SAN JOSÉ STATE UNIVERSITY

Groceries Recommendation System

CMPE 256 - Large Scale Analytics

Summer 2019

Individual Project Report

Miguel Covarrubias <miguel.covarrubias@sjsu.edu> - 008809892

Table of Contents

Abstract

Having a recommendation system is now a necessity for businesses to succeed in this market. Ecommerce has been increasing very rapidly over the past years because it is easier for people to buy items from their electronic devices while having the items delivered anywhere. The online grocery shopping sales is a very good example of an industry that is growing year by year. Business such as Walmart, Safeway, Instacart etc. are well aware that this is happening and they are building better recommendations systems to improve the online customer experience and to increase their profits. Instacart for example has a recommendation model that when the user selects an item or a series of items it will suggest other items to the user based on the term “often bought with”. This will clearly improve their sales and it will create a good customer service experience because people will discover new products. Now, how are recommendations systems build to support these use cases? This report will go over the process.

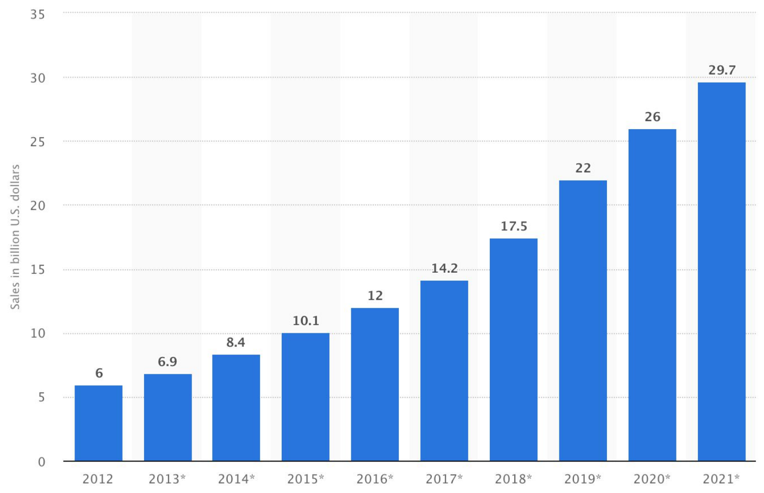


Figure 1: Online grocery shopping sales in the United States from 2012 to 2021 (in billion U.S. dollars)

Introduction

My inspiration for building a recommendation system started in the kitchen. From breakfast, lunch and dinner cooking is one of my passions and I'm always looking for ways to incorporate new items into my dishes. This means that building a recommendation system that helps people discover new related/similar grocery products would be very useful to me. The work I will do in this project will use a modelling technique called "Market Basket Analysis" which is based on the theory that if you buy a certain group of items, you are more likely to buy another group of items. For example, if a person selects a pack or eggs the recommendations engine will probably suggest to buy sausage links or white bread according to other shoppers that also bought eggs along with sausage links. For this project, I will use a very large dataset from the Instacart's website order transactions. What is Instacart? Instacart is one of the bigger online grocery stores that allows you to order from shops in your neighborhood (usually at the same prices). You place an order online, then someone shops for those items and delivers it to you the same day, usually in two hours or less.This dataset from Instacart needs several steps of preprocessing before it can be used in the model as an input. The main algorithm for my model will be the "Association Rule Mining". Association Rule Mining is one of the ways to find patterns in data. It helps us find the features (dimensions) which occur together or that are "correlated" such as the example mentioned above.

