Instacart Market Basket Analysis

DTSA 5510

November 24, 2024

Data Challenge Introduction

The contest is sponsored by Instacart (Maplebear Inc.) 

The data challenge is to predict which products will be in a user's next order.

The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, there is between 4 and 100 of their orders, with the sequence of products purchased in each order. It also provide the week and hour of day the order was placed, and a relative measure of time between orders.

The data challenge is available :*https://www.kaggle.com/competitions/instacart-market-basket-analysis/*

My GitHub is available: *https://github.com/lzheng01/Instacart-Market-Basket-Analysis*

Analysis Proposal

Explore this dataset and build a recommender system for users to predict their next purchase using unsupervised machine learning algorithms.

- Apriori Algorithm

- Frequent Pattern Growth Algorithm

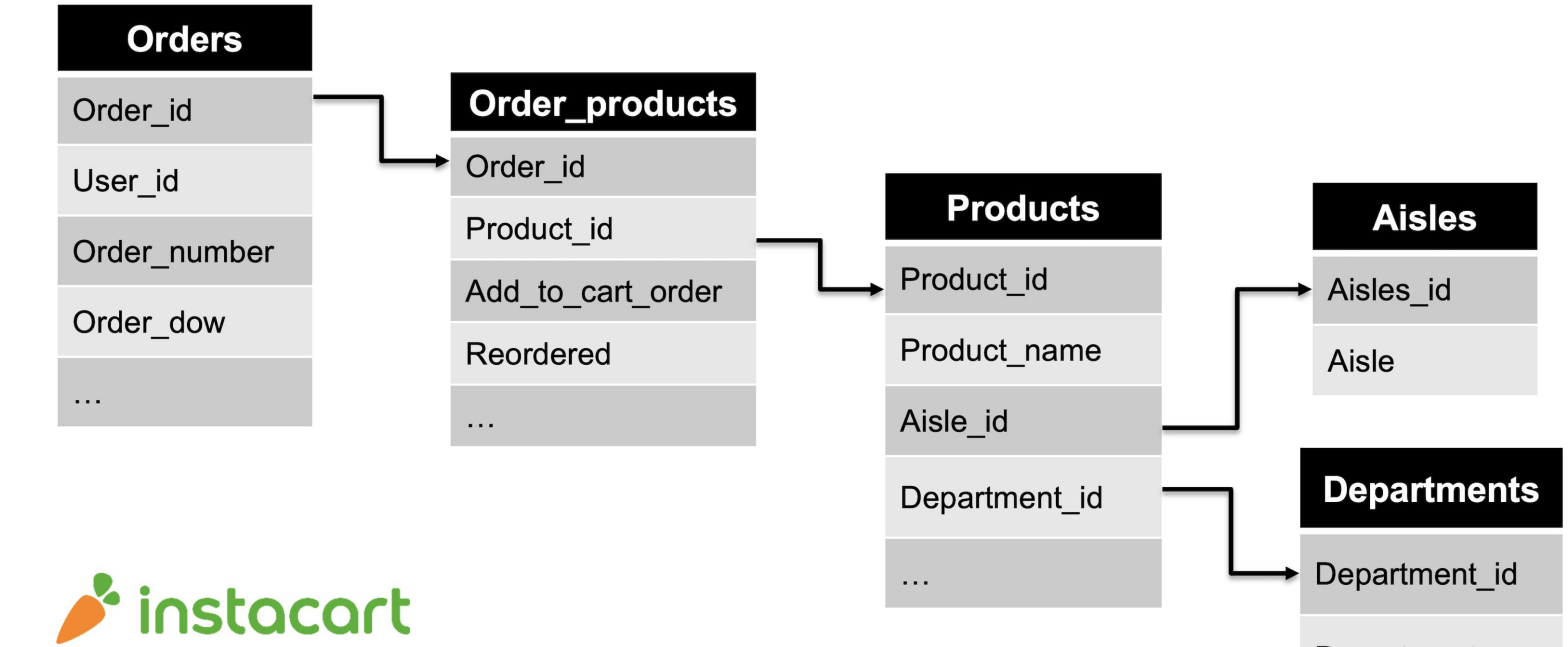
Data Overview

This data challenge contains 6 different csv files, more than 3.4 million orders, and over 4000 different products.

