Healthy Grocery Item Recommendation Engine

Business Case

Save consumers time at the grocery store

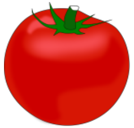
Americans are spending 1.48 billion minutes in the grocery store on an average day.

Our recommendation engine will allow shoppers to quickly research food items that meet their diet restrictions and will save shoppers valuable time.

Understand shopping behavior

Diet trends create huge opportunities for grocery stores to highlight food items that meet various restrictions.

Ex. 2016, gluten-free food sales were $1.66bn and are forecasted to reach to over $2.0bn by 2020.



Expose shoppers to new products

Data collected from our recommendation engine will quickly identify trends.

This data can then be utilized by manufactures to bring new products to market quicker and with less risk.

Allow consumers to monitor their food intake

Consistency (with variety) is key to maintaining any lifestyle change.

Healthy diet changes can significantly reduce a consumer's long-term healthcare costs.

Diets Analyzed

- Tagged items by ingredient content
- Cross referenced with allergens and trace ingredients

Vegan	• Excludes: All dairy, meat, eggs, honey, gelatin, and seafood products (any animal products) • 0.5% of Americans are vegan but more than 50% of shoppers have purchased vegan alternatives
Vegetarian	• Excludes all meat and seafood products but includes dairy and eggs • Estimated 7 million Americans are vegetarian
Pescatarian	• Excludes meat but includes seafood and egg products • Popular diet when transitioning to vegan or vegetarian
Gluten Free	• Excludes products containing wheat, oats or barley • Treatment for celiac disease but also popular for those without the disease
Soy Free	• Excludes any products containing soy • Very common ingredient in many processed foods
Dairy Free	• Excludes dairy products • Est. 30-50 million Americans are lactose intolerant

Process

Data Selection	Pre-Processing	Feature Engineering	Merging Data	Analysis
Open Food Facts (OFF) Instacart Order Data (ICD)	OFF required extensive cleaning, filtering Interpolating prior to analysis Data entry not consistent among countries	Created 6 new binary features for diets tagged according to ingredient content	Fuzzy Matching	Association Rule Mining (Apriori Algorithm) Hybrid Approach to recommender system: Collaborative Filtering and Content-based recommendations

Data

Open source food database in which users add nutritional for products around the world.

173 features that include complete nutritional information and ingredients for given products.

Semi-clean data.