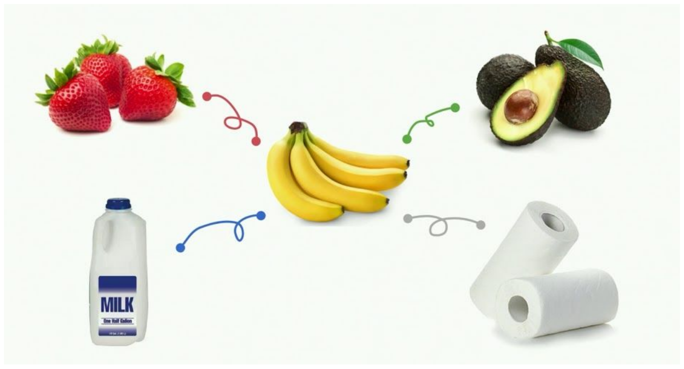


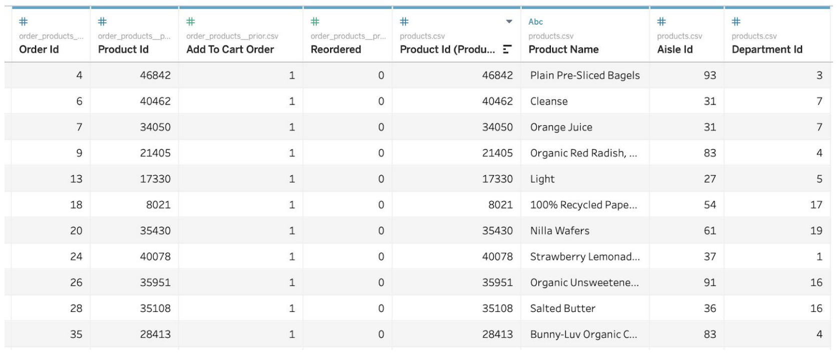
Figure 2: Sample Recommendation Bundles

Dataset Adquisicion and Processing

The data needed for this recommendation system needs to be large enough to capture the frequency of items bought together. The main dataset used in this project is from Instacart. Instacart has an open dataset of 3 million online user orders. The dataset contains many details about the orders and products such as the hour of the day the order was placed on, days since the last order ect. However, I will only focus on the orders and the type of products in the order.

Data Preparation:

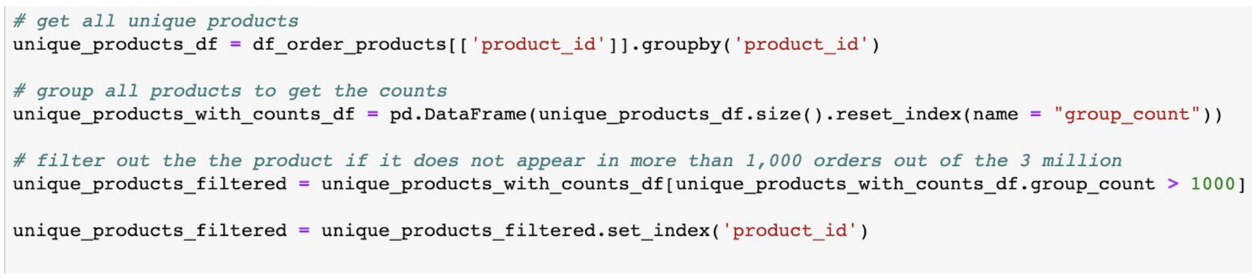
There are two main datasets the orders.csv and the products.csv. The orders.csv is the largest dataset and it has multiple rows per *order id* because each *order id* has multiple *product ids* that are part of that order. The product id is just a number and we need to join the orders dataset with the products dataset in order to get the real name of the product. This will help us better visualize our recommendations. Figure 3 shows the data between orders and products joined into one. This will be the initial transformation of the data.



order_products_... Order Id	order_products__p... Product Id	order_products__prior.csv Add To Cart Order	order_products__pr... Reordered	products.csv Product Id (Produ...	products.csv Product Name	products.csv Aisle Id	products.csv Department Id
4	46842	1	0	46842	Plain Pre-Sliced Bagels	93	3
6	40462	1	0	40462	Cleanse	31	7
7	34050	1	0	34050	Orange Juice	31	7
9	21405	1	0	21405	Organic Red Radish, ...	83	4
13	17330	1	0	17330	Light	27	5
18	8021	1	0	8021	100% Recycled Pape...	54	17
20	35430	1	0	35430	Nilla Wafers	61	19
24	40078	1	0	40078	Strawberry Lemonad...	37	1
26	35951	1	0	35951	Organic Unsweetene...	91	16
28	35108	1	0	35108	Salted Butter	36	16
35	28413	1	0	28413	Bunny-Luv Organic C...	83	4

Figure 3: Order and Products dataset joined

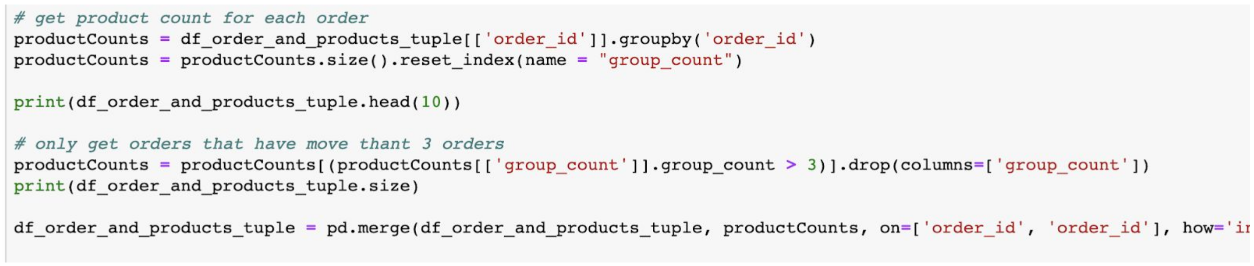
The next step is to start filtering some of the data that I don't consider meaningful because it will slow down the preprocessing. Note that I keep all our data inside pandas data frame. Figure 4 shows how the data is filtered to only allow products that appear in more than 500 orders out of the 3 million and this will allow the data to have products that are frequently purchased by users and to improve data processing.



```
# get all unique products
unique_products_df = df_order_products[['product_id']].groupby('product_id')

# group all products to get the counts
unique_products_with_counts_df = pd.DataFrame(unique_products_df.size().reset_index(name = "group_count"))

# filter out the the product if it does not appear in more than 1,000 orders out of the 3 million
unique_products_filtered = unique_products_with_counts_df[unique_products_with_counts_df.group_count > 1000]

unique_products_filtered = unique_products_filtered.set_index('product_id')
```