Data Explorer

205.77 MB

- aisles.csv.zip
- departments.csv.zip
- order_products__prior.csv.zip
- order_products__train.csv.zip
- orders.csv.zip
- products.csv.zip



Exploartory Data Analysis

- **Check missing values and duplicates for Orders dataset**
- **Combine Orders and Order_products data**
- **Combine Products, Aisles and Departments data**

```
[24] # Check for NUll value in orders
     orders.isnull().sum()
```

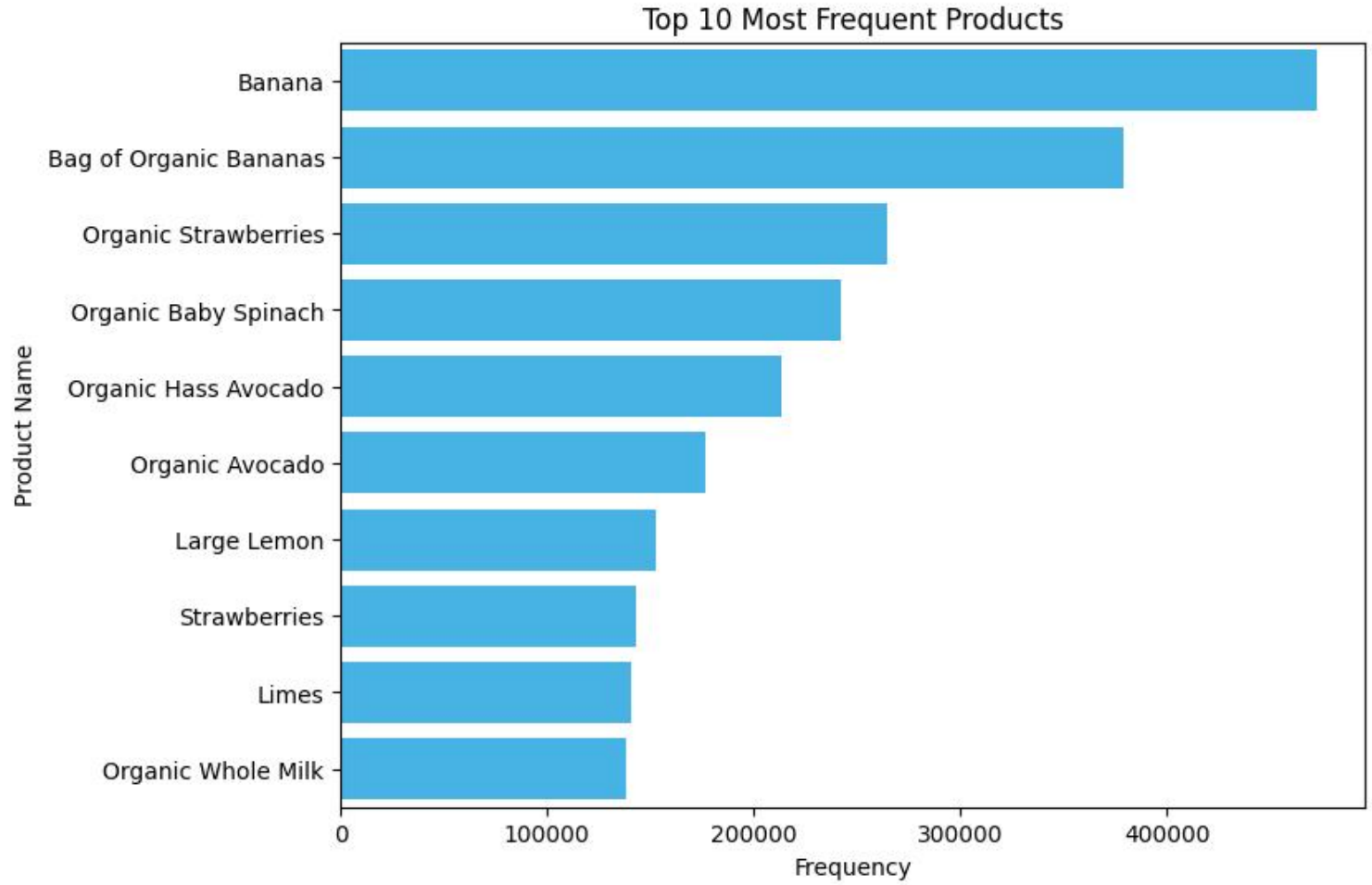
	0
order_id	0
user_id	0
eval_set	0
order_number	0
order_dow	0
order_hour_of_day	0
days_since_prior_order	206209

