Instacart product recommendation

VALENTIN FEHR

Introduction

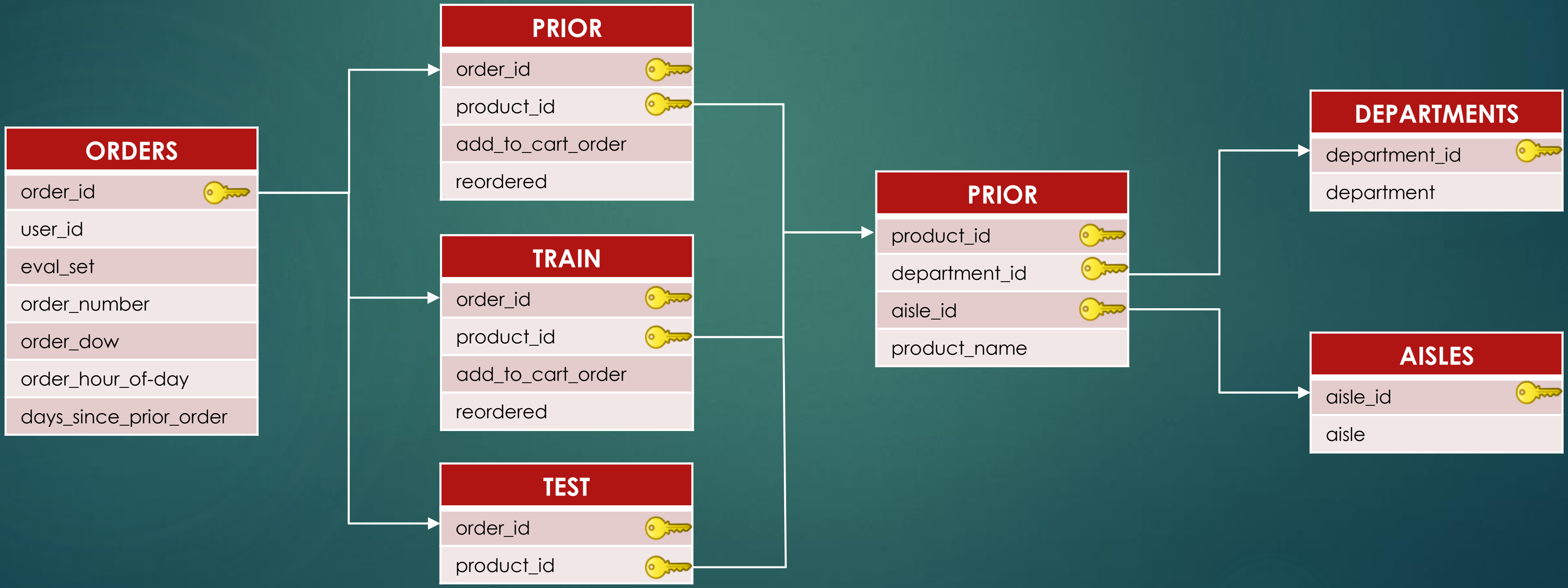
- Recommendations systems have several benefits, can save time for the user but can also make him consider buying something he might not have.

- Today we are looking at Instacart, an app that aims to make it easier for its users to shop for what they want efficiently.

- The goal is to build a recommendation engine based on data gathered from previous orders.