Figure 4: Filter out products that are not too frequent

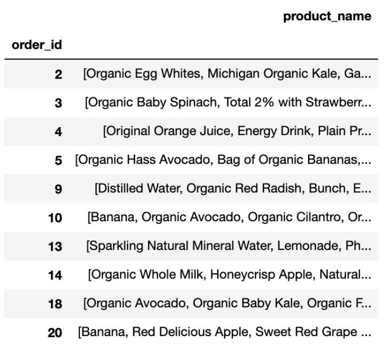
We continue by filtering our more data, in this case it makes sense to drop the orders which have less than 3 products. This is again, to improve our processing by having orders with more than 3 products it will be easy for the system to find better patterns. Figure 5 shows how this is accomplished. Then we need to do one more step which is join our most updated orders dataframe against the "products" dataframe in order to capture the actual names of the items.



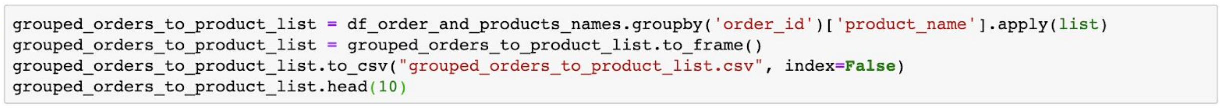
```
# get product count for each order
productCounts = df_order_and_products_tuple[['order_id']].groupby('order_id')
productCounts = productCounts.size().reset_index(name = "group_count")

print(df_order_and_products_tuple.head(10))

# only get orders that have move thant 3 orders
productCounts = productCounts[(productCounts[['group_count']].group_count > 3)].drop(columns=['group_count'])
print(df_order_and_products_tuple.size)

df_order_and_products_tuple = pd.merge(df_order_and_products_tuple, productCounts, on=['order_id', 'order_id'], how='ir
```

Figure 5: Remove orders with few products bought

The next step is one of the most important and challenging because we must calculate the one-hot-encoding for a very large set of data. In this case the encoding is not easy because there are still multiple rows for the same order. This means that we have to implement a grouping procedure to only allow one order per row. Figure 6 shows the implementation as how the data looks like now that the products are lists.



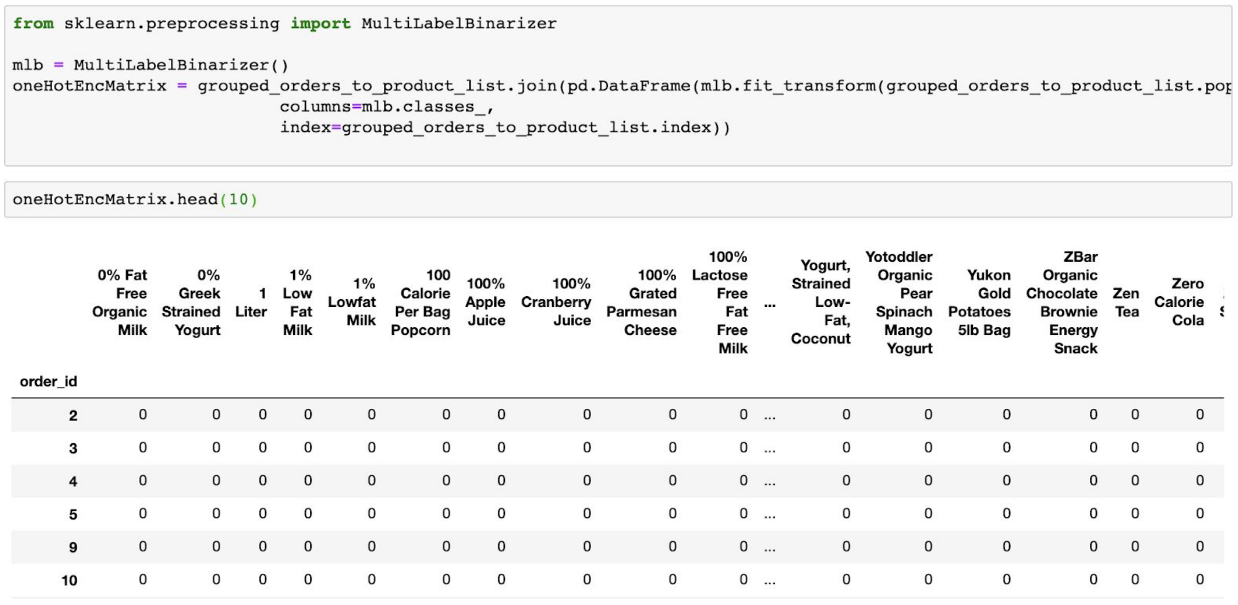
	product_name
order_id	
2	[Organic Egg Whites, Michigan Organic Kale, Ga...
3	[Organic Baby Spinach, Total 2% with Strawberr...
4	[Original Orange Juice, Energy Drink, Plain Pr...
5	[Organic Hass Avocado, Bag of Organic Bananas,...
9	[Distilled Water, Organic Red Radish, Bunch, E...
10	[Banana, Organic Avocado, Organic Cilantro, Or...
13	[Sparkling Natural Mineral Water, Lemonade, Ph...
14	[Organic Whole Milk, Honeycrisp Apple, Natural...
18	[Organic Avocado, Organic Baby Kale, Organic F...
20	[Banana, Red Delicious Apple, Sweet Red Grape ...



```
grouped_orders_to_product_list = df_order_and_products_names.groupby('order_id')['product_name'].apply(list)
grouped_orders_to_product_list = grouped_orders_to_product_list.to_frame()
grouped_orders_to_product_list.to_csv("grouped_orders_to_product_list.csv", index=False)
grouped_orders_to_product_list.head(10)
```