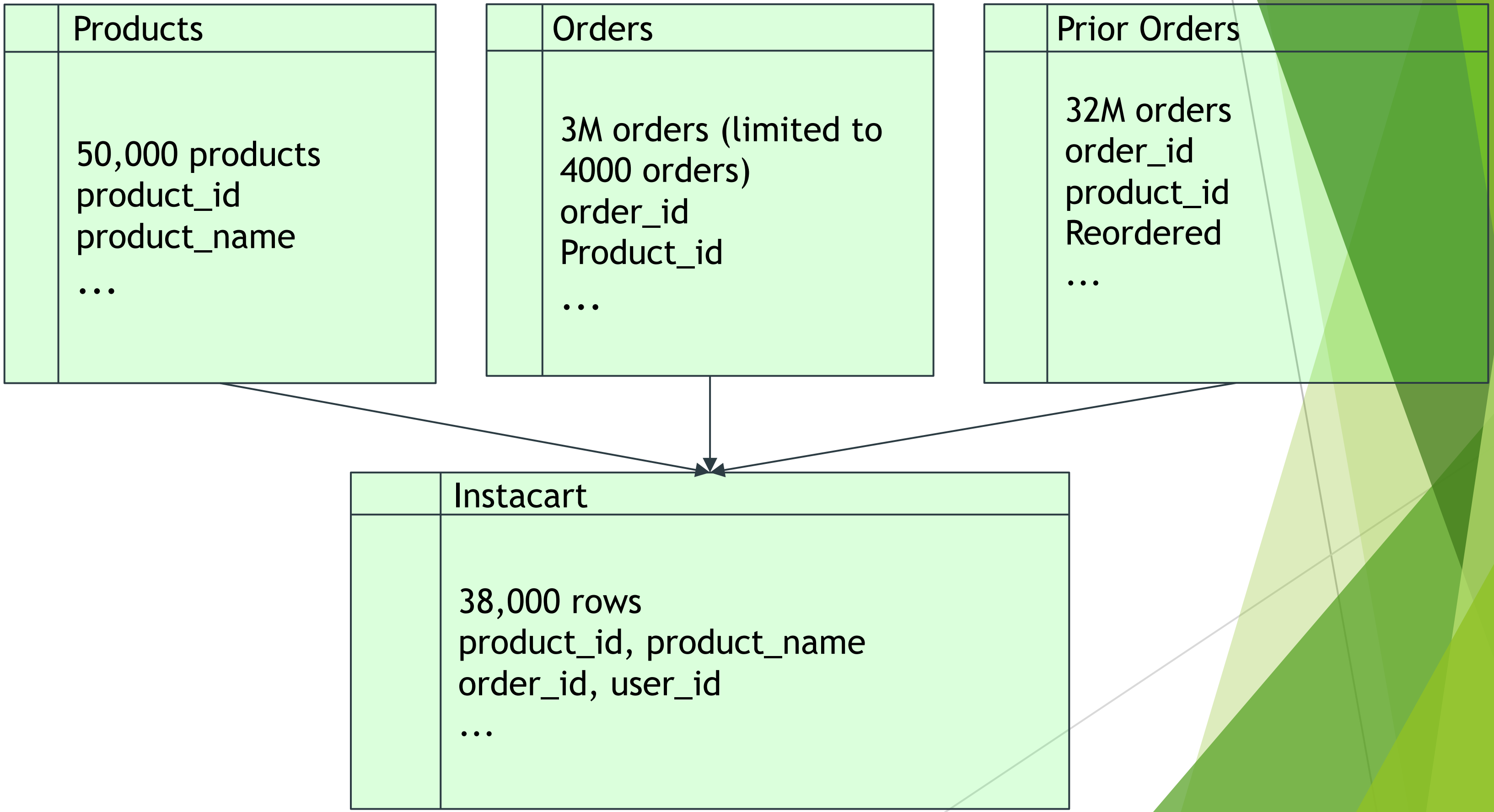
Over **350,000 products** for the US alone, including products across categories including meats, produces, brand names,store names and more

Anonymized and contains a sample of over **3** million grocery orders from more than **200,000** Instacart users.

Utilized 5000 orders of this data.

Tables used: ‘Order_Products’ , ‘Products’ tables and ‘Prior_Orders’ -- the products table was needed to match specific OFF data to ICD orders.

instacart


Products
50,000 products
product_id
product_name
...
Orders
3M orders (limited to 4000 orders)
order_id
Product_id
...
Prior Orders
32M orders
order_id
product_id
Reordered
...
Instacart
38,000 rows
product_id, product_name
order_id, user_id
...

Pre-Processing

- Filtered to U.S. only products
- Drop any observations without product names
- Limited to key features
  - Product name and ingredient content were the most important
  - Dropped unimportant nutrition variables
- Missing values
- Change data type
- Tagging food
  - Standardizing the format of ingredients (upper vs. lowercase, spelling, etc.)

```
# Interpolate null valuess
for x in features:
    reduced_data[x] = reduced_data[x].interpolate()
```

```
#Put string variables in lowercase
stringVarList = ['product_name', 'brands', 'countries_en', 'ingredients_text',
                 'serving_size', 'additives']

for var in stringVarList:
    reduced_data[var] = reduced_data[var].apply(lambda x: str(x).lower())
```