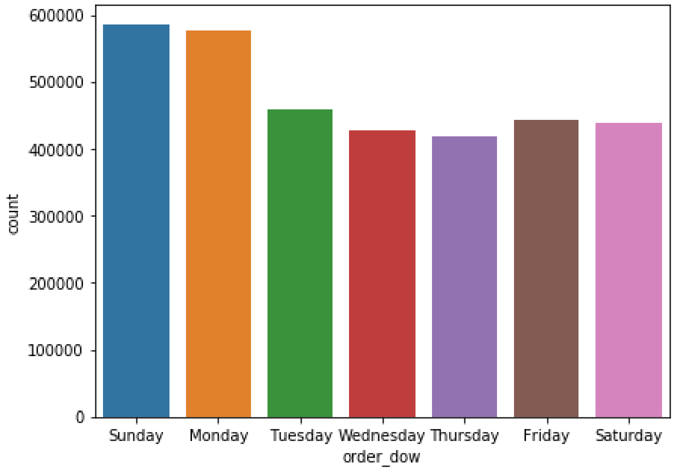
The Data

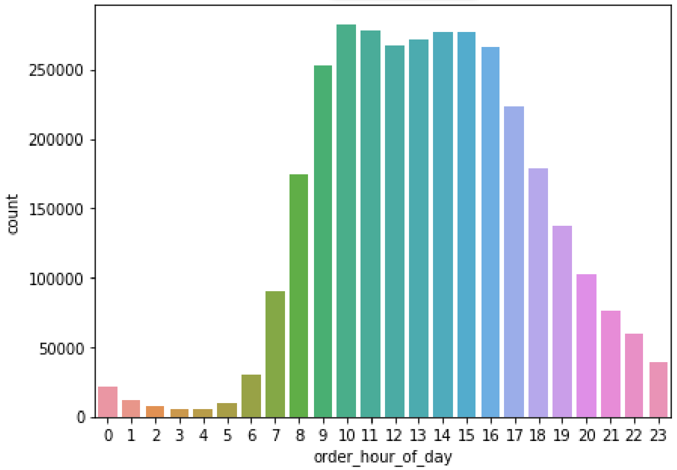
- The data is divided into 7 .csv files with the following schema :

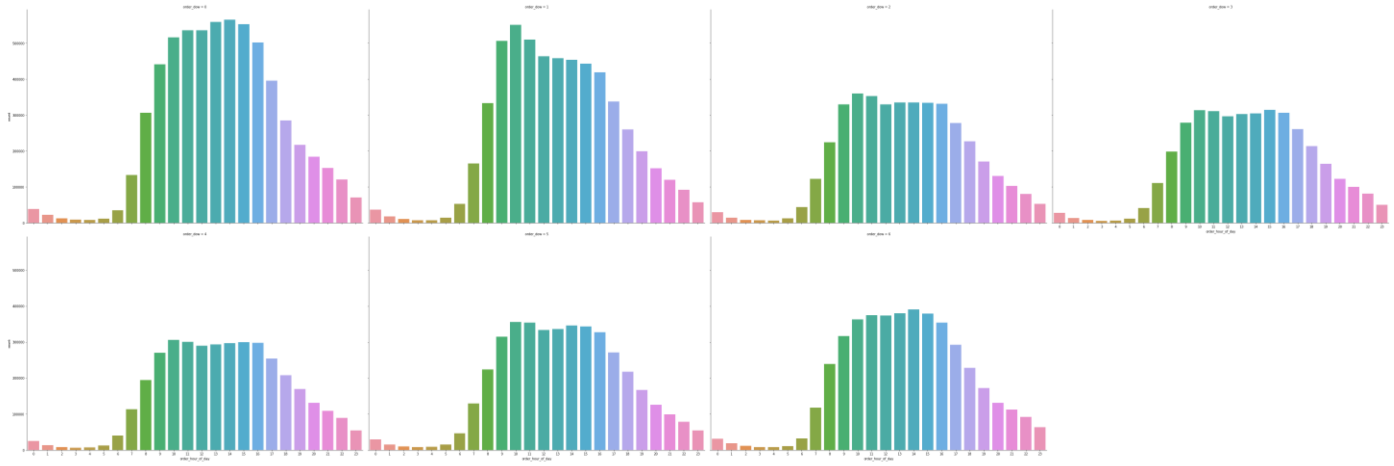


Data Exploration

- Sunday and Monday seem to be more common.
- 10 and 11am seem to be the most popular times to buy.
- On the weekends people tend to make orders later in the day.