Figure 6: Group by orders to products in a list

The one-hot-encoding is now ready to be calculated. For this task we will use the MultiLabelBinarizer class from sklearn.preprocessing. It will go over each row in the dataframe and it will encode (0 or 1) the products as columns while the orders stay as rows. Figure 7 shows the implementation and well as how the data looks like.



```
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
oneHotEncMatrix = grouped_orders_to_product_list.join(pd.DataFrame(mlb.fit_transform(grouped_orders_to_product_list.pop
                          columns=mlb.classes_,
                          index=grouped_orders_to_product_list.index))
```

```
oneHotEncMatrix.head(10)
```

	0% Fat Free Organic Milk	0% Greek Strained Yogurt	1 Liter	1% Low Fat Milk	1% Lowfat Milk	100 Calorie Per Bag Popcorn	100% Apple Juice	100% Cranberry Juice	100% Grated Parmesan Cheese	100% Lactose Free Fat Free Milk	...	Yogurt, Strained Low-Fat, Coconut	Yotoddler Organic Pear Spinach Mango Yogurt	Yukon Gold Potatoes 5lb Bag	ZBar Organic Chocolate Brownie Energy Snack	Zen Tea	Zero Calorie Cola
order_id																	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

Figure 7: Full one-one-encoding implementation and data schema

Data Exploration and Interesting Findings:

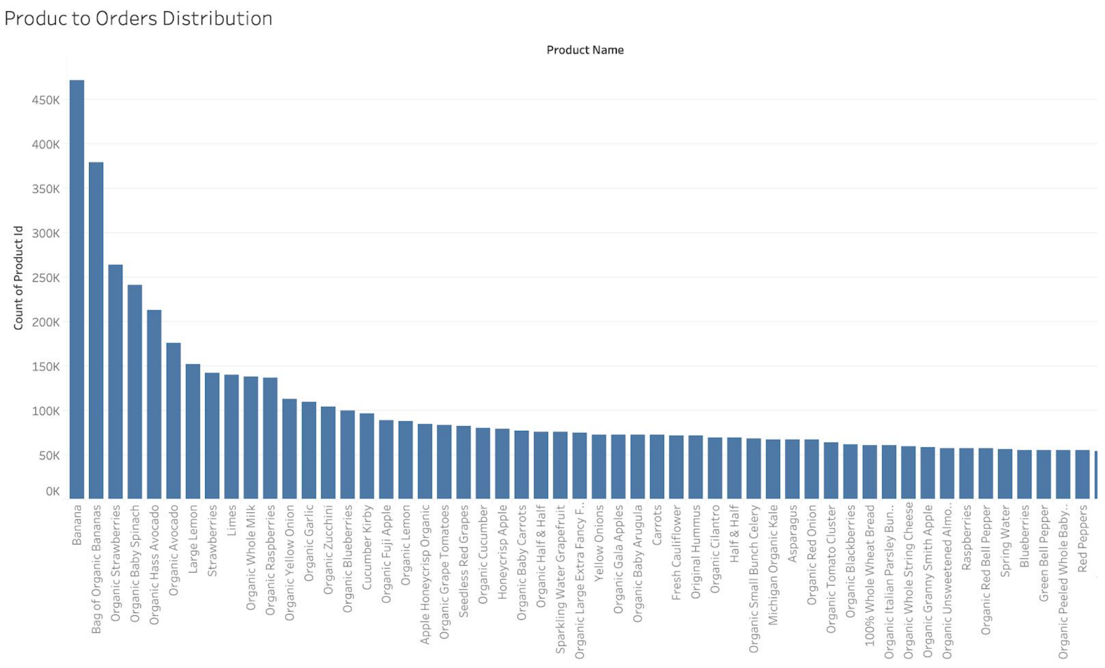


Figure 8: Distribution between products and orders

Model Implementation

Association Rule Mining - Apriori Algorithm

The Association Rule Mining is a very well known algorithm in machine learning which is used in different market basket analysis. I decided to use this algorithm after I learned about it during my Large Scale Analytics class. Also, this algorithm will be well suitable for this project. So, what is this algorithm?

The Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large datasets. It identifies frequent if-then associations called association rules which consists of an antecedent (if) and a consequent (then).

For example, *if a cereal box, then milk. This means that if a customer adds a cereal box to their shopping cart then it means that they are very likely to buy milk as well.* In this case the **Antecedent** is cereal box and the **Consequent** is the milk.

What are some common metrics to measure this the association using this algorithm?

- **Support** is the percentage of orders that contains the item (products) set. For example, how many times the set of *{cereal, milk}* appears in the different orders. This can be represented in a mathematical formula as follows:

$$Support = \frac{Number\ of\ transactions\ with\ both\ A\ and\ B}{Total\ number\ of\ transactions} = P(A \cap B)$$

- **Confidence** determines the number of times the if-then statements are found true. For example given two products, *cereal* and *milk*, confidence measures the percentage of times that product *milk* is purchased, given that product *cereal* was purchased. This can be represented in a mathematical formula as follows:

$$Confidence = \frac{Number\ of\ transactions\ with\ both\ A\ and\ B}{Total\ number\ of\ transactions\ with\ A} = \frac{P(A \cap B)}{P(A)}$$

- **Lift** can be used to compare confidence with expected confidence. For example, given two items, *cereal* and *milk*, lift indicates whether there is a relationship between *cereal* and *milk*, or whether the two items are occurring together in the same orders simply by chance. This can be represented in a mathematical formula as follows:

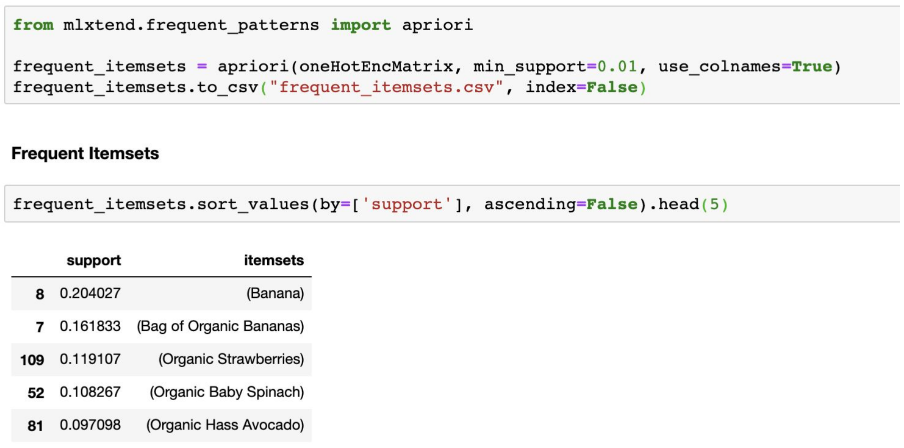
$$ExpectedConfidence = \frac{Number\ of\ transactions\ with\ B}{Total\ number\ of\ transactions} = P(B)$$

$$Lift = \frac{Confidence}{Expected\ Confidence} = \frac{P(A \cap B)}{P(A) \,.\, P(B)}$$

Association Rule Mining Implementation

In the data processing section we talked about how the data needed to be in the One-Hot-Encoder format because this is how the Association Rule algorithm expect it. There are very useful tools that I used from the MLxtend library (python). The main libraries I will be using are *"**apriori** and **association_rules** from mlxtend.frequent_patterns"* for this project. I initially started to implement my own version of this algorithm but I did not have enough time to make it efficient and since we are dealing with a very large dataset then it would not be feasible.

Let's run the model and answer the following questions:
What are frequent items in the dataset? Figure 9 shows the implementation details as well as how the data looks. It is interesting to note that the set of frequent items shown are in the set of popular items as seen in Figure 8. This can help up conclude that the model is selecting the correct product as one would expect. The min support is a very important parameter but it takes more time as the number decreases. The min support parameter is used to exclude rules in the result that have a support or confidence lower than the minimum support and minimum confidence respectively.Figure 10 shows the association between products in the dataset. This is where the recommendation model will be selection the recommendations given a user input. Finally, Figure 11 shows an example of simple recommendation. This recommendation function uses the association rules dataset to search for items given by the used based on the maximum support metric.