dtype: int64

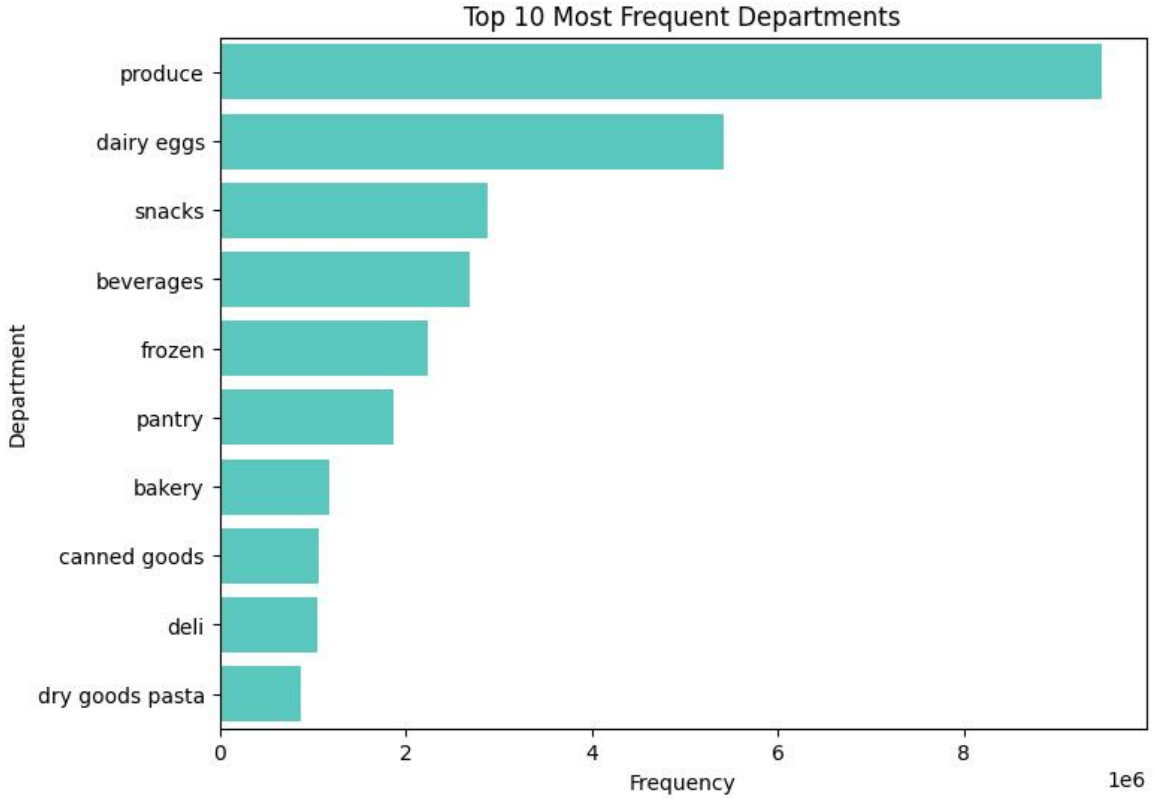
```
[25] # Check for duplicated values
     orders.duplicated().sum()
```

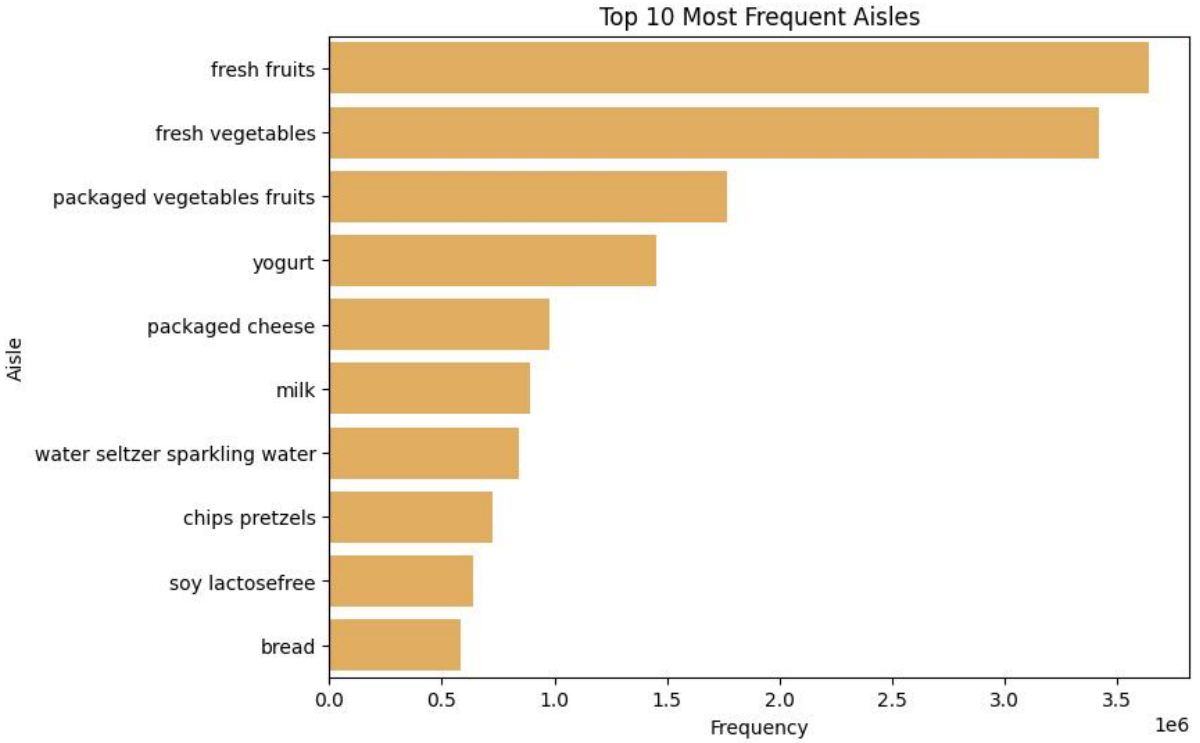
```
0
```

Exploartory Data Analysis



Exploartory Data Analysis





Exploartory Data Analysis

- Select the most Frequent Product and most Loyal Customer for the training dataset

```
[33] # Select the records which fall in the 95% confidence interval of the most frequent product
     low_conf, up_conf = sms.DescrStatsW(df_order["product_id"].value_counts()).tconfint_mean()
     important_products = df_order["product_id"].value_counts()[df_order["product_id"].value_counts() > low_conf].index
     df_order = df_order[df_order["product_id"].isin(important_products)]
```

```
[34] # Select the records which fall in the 95% confidence interval of the most loyalful customer
     df_order["user_id"].value_counts()
     low_conf, up_conf = sms.DescrStatsW(df_order["user_id"].value_counts()).tconfint_mean()
     important_baskets = df_order["user_id"].value_counts()[df_order["user_id"].value_counts() > low_conf].index
     df_order = df_order[df_order["user_id"].isin(important_baskets)]
```

Model Building

Apriori Algorithm

```
[ ]  random_product = rules.sample(1,random_state=45)["antecedents"].explode().iloc[0]
     recommended_products = arl_recommender(rules,random_product, 2)

     print("Random picked product:", random_product)
     print("Recommended products based on current cart:", get_product_names(recommended_products))
```

```
Random picked product: 47626
Recommended products based on current cart: {26209: 'Limes', 21903: 'Organic Baby Spinach'}
```

FP-Growth Algorithm

```
[ ]  current_cart = ['47626']
     print('Current cart:', get_product_names(current_cart))
     recommended_products = recommend_products(current_cart, rules)
     print("Recommended products based on current cart:", get_product_names(recommended_products))
```

```
Current cart: {'47626': 'Large Lemon'}
Recommended products based on current cart: {26209: 'Limes', 24852: 'Banana'}
```

Model Performance Comparison

- **Excution time:**

```
Apriori Execution Time: 25.9413 seconds
FP-growth Execution Time: 48.6568 seconds
```

- **Memory Usage:**

```
Apriori Memory Usage: 24749.4140625 MB
FP-growth Memory Usage: 49544.30078125 MB
```

Conclusions and Discussion

Results:

- Recommended products similar
- Apriori executed faster than FP-Growth
- Apriori took less memory compared with FP-Growth
- Imbalanced data may limit FP-growth effectiveness

Possible Improvements:

Combine FP-Growth with other recommender algrithm like collaborative filtering or matrix factorization.

Thank You!