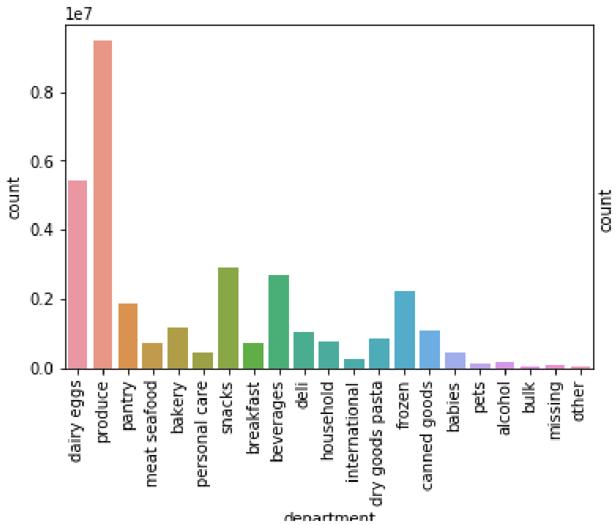


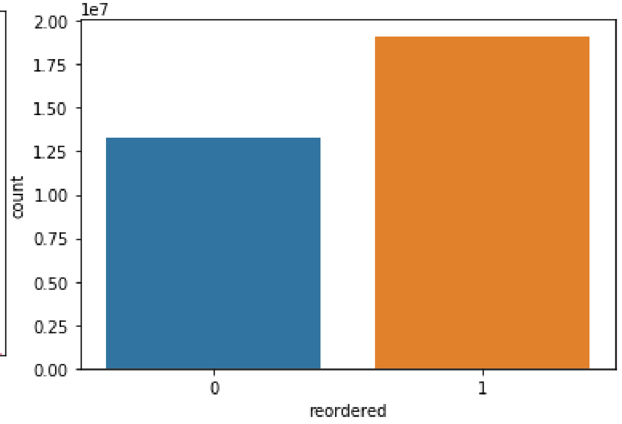


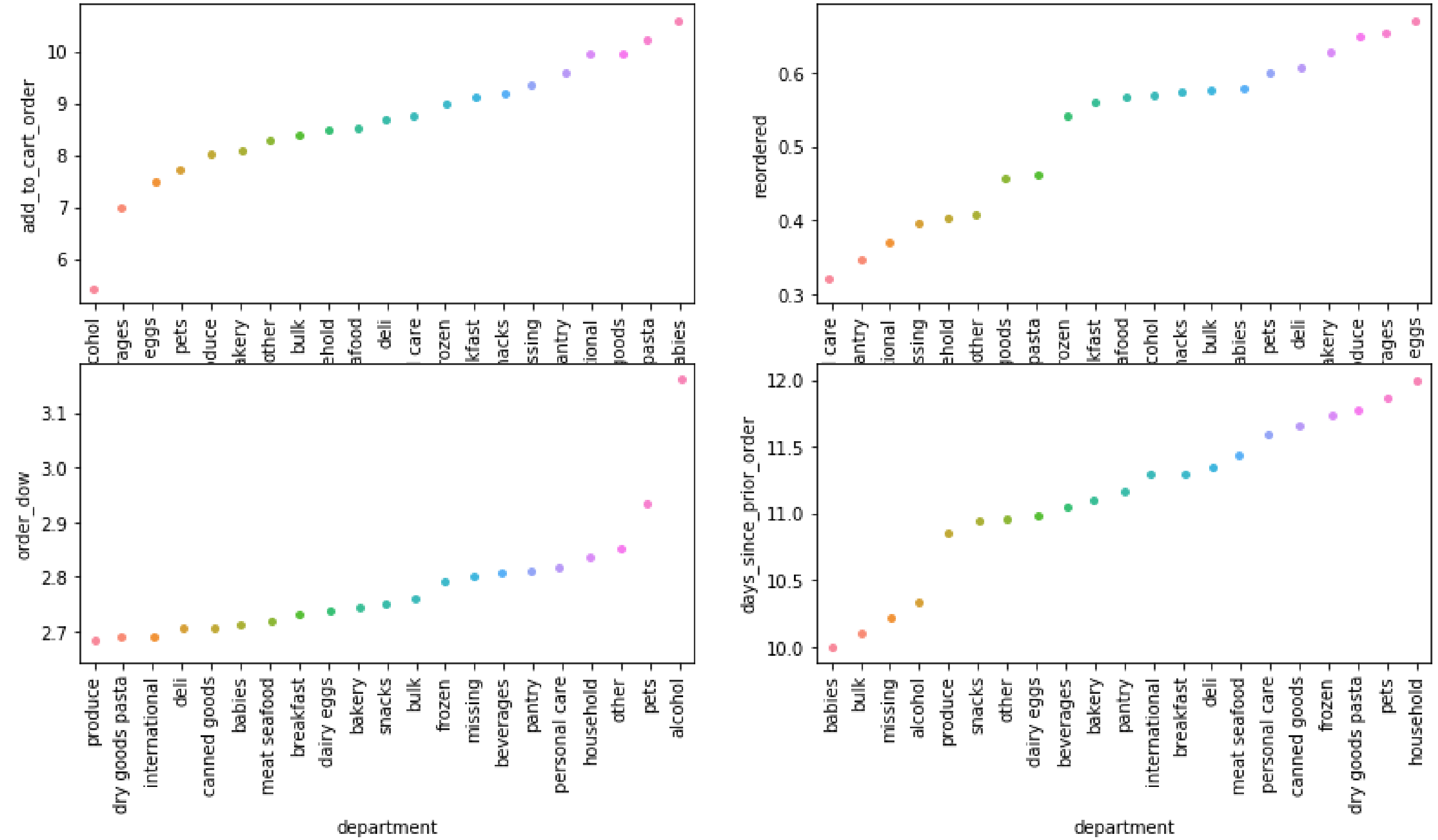


Data Exploration

- Most commonly bought products are produce and dairy/eggs.
- A good proportion of the items bought are not reorders.
- Some products show different kind of behaviors from buyers.







Feature Engineering