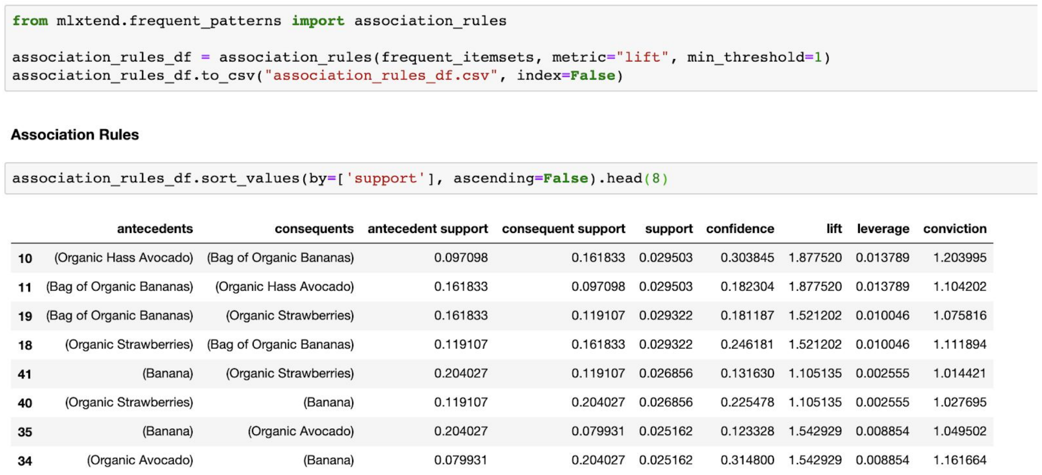
```
from mlxtend.frequent_patterns import apriori

frequent_itemsets = apriori(oneHotEncMatrix, min_support=0.01, use_colnames=True)
frequent_itemsets.to_csv("frequent_itemsets.csv", index=False)
```

Frequent Itemsets

```
frequent_itemsets.sort_values(by=['support'], ascending=False).head(5)
```

	support	itemsets
8	0.204027	(Banana)
7	0.161833	(Bag of Organic Bananas)
109	0.119107	(Organic Strawberries)
52	0.108267	(Organic Baby Spinach)
81	0.097098	(Organic Hass Avocado)

Figure 9: Frequent Patterns in the dataset



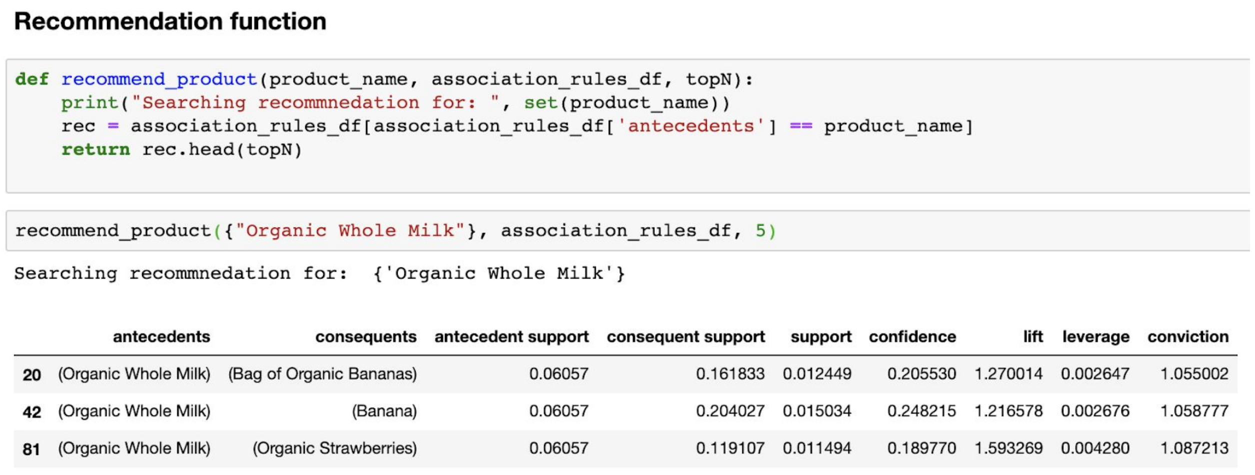
```
from mlxtend.frequent_patterns import association_rules

association_rules_df = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
association_rules_df.to_csv("association_rules_df.csv", index=False)
```

Association Rules

```
association_rules_df.sort_values(by=['support'], ascending=False).head(8)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
10	(Organic Hass Avocado)	(Bag of Organic Bananas)	0.097098	0.161833	0.029503	0.303845	1.877520	0.013789	1.203995
11	(Bag of Organic Bananas)	(Organic Hass Avocado)	0.161833	0.097098	0.029503	0.182304	1.877520	0.013789	1.104202
19	(Bag of Organic Bananas)	(Organic Strawberries)	0.161833	0.119107	0.029322	0.181187	1.521202	0.010046	1.075816
18	(Organic Strawberries)	(Bag of Organic Bananas)	0.119107	0.161833	0.029322	0.246181	1.521202	0.010046	1.111894
41	(Banana)	(Organic Strawberries)	0.204027	0.119107	0.026856	0.131630	1.105135	0.002555	1.014421
40	(Organic Strawberries)	(Banana)	0.119107	0.204027	0.026856	0.225478	1.105135	0.002555	1.027695
35	(Banana)	(Organic Avocado)	0.204027	0.079931	0.025162	0.123328	1.542929	0.008854	1.049502
34	(Organic Avocado)	(Banana)	0.079931	0.204027	0.025162	0.314800	1.542929	0.008854	1.161664

Figure 10: Product associations in the dataset



Recommendation function