Tagging

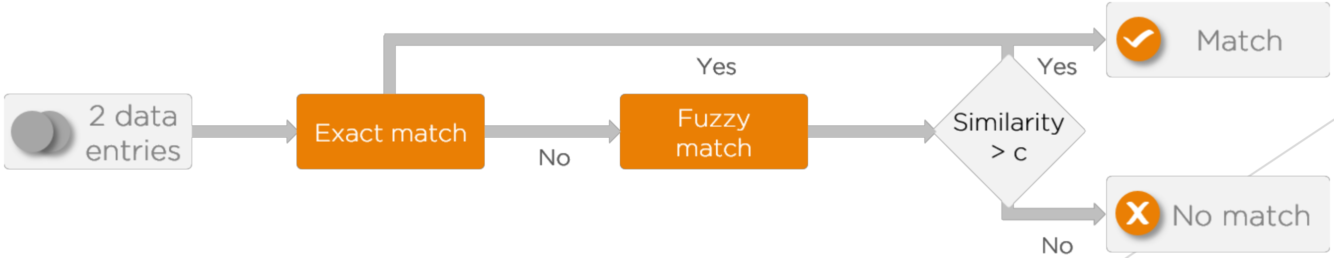
```
reduced_data['gluten_free'] = np.nan
reduced_data.loc[~reduced_data['ingredients_text'].str.contains(
    'wheat|barley|oats|gluten,|gluten]|flour',na=False),'gluten_free'] = 1
```

	product_name	ingredients_text	vegan	vegetarian	pescatarian	gluten_free	dairy_free	soy_free
0	banana chips sweetened (whole)	bananas, vegetable oil (coconut oil, corn oil ...	1.0	1.0	1.0	1.0	1.0	1.0
1	peanuts	peanuts, wheat flour, sugar, rice flour, tapio...	1.0	1.0	1.0	NaN	1.0	NaN
2	organic salted nut mix	organic hazelnuts, organic cashews, organic wa...	1.0	1.0	1.0	1.0	1.0	1.0
3	organic polenta	organic polenta	1.0	1.0	1.0	1.0	1.0	1.0
4	breadshop honey gone nuts granola	rolled oats, grape concentrate, expeller press...	1.0	1.0	1.0	NaN	1.0	1.0

Matching and Merging Data

- Matched products between datasets despite the lack of commonality
- Fuzzy Match(package: FuzzyWuzzy) was used as the Solution!
- Fuzzy String Matching, also called Approximate String Matching, is the process of finding strings that approximately match a given pattern. The closeness of a match is often measured in terms of edit distance, which is the number of primitive operations necessary to convert the string into an exact match.”- Marco Bonzanini
- highest = process.extractOne(unsweetened chocolate almond breeze almond milk,OpenFoodFacts_name)
- We then merge the two data sets with these newly matched items and now we have our dataset for modeling!

Instacart Name	Open Food Facts Name
all natural no stir creamy almond butter	all natural creamy almond butter
classic blend cole slaw	classic blend cole slaw
total 2% with strawberry lowfat greek strained yogurt	greek strained yogurt with strawberry, strawberry
unsweetened almondmilk	unsweetened almondmilk
lemons	lemons
baby spinach	baby spinach
unsweetened chocolate almond breeze almond milk	almond breeze, almond milk, chocolate
ginger root	ginger roots
air chilled boneless skinless chicken breasts	boneless & skinless chicken breasts
ezekiel 49 bread cinnamon raisin	bread, cinnamon raisin
plain pre-sliced bagels	plain presliced bagels



Source: Medium.com

Association Rule Mining

- We used the Apriori algorithm to determine which items might be purchased together
- Parameter values: 0.015 for minimum support, 0.55 for minimum confidence, and 3 for minimum lift
- A drawback of this approach is that the number of items in a database increases, the number of possible combinations of items to be purchased together also increases exponentially
- This leads to a very computationally expensive algorithm
- Because of this we decided to use PySpark to speed up computation of what was a fairly large dataset for basic python
- Using PySpark cut our process time significantly and led us to a few key insights:
  - Not surprisingly people who buy fruits and veggies usually pair them together
  - String cheese is often paired with strawberries
  - Buttermilk biscuits are often paired with baby spinach

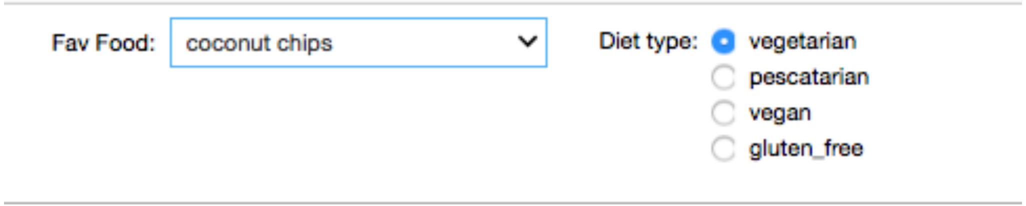
```
Rule: mango chunks -> avocado
Support: 0.021505376344086023
Confidence: 0.6666666666666667
Lift: 6.2
=========================================

Rule: buttermilk biscuits -> baby spinach
Support: 0.021505376344086023
Confidence: 1.0
Lift: 8.454545454545453
=========================================

Rule: unsweetened almondmilk -> baby spinach
Support: 0.021505376344086023
Confidence: 0.6666666666666667
Lift: 5.636363636363637
```