- Three types of features:

  User related / Product Related / Order Related

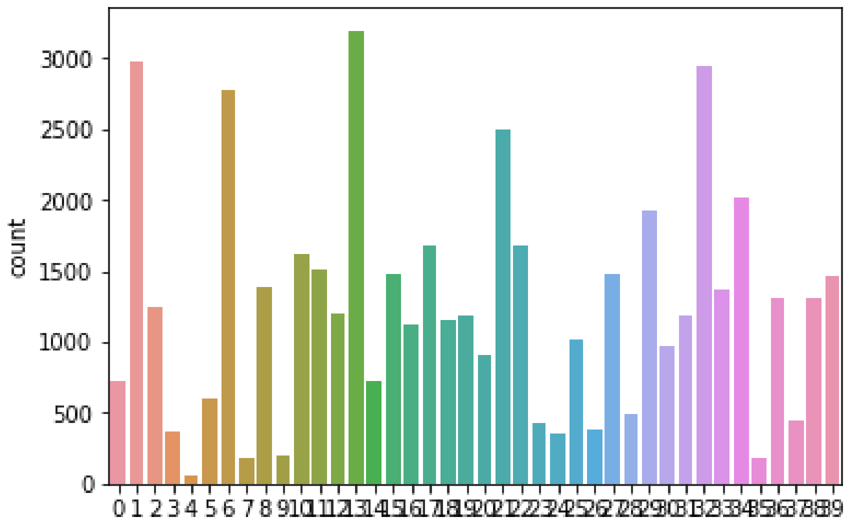
- User features : - Related to frequency : number of orders, average days between orders, …
  - Related to timing : average day of week, average hour of day, …
  - Related to orders : average order size, number of distinct products bought, ...

