1: Processing Data_1

Preprocessing datasets

In [116]:
```
# importing neccessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [117]:
```
# loading datasets
department = pd.read_csv('departments.csv')
aisles = pd.read_csv('aisles.csv')
sample = pd.read_csv('sample_submission.csv')
product = pd.read_csv('products.csv')
train = pd.read_csv('order_products__train.csv')
order = pd.read_csv('orders.csv')
prior = pd.read_csv('order_products__prior.csv')
```

In [118]:
```
# grouping all datasets into a list
dfs = [department, aisles, sample, product, train, order, prior]

# naming all datasets
df_names = ['department', 'aisles', 'sample', 'product', 'train', 'order', 'prior']

# zipping named datasets and dataframes
for (name, df) in zip(df_names, dfs):
    """checking NaNs among all datasets and printing the results"""
    print('The null value sum of dataframe: ' + name)
    display(df.isnull().sum())
```

```
The null value sum of dataframe: department

department_id    0
department       0
dtype: int64

The null value sum of dataframe: aisles

aisle_id    0
aisle       0
dtype: int64
```

Appendix 2: Processing Data_2

In [119]:
```
# zipping named datasets and dataframes
for (name, df) in zip(df_names, dfs):
    """previewing all datasets"""
    print('Preview of the dataframe: ' + name)
    display(df.head())
```

```
Preview of the dataframe: department
```

	department_id	department
0	1	frozen
1	2	other
2	3	bakery
3	4	produce
4	5	alcohol

```
Preview of the dataframe: aisles
```

	aisle_id	aisle
0	1	prepared soups salads
1	2	specialty cheeses
2	3	energy granola bars
3	4	instant foods
4	5	marinades meat preparation

```
Preview of the dataframe: sample
```

	order_id	products
0	17	39276 29259
1	34	39276 29259
2	137	39276 29259
3	182	39276 29259

Appendix 3: Processing Data_3

In [5]:
```
# description about datasets
for (name, df) in zip(df_names, dfs):
    """previewing all datasets"""
    print('Preview of the dataframe: ' + name)
    display(df.describe())
```

```
Preview of the dataframe: department
```

	department_id
count	21.000000
mean	11.000000
std	6.204837
min	1.000000
25%	6.000000
50%	11.000000
75%	16.000000
max	21.000000

```
Preview of the dataframe: aisles
```

	aisle_id
count	134.000000
mean	67.500000
std	38.826537
min	1.000000

Appendix 4. Data Manipulation

In [123]:
```
# combining two datasets together as a whole part
order_sum = pd.concat([train, prior], axis = 0)

# randomly selectint 100000 samples for further analysis (due to extreme large amount of data)
order_sample = order_sum.sample(n = 100000, random_state = 0).reset_index(drop = True)

# merging other categorical datasets into one dataset
merged_product = pd.merge(order_sample, product, on = 'product_id')
merged_aisles = pd.merge(merged_product, aisles, on = 'aisle_id')
sample_df = pd.merge(merged_aisles, department, on = 'department_id')

# dropping duplicated columns (just remaining categorical variables instead of categorical 'id's)
sample_df = sample_df.drop(['product_id', 'aisle_id', 'department_id'], axis = 1)

# previewing the final dataset
sample_df.head()
```

Out[123]:

	order_id	add_to_cart_order	reordered	product_name	aisle	department
0	127212	4	1	Organic Baby Spinach	packaged vegetables fruits	produce
1	1316042	9	1	Organic Baby Spinach	packaged vegetables fruits	produce
2	1769655	2	1	Organic Baby Spinach	packaged vegetables fruits	produce
3	2781831	8	1	Organic Baby Spinach	packaged vegetables fruits	produce
4	1686819	2	1	Organic Baby Spinach	packaged vegetables fruits	produce

Appendix 5. Time-series Overview on Days of The Week & Hours

In [120]:
```
# grouping order by order day of week and hour of day
dow_df = order.groupby(['order_dow', 'order_hour_of_day'])['order_number'].count().reset_index
()

# plotting grouped total order trend based on day of week and hour of day
sns.set_style('whitegrid')
plt.figure(figsize = (20, 8))
g = sns.lineplot(x = 'order_hour_of_day', y = 'order_number', hue = 'order_dow', data = dow_d
f, palette = 'Blues')
g.set(title = 'Hourly Total Order', xlabel = 'Order Hour', ylabel = 'Order Amount')
plt.show()
```

Appendix 6. Daily Trend of Days since Prior Order