```
def recommend_product(product_name, association_rules_df, topN):
    print("Searching recommnedation for: ", set(product_name))
    rec = association_rules_df[association_rules_df['antecedents'] == product_name]
    return rec.head(topN)
```

```
recommend_product({"Organic Whole Milk"}, association_rules_df, 5)
```

```
Searching recommnedation for:  {'Organic Whole Milk'}
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
20	(Organic Whole Milk)	(Bag of Organic Bananas)	0.06057	0.161833	0.012449	0.205530	1.270014	0.002647	1.055002
42	(Organic Whole Milk)	(Banana)	0.06057	0.204027	0.015034	0.248215	1.216578	0.002676	1.058777
81	(Organic Whole Milk)	(Organic Strawberries)	0.06057	0.119107	0.011494	0.189770	1.593269	0.004280	1.087213

Figure 11: Recommendation Function Example

Evaluation

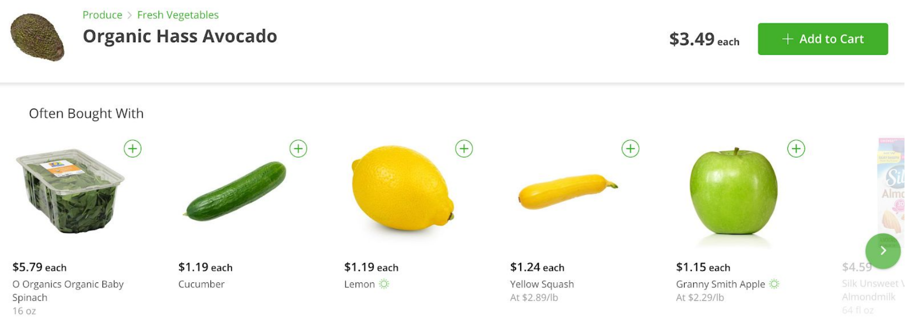
There are many ways how to evaluate our model, but what if I compare the recommendations engine with the actual recommendations from the Instacart website? I will access to the website and I will take a screenshot of a items that should recommend me similar products when I click on them.

Comparisons:

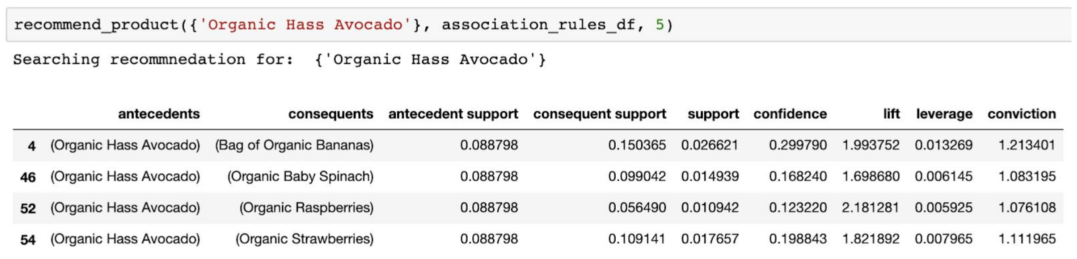
1. **Organic Hass Avocado**

   This was a successful comparison because Organic Baby Spinach is in both recommendations! I believe that if I decrease the minimum support in my function it should give me better results.