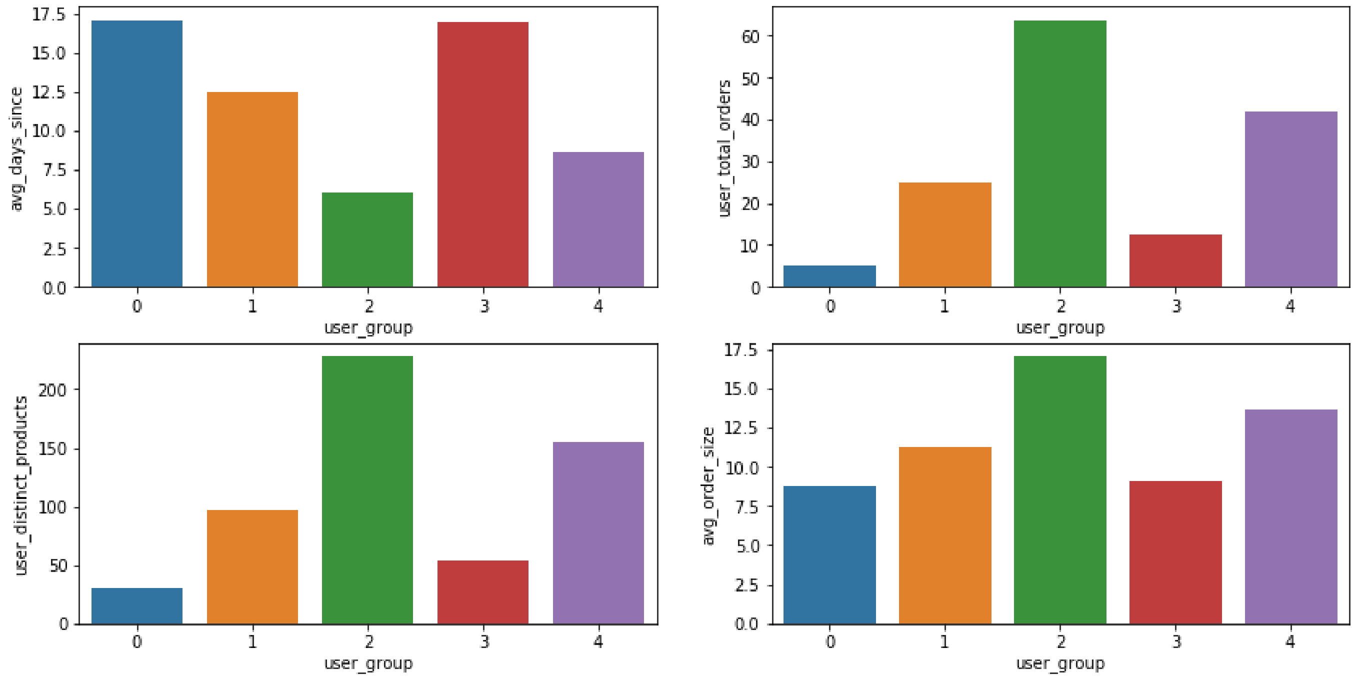
- Product features : - Related to frequency : number of times bought, number of times reordered, …
  - Related to timing : average day of week, average hour of day, …
  - Related to its nature : department id, aisle id,

- Order features : - Provided in the data : order number, day of the week, hour of the day, days since last order.

Clustering

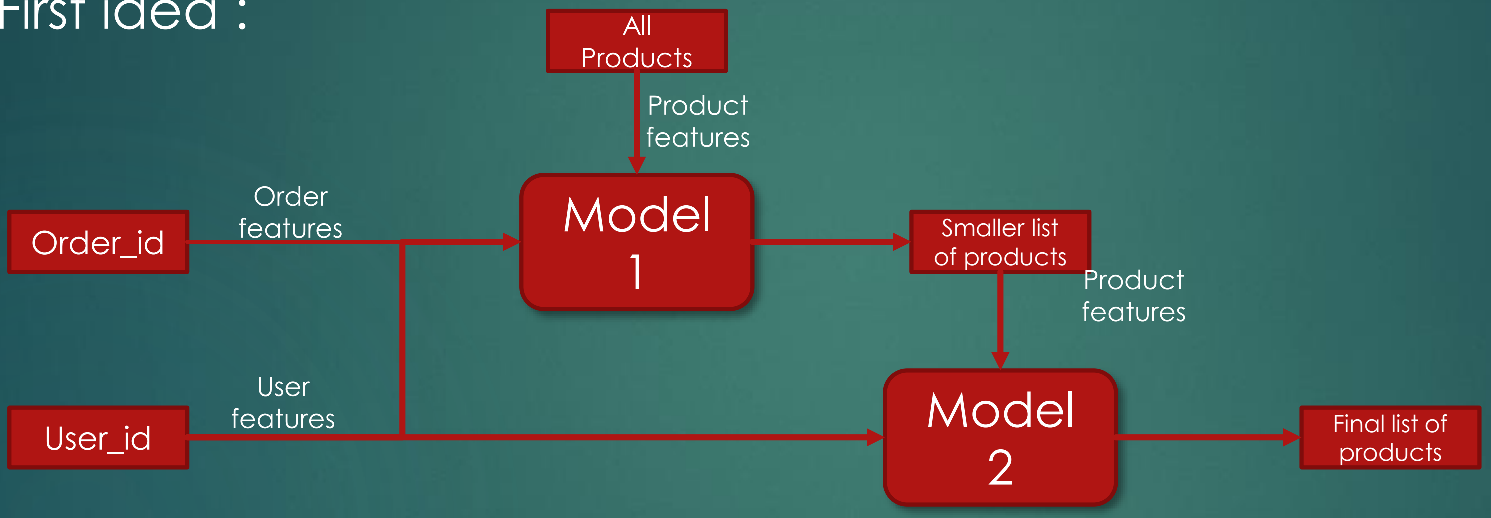
- First approach : user and product segmentation
  - Try to reduce size of data to work with and make associations.
- Using KMeans algorithm with silhouette score metric
- 5 clusters of users
- 40 clusters of products