```
In [122]: # plotting time trend based on days since prior order
          plt.figure(figsize = (15, 5))
          sns.distplot(order.days_since_prior_order.dropna(), hist = False).set_title('Days since Prior
          Order')
          plt.show()
```

Appendix 7. Distribution on Add to Cart Order (Reorder Status Oriented)

```
In [122]: # plotting time trend based on days since prior order
          plt.figure(figsize = (15, 5))
          sns.distplot(order.days_since_prior_order.dropna(), hist = False).set_titl
          e('Days since Prior Order')
          plt.show()
```


Appendix 8. Add to Cart Order Number (Reorder Status & Department-oriented)

```
In [125]: # plottting the department-oriented add to cart total order based on reord
          ered order and non-reordered order
          g = sns.catplot(x = 'department', y = 'add_to_cart_order', hue = 'reordere
          d',\
                          data = sample_df, kind = 'boxen', height = 6, aspect = 2)
          g.set_xticklabels(rotation = 45)
          plt.show()
```

Appendix 9. Top 10 & Bottom 10 Reordered Products based on Add to Cart Order Number

```
In [126]: from wordcloud import WordCloud
          from nltk.tokenize import word_tokenize

          # multiplying add to cart order by tokenized product name (generating word
          frequency)
          sample_df['product_words'] = sample_df.product_name.apply(word_tokenize) *
          sample_df.add_to_cart_order

          # seperating the dataframe into two dataframes: reordered & non-reordered
          reordered_df = sample_df[sample_df.reordered == 1]
          nonreordered_df = sample_df[sample_df.reordered == 0]

          # selecting the top & bottom 15 product names based on add to cart order
          top_15_product = reordered_df.groupby('product_name').add_to_cart_order.su
          m().sort_values(ascending = False).head(15).reset_index()
          bottom_15_product = reordered_df.groupby('product_name').add_to_cart_orde
          r.sum().sort_values().head(15).reset_index()

          # subplotting the results for two plots
          fig, axes = plt.subplots(2, 1, figsize = (15, 8))
          sns.set_color_codes("pastel")

          sns.barplot(x = 'add_to_cart_order', y = 'product_name', data = top_15_pro
          duct, ax = axes[0], color = 'b')
          axes[0].set_title('Most Popular Products', fontsize = 16)
          axes[0].set_xlabel('Total Reordered Amount')

          sns.barplot(x = 'add_to_cart_order', y = 'product_name', data = bottom_15_
          product, ax = axes[1], color = 'r')
          axes[1].set_title('Least Popular Products', fontsize = 16)
          axes[1].set_xlabel('Total Reordered Amount')

          plt.tight_layout()
          plt.show()
```

Appendix 10. Time-series ACF on Hours & Days of The Week

In [121]:
```
from statsmodels.graphics import tsaplots

# plotting autocorrelation function based on hour of day and day of week
fig, axes = plt.subplots(2, 1, figsize = (12, 8))
tsaplots.plot_acf(dow_df.order_hour_of_day, lags = 23, title = 'Autocorrel
ation among Hours', ax = axes[0])
tsaplots.plot_acf(dow_df.order_dow, lags = 6, title = 'Autocorrelation amo
ng Days of Week', ax = axes[1])
plt.show()
```

Appendix 11. Word Cloud Comparison on Products (Reorder Status Oriented)

In [127]:
```
# defining a function to generate wordcloud
def wordcloud(df):
    """returning the wordcloud based on product names' frequency"""
    text = [' '.join(word) for word in df['product_words']]
    # limitting the size, background color and maximum words for the wordc
loud
    wordcloud = WordCloud(width = 1500, height = 800, max_words = 50, back
ground_color = 'white').generate(str(text))
    return wordcloud

# applying the wordcloud function into the dataframes
reordered_wc = wordcloud(reordered_df)
cart_no_order_wc = wordcloud(nonreordered_df)

# plotting the comparison plots between reordered product name frequency a
nd non-reordered product name frequency
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 8))
ax1.imshow(reordered_wc)
ax1.set_title('Reordered Products', fontsize = 24)
ax1.axis('off')

ax2.set_title('Add to Cart without Reordered Products', fontsize = 24)
ax2.imshow(cart_no_order_wc)
ax2.axis('off')

plt.tight_layout()
plt.show()
```

Appendix 12. Tree Map on Top 50 Reorder Aisles Based on Add to Cart Order Amount

In [128]:
```
import squarify
import matplotlib