PySpark Output

antecedent	consequent	confidence	lift
[blackberries, raspberries]	[strawberries]	0.6451612903225806	4.747129579004921
[unsweetened almond-milk, original]	[banana]	0.5116279069767442	2.9964312325973568
[blackberries, strawberries]	[raspberries]	0.5	7.175619834710744
[raspberries, hass avocados]	[bananas]	0.47058823529411764	3.3218555714968914
[avocado, strawberries]	[banana]	0.4375	2.562289207419899
[lemon, hass avocados]	[bananas]	0.41379310344827586	2.92094196804037
[honeycrisp apples]	[banana]	0.4094488188976378	2.398002947776553
[nectar girl, cocktail mix, cucumber, lime mint & agave nectar]	[banana]	0.4	2.342664418212479
[hass avocados, banana]	[strawberries]	0.3888888888888889	2.861464218455744
[blackberries]	[strawberries]	0.3883495145631068	2.8574954747408263
[fuji apple]	[banana]	0.3805309734513274	2.2286409288304556
[avocado, baby spinach]	[banana]	0.375	2.196247892074199
[large brown grade a eggs]	[bananas]	0.375	2.6471036585365852
[raspberries, banana]	[strawberries]	0.36	2.648898305084746
[lemon, bananas]	[hass avocados]	0.35294117647058826	3.637283993716181
[original hummus]	[banana]	0.34782608695652173	2.0370994940978076
[string cheese]	[strawberries]	0.34782608695652173	2.5593220338983054
[roasted turkey breast]	[bananas]	0.33962264150943394	2.397376898297285
[red peppers]	[banana]	0.3333333333333333	1.952220348510399
[100% whole wheat bread]	[bananas]	0.3333333333333333	2.352981029810298

Recommendation Modeling

1. Raw input (diet type and fav food if no purchasing history)
2. SVD to figure out consumer preferences
3. KNN to group products based on nutrition and size
4. Select overlaps from step 2 and step 3
5. Optimal number of products being shown on the screen
6. Customize the recommendation based on other info



Load inputs: diet type, fav food (or potentially food you hate the most, or purchasing behaviours)

SVD Recommendation

```
# type in any kind of product that you like, and list of diets
recp = findSimilar(12036,'vegetarian')
recp
```

	product_id	pd_name
0	432	Vanilla Almond Breeze Almond Milk
1	1025	Organic Fresh Basil
2	1360	Crunchy Peanut Butter
3	2014	Low Fat Kefir Cultured Milk Smoothie Lowfat Pr...
4	3583	Unsweetened Coconut Milk Beverage
5	3957	100% Raw Coconut Water
6	4472	Toasted Coconut Almondmilk Blend
7	4658	Imported Mineral Water
8	8309	Nonfat Icelandic Style Strawberry Yogurt
9	9327	Garlic Powder

Matched consumers based on their purchasing behaviour using SVD

1. Score comes from how many food one person have purchased in the past
2. Sparse matrix problem;

   A.Dealing with product by department

   B.Only use top items

KNN Clustering

Group products together based on nutrition amount, and volume (milk and cheese are not substitutable in this case, some food perishable)

How many K to choose?