Models – Introduction

- First idea :



- Tried with simple Model 1 : outputting only items previously bought.
- Finding what to train Model 2 on and the size of data were an issue.

Models – Size Prediction

- Second idea :
  - First model predicting size
  - Second model predicting probabilities of being in order

- Tried RandomForestClassifier to predict size of the order :
  - Awful accuracy (~11%)
  - But the guesses are not too far off of reality. The model predicts a size 15% bigger than the actual one on average.

Models - Recommendation

- Finally better idea for training
- Out of the 1million train orders, change half of them to have the wrong product (at random)
- Create new feature 'in_order', 1 for good product, 0 for fake
- Use user, product (real or fake) and order features to train gradient boosted tree (XGBoost) model predicting if product in order or not.
  - Good accuracy and f1 score (~85%) on validation set

	product_id	Product features	User features	Order features	in_order
Wrong product	Random product_id	Features of wrong product	Correct user features	Correct user features	0
Right product	Original product_id	Features of correct product	"	"	1

Models - Recommendation

- Different approach for test set
- Use order and user data from test set
- Join this data to all 50k product features
- Run our trained model to output probability that product is not fake
- Select a high threshold to only keep a handful of predicted products
- Repeat on all orders

product_id	Product features	User features	Order features	in_order (probability)
1	Product features (1)	User features from test order	Order features from test order	$0 < p(1) < 1$
2	Product features (2)	Same user features in all rows	Same order features in all rows	$0 < p(2) < 1$
etc. (50k)				

Models - Evaluation

- Bad accuracy but for a recommendation engine we care more about similarity of products : can manually check for similarity.
- Example result :

	product_id	product_name	aisle_id	department_id	selected
13172	13173	Beef Ramen Noodle Soup	69	15	1
16793	16794	Compleats Microwavable Meal Chicken & Rice	4	9	1
21133	21134	Cream Top Apricot Mango Fruit on the Bottom Yo...	120	16	1
21899	21900	Alive! Children's Chewable Multivitamin Natura...	47	11	1
24848	24849	Cheese & Fresh Herb Flatbread Pizza	79	1	1
26204	26205	California Pinot Grigio	62	5	1
27839	27840	Icelandic White Ale 6 Pack	27	5	1
27960	27961	Mentho Lyptus Cough Suppressant Oral Anestheti...	11	11	1
47198	47199	Rotelle	131	9	1
47615	47616	Sesame Street Organic Mini Blueberry Pancakes	52	1	1
47755	47756	RevitaLens Ocutec Multi-Purpose Disinfecting S...	44	11	1
49671	49672	Cafe Mocha K-Cup Packs	26	7	1

	product_name	aisle_id	department_id
10245	Organic Celery Hearts	83	4
11108	Organic 4% Milk Fat Whole Milk Cottage Cheese	108	16
13175	Bag of Organic Bananas	24	4
22034	Organic Whole String Cheese	21	16
43632	Lightly Smoked Sardines in Olive Oil	95	15
47208	Organic Hass Avocado	24	4
49301	Bulgarian Yogurt	120	16
49682	Cucumber Kirby	83	4

Next Steps

- Deploy model and conduct A/B testing tracking conversion rate
  - If not satisfactory refine model
- Refining :
  - Products association by name :
    - word2vec using product names from same order as sentences
    - Unsupervised learning to detect similar products (and maybe use as feature or preprocessing)
  - Better features (relational) :
    - Once an item has been picked change prediction for other items
  - Better models:
    - Probably need deep learning (NN) and Big Data methods to handle amount and complexity of data.

Thank you !

- Any questions ?