# grouping and sorting the total add to cart order based on aisle and sele
cting top 50 aisles
tree_df_aisle = reordered_df.groupby('aisle')['add_to_cart_order'].sum().s
ort_values(ascending = False).reset_index().head(50)

# setting color and ranges for the plot
aisle_values = [i for i in range(len(tree_df_aisle.aisle))]
cmap = matplotlib.cm.Blues
norm = matplotlib.colors.Normalize(vmin = min(aisle_values), vmax = max(ai
sle_values))
colors = [cmap(norm(value)) for value in aisle_values][::-1]

# plotting tree map
plt.figure(figsize = (20, 12))
squarify.plot(sizes = tree_df_aisle.add_to_cart_order, label = tree_df_ais
le.aisle, alpha = .8, color = colors)
plt.title('Top 50 Popular Aisle Tree Map', fontsize = 20)
plt.axis('off')
plt.show()
```

Appendix 13. WordCloud Reordered Products Comparison between Top & Bottom 20 Aisles

In [129]:
```
# selecting top 20 aisles based on reorders
pop_reordered_aisle = reordered_df.groupby('aisle').reordered.sum().sort_v
alues(ascending = False).head(20)
# getting the the name list of the aisle
pop_aisle_list = list(pop_reordered_aisle.index.get_level_values(0))
# filtering the data based on the listed aisles
pop_aisle_df = reordered_df[reordered_df.aisle.isin(pop_aisle_list)]

# selecting leaset 20 aisles based on reorders
bottom_reordered_aisle = reordered_df.groupby('aisle').reordered.sum().sor
t_values().head(20)
# getting the the name list of the aisle
bottom_aisle_list = list(bottom_reordered_aisle.index.get_level_values(0))
# filtering the data based on the listed aisles
bottom_aisle_df = reordered_df[reordered_df.aisle.isin(bottom_aisle_list)]

# generating wordcloud based on defined function
pop_aisle_wc = wordcloud(pop_aisle_df)
bottom_aisle_wc = wordcloud(bottom_aisle_df)

# plotting the comparison betwwen top 20 and bottom 20 reordered product n
ames based on aisles
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 8))
ax1.imshow(pop_aisle_wc)
ax1.set_title('Top 20 Reordered Aisles Products', fontsize = 24)
ax1.axis('off')

ax2.set_title('Bottom 20 Reordered Aisles Products', fontsize = 24)
ax2.imshow(bottom_aisle_wc)
ax2.axis('off')

plt.tight_layout()
plt.show()
```

Appendix 14. Tree Map on Top 50 Reorder Aisles Based on Add to Cart Order Amount

In [130]:
```
# grouping and sorting the total add to cart order based on department
tree_df_department = reordered_df.groupby('department')['add_to_cart_orde
r'].sum().sort_values(ascending = False).reset_index()

# setting color and ranges for the plot
department_values = [i for i in range(len(tree_df_department.department))]
cmap = matplotlib.cm.Reds
norm = matplotlib.colors.Normalize(vmin = min(department_values), vmax = m
ax(department_values))
colors = [cmap(norm(value)) for value in department_values][::-1]

# plotting tree map
plt.figure(figsize = (18, 8))
squarify.plot(sizes = tree_df_department.add_to_cart_order, label = tree_d
f_department.department, alpha = .8, color = colors)
plt.title('Top 50 Popular Department Tree Map', fontsize = 20)
plt.axis('off')
plt.show()
```

Appendix 15. WordCloud Reordered Products Comparison between Top & Bottom 5 Departments.

In [131]:
```
# filtering the data based on the top 5 departments based on total reorder
ed amount
pop_reordered_dpm = reordered_df.groupby('department').reordered.sum().sor
t_values(ascending = False).head(5)
pop_dpm_list = list(pop_reordered_dpm.index.get_level_values(0))
pop_dpm_df = reordered_df[reordered_df.department.isin(pop_dpm_list)]

# filtering the data based on the bottom 5 departments based on total reor
dered amount
bottom_reordered_dpm = reordered_df.groupby('department').reordered.sum().
sort_values().head(5)
bottom_dpm_list = list(bottom_reordered_dpm.index.get_level_values(0))
bottom_dpm_df = reordered_df[reordered_df.department.isin(bottom_dpm_lis
t)]