1. Elbow method

2. How many products we need

3. How to maintain the dictionary

Recommendation:

Once we have all the food info, we can store all type of food clustering in dictionaries and call that every time to be efficient.

```
recn = findNut('12036','vegetarian')
recn
```

0	33506
1	12036
6	14168
15	46077
16	39636
18	38782
20	35134
116	45253
183	20142
185	43511
186	32734
203	14477
206	45312
208	17889
210	37710
334	15902
364	17568
371	42579
378	7010
389	48002

Find overlaps

1. Select the overlaps between two lists
2. If no overlaps then decide between preference/ similarity tradeoff / AR
3. If too many then top 3 - 5 (not overwhelming)

	product_id	pd_name
0	1360	Crunchy Peanut Butter
1	12036	Gluten-Free Supergrain Pasta Organic Corn & Qu...
2	14477	Organic Spaghetti

Optimizing based on additional info:

A. Retail side: min cost, adjustments on inventory

B. Consumer side: price sensitive, prefer smaller size because always fresh

C. Personal info: allergies, type of food : liquid, chewy

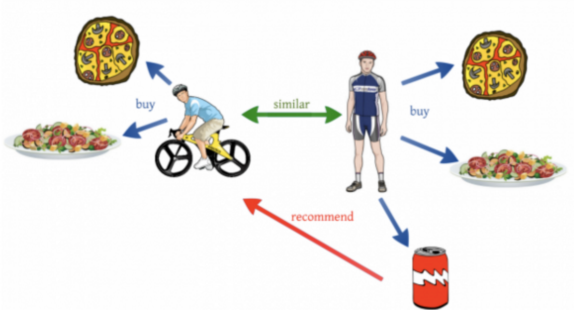
D. Other info: demographic, age, gender

Model Evaluation and Implementation

- Two step approach
  - Hybrid recommendation system
    - Customizable
    - Takes into account preferences for features such as size of product, type of product (beverage vs. food)
    - User similarity is dynamic over time
      - Two similar customers' purchasing patterns can change, which impacts the similarity recommendation
  - Association rule mining
    - Generalizable
    - Useful for "Cold Start" problem
    - Can be used as a "sanity check" for our previously generated suggested products
    - Computation grows exponentially as items are added to database
- Demo!

Results & Potential Extensions

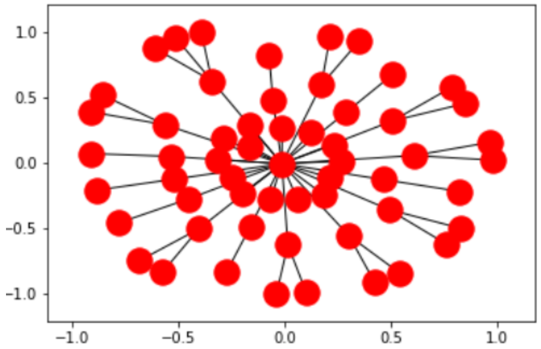
- Expand recommendation system to be applicable to additional diet types
  - For example, diets based on nutrition measured to the gram (individuals who eat Keto, eat a particular ratio of fat, carbs, and protein)
- Use algorithms to determine diet type groupings on their own - instead of manually assigning/tagging
- Model could take into account brand loyalty. For the most part model looks at generalized products but opening up to specific brands could present a more detailed if not much more challenging model
- Vary recommendations by time of day
  - Enter hour of order as a key input to the recommendation system

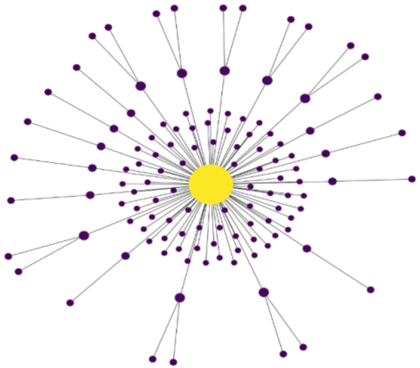


Lessons Learned

- Ideally we could have used all the data in the dataset but this quickly turned into a big data problem and something that was beyond our scope. As we added products our rows increased to millions and it was too computationally expensive to maintain the dataset
- Matching datasets with no column the exact same is difficult and an NLP problem.
  - The project would have been streamlined by having unique identifier such as a UPC to match on
  - Additionally, some matches were off. Could create a rule for exact matches but number of products small enough we could manually check.
- Association rule mining becomes increasingly time-consuming as the size of the data increases.
  - This is due to the fact that as the number of items increases, the number of possible combinations increases exponentially

Appendix(Graph-Based Modeling)





Shows how orders and users are connected to each other based on diet dummies and purchase content

Interpretation is less intuitive than in association rules and collaborative filtering