# Predicting Consumer Appetites: What Food Will Be In Your Next Grocery Shopping Cart?

Ben Seifert
General Assembly
Ben.seifert14@gmail.com

## Abstract

Machine learning algorithms have become an essential tool for predictive analyses. I have applied machine learning techniques to assess the accuracy of predicting consumer behavior. More specifically, I intended to predict what would be in a future order of a grocery shopper's shopping cart. My approach was to use a K-fold Cross-Validation algorithm. My results led to indicate that K-fold is a great start to approaching this problem.

## 1. Introduction

As the world has become more tuned into their mobile devices than ever, there seems to be multitudes of apps that address both simple and complex problems. A company named, Instacart, sought after to be the world's leading provider in grocery shopping. They created an app that allowed a user to select from geo-relative grocery stores, their desired items. The user can then purchase those items and have them delivered to their front door. Instacart has collected data on each one of their user's orders since 2012 (leaving out the identity of each user).

I aimed to predict what the next cart will look like for a reoccurring Instacart user. I used many machine learning algorithms but when using a K-Fold Cross-Validation method, it generated the best accuracy. I received a score that deviated only 8.35% from the Kaggle Leader Board where the original competition was posted. In the next section I will define my approach to the problem. More specifically, I will describe in depth how the data was obtained, how I analyzed the data, and what methods I used to create a model. I will conclude with a summary of my obtained results.

## 2. Method

This project aimed to predict whether a user would have a particular item in their shopping cart the next time that they shopped. I will first describe how I obtained my data

### 2.1 Data Acquisition

The data acquisition process was simple. I collected it on Kaggle.com under a Kaggle Competition labeled: Instacart Data Analysis. Kaggle.com is a platform for predictive modelling and analytics competitions. Instacart placed their competition on Kaggle along with the relevant data. Using my Kaggle account, BenS, I was able to sign up for this competition and obtain the data by downloading their csv files. The data came as seven separate csv's, one being a sample submission. The files were (as labeled): aisles, departments, order_products_prior, order_products_train, orders, products, and sample_submission.

### 2.2 Data Exploration

Much of my time was spent cleaning up and analyzing events that occurred in the data.

One that I would like to point out was through the use of a heatmap that defined the frequency of when orders were placed relative to hour of the day and day of the week. I discovered that most orders are placed between 9:00AM and 5:00PM Sunday and Monday.

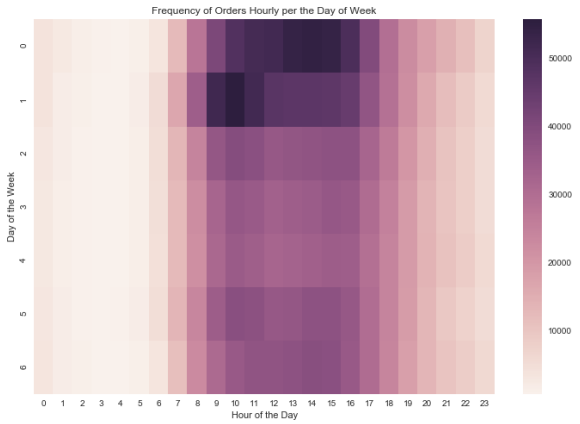


*Figure 1. Heatmap depicting frequency of orders given the day of the week and the hour of the day. 0=Sunday, 1=Monday.*

I then analyzed each department, the top three items ordered belonged to these categories (in order of frequency): Produce, Dairy/Eggs, and Beverages.

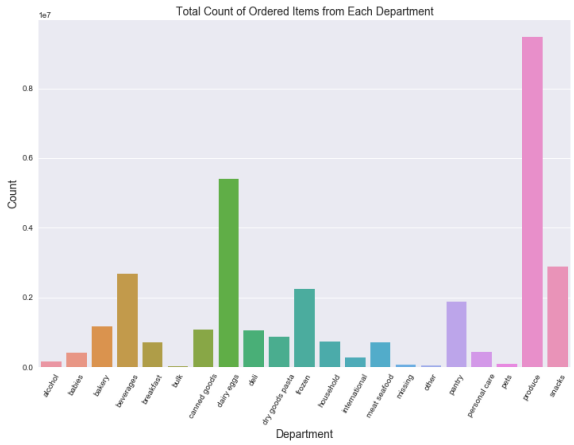


*Figure 2. Barchart depicting the total count of ordered items in each department.*

Looking further into the behavior of the users provided better insight on how to approach this problem. For instance, looking at the heatmap of what day and what time users ordered alcohol was interesting to me. It seems that the users of this app are preparing for weekend festivities as a majority of alcohol orders were placed on Thursday and Friday.

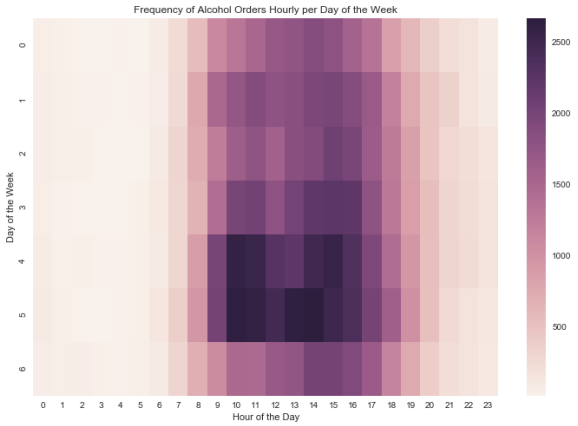


*Figure 3. Heatmap depicting frequency of alcohol orders given the day of the week and the hour of the day. 4=Thursday, 5=Friday.*

When looking at the reoccurrence of users, there was a positive skew with a strong right tail. This indicates that most of Instacart's users do not prolong their use of the app for more than roughly 20 orders.

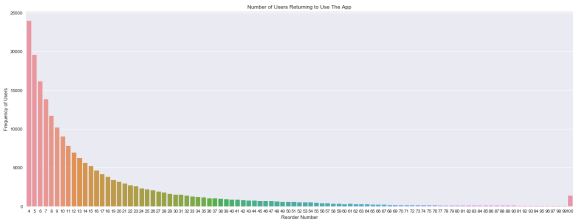

*Figure 4. This barchart depicts the frequency of users reordering. Any user that ordered over 100 times was placed in the 100 orders bin.*

## 2.3 Feature Selection

In choosing which features to use for my model, Instacart made it