# generating product name's wordcloud basd on filted data
pop_dpm_wc = wordcloud(pop_dpm_df)
bottom_dpm_wc = wordcloud(bottom_dpm_df)

# plotting comparison between top 5 and bottom 5 reordered product names b
ased on departments
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 8))
ax1.imshow(pop_dpm_wc)
ax1.set_title('Top 5 Reordered Department Products', fontsize = 24)
ax1.axis('off')

ax2.set_title('Bottom 5 Reordered Department Products', fontsize = 24)
ax2.imshow(bottom_dpm_wc)
ax2.axis('off')

plt.tight_layout()
plt.show()
```

Appendix 16. Add Features

```
print('user related features')
df['user_id'] = df.order_id.map(orders.user_id)
df['user_total_orders'] = df.user_id.map(users.nb_orders)
df['user_total_items'] = df.user_id.map(users.total_items)
df['total_distinct_items'] = df.user_id.map(users.total_distinct_items)
df['user_average_days_between_orders'] = df.user_id.map(users.average_days_between_orders)
df['user_average_basket'] =  df.user_id.map(users.average_basket)

print('order related features')
# df['dow'] = df.order_id.map(orders.order_dow)
df['order_hour_of_day'] = df.order_id.map(orders.order_hour_of_day)
df['days_since_prior_order'] = df.order_id.map(orders.days_since_prior_order)
df['days_since_ratio'] = df.days_since_prior_order / df.user_average_days_between_orders

print('product related features')
df['aisle_id'] = df.product_id.map(products.aisle_id)
df['department_id'] = df.product_id.map(products.department_id)
df['product_orders'] = df.product_id.map(products.orders).astype(np.int32)
df['product_reorders'] = df.product_id.map(products.reorders)
df['product_reorder_rate'] = df.product_id.map(products.reorder_rate)

print('user_X_product related features')
df['z'] = df.user_id * 100000 + df.product_id
df.drop(['user_id'], axis=1, inplace=True)
df['UP_orders'] = df.z.map(userXproduct.nb_orders)
df['UP_orders_ratio'] = (df.UP_orders / df.user_total_orders).astype(np.float32)
df['UP_last_order_id'] = df.z.map(userXproduct.last_order_id)
df['UP_average_pos_in_cart'] = (df.z.map(userXproduct.sum_pos_in_cart) / df.UP_orders).asty
df['UP_reorder_rate'] = (df.UP_orders / df.user_total_orders).astype(np.float32)
df['UP_orders_since_last'] = df.user_total_orders - df.UP_last_order_id.map(orders.order_nu
df['UP_delta_hour_vs_last'] = abs(df.order_hour_of_day - df.UP_last_order_id.map(orders.ord
```

Appendix 17. Accuracy Report

In [198]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

lgb_model = LGBMClassifier()
lgb_model.fit(X_train, y_train)
print(accuracy_score(y_test, lgb_model.predict(X_test)))
print(classification_report(y_test, lgb_model.predict(X_test)))
```

Appendix 18. ROC Curve

In [211]:
```
y_pred_prob = lgb_model.predict_proba(X_test)[:, 1]

y_pred_prob

fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.grid(True)
```

Appendix 19. Confusion Matrix

In [216]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_matrix = confusion_matrix(y_test, y_pred)
f, ax = plt.subplots(figsize=(12, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", linewidths=.5, ax=ax)
plt.title("Confusion Matrix", fontsize=20)
plt.subplots_adjust(left=0.15, right=0.99, bottom=0.15, top=0.99)
ax.set_xticklabels(['Predicted no-reorder ', 'Predicted reorder '])
ax.set_yticklabels(['Actual no-reorder ', 'Actual reorder '], fontsize=16, rotation=360)
ax.set_yticks(np.arange(conf_matrix.shape[0]) + 0.5, minor=False)
plt.xlabel('Predicted label')
plt.show()
```

Appendix 20. Feature Selection

In [210]:
```
feature_imp = pd.DataFrame(sorted(zip(lgb_model.feature_importances_,X.columns)), columns=['Val

plt.figure(figsize=(20, 10))
sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascending=False))
plt.title('LightGBM Features (avg over folds)')
plt.tight_layout()
plt.show()
plt.savefig('lgbm_importances-01.png')
```