   a. **Instacart Recommendations**



   b. **Our Recommendation**



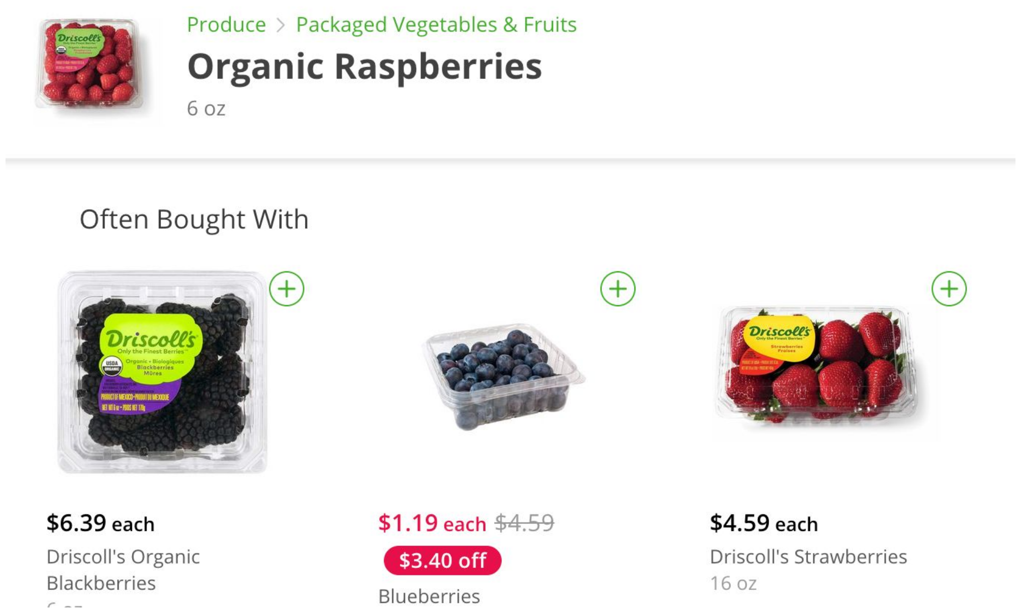
```
recommend_product({'Organic Hass Avocado'}, association_rules_df, 5)
Searching recommnedation for:  {'Organic Hass Avocado'}
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
4	(Organic Hass Avocado)	(Bag of Organic Bananas)	0.088798	0.150365	0.026621	0.299790	1.993752	0.013269	1.213401
46	(Organic Hass Avocado)	(Organic Baby Spinach)	0.088798	0.099042	0.014939	0.168240	1.698680	0.006145	1.083195
52	(Organic Hass Avocado)	(Organic Raspberries)	0.088798	0.056490	0.010942	0.123220	2.181281	0.005925	1.076108
54	(Organic Hass Avocado)	(Organic Strawberries)	0.088798	0.109141	0.017657	0.198843	1.821892	0.007965	1.111965

2. Organic Raspberries

This recommendation as still okay even though the recommended strawberries were not organic in the website. However, the good thing to note here is that our recommendations actually recommended all organic products so that is a good sign.

a. Instacart Recommendations



Produce > Packaged Vegetables & Fruits

Organic Raspberries

6 oz

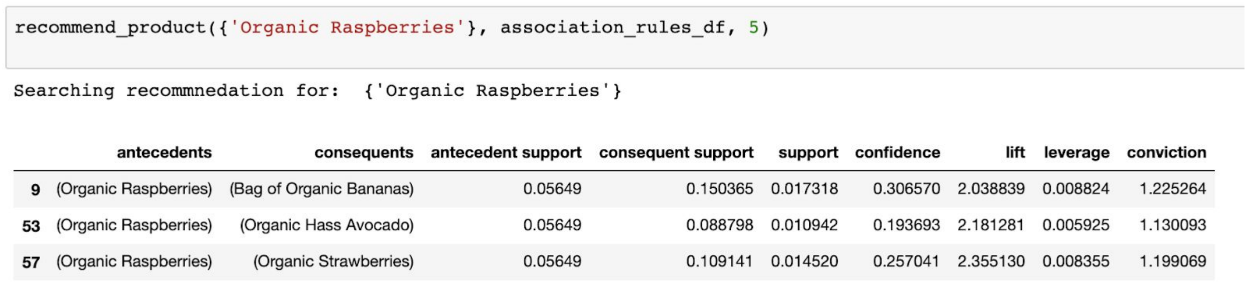
Often Bought With

$6.39 each
Driscoll's Organic Blackberries

$1.19 each ~~$4.59~~
$3.40 off
Blueberries

$4.59 each
Driscoll's Strawberries
16 oz

b. Our Recommendation



```
recommend_product({'Organic Raspberries'}, association_rules_df, 5)
```

```
Searching recommnedation for:  {'Organic Raspberries'}
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
9	(Organic Raspberries)	(Bag of Organic Bananas)	0.05649	0.150365	0.017318	0.306570	2.038839	0.008824	1.225264
53	(Organic Raspberries)	(Organic Hass Avocado)	0.05649	0.088798	0.010942	0.193693	2.181281	0.005925	1.130093
57	(Organic Raspberries)	(Organic Strawberries)	0.05649	0.109141	0.014520	0.257041	2.355130	0.008355	1.199069

Conclusion

Overall, I am very happy with this project and the results I was able to come up with. I was able to apply what I learned in the class and I can say that I have a good understanding of recommendation systems. Groceries recommendation is something I am passionate about and I had fun trying to find the best ways to recommend products to users. I had a few challenges dealing with such a large dataset but at the end I was able to figure out optimization techniques in python to work better with the data. There is definitely many other models and algorithms that can be used to accomplish the same task but I know the Association Rule Mining algorithm was a good one. I would investigate more algorithms if I had more time and perhaps I would create a website. Datasets can also be collected from other data sources not just Instacart. There are many items that are very similar such as a “bag bananas” or a “single banana” that maybe would make sense to put together in the same bin but it needs to be tested it of course. When one-hot-encoding is applied to the dataset it takes a good amount of time to complete. I would improve the way this is done, such as using spark in a cluster of machines. One thing to note here is that this is not a live recommendation model which means that we need to train the model offline when new data is available to consume.

References

- Data link from Instacart: https://www.instacart.com/datasets/grocery-shopping-2017
  - Data dictionary: https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b
- Project code link: https://github.com/miguelcovarrubias/cmpe-256-individual-project
- Algorithm Definitions: https://blog.usejournal.com/association-rule-mining-apriori-algorithm-c517f8d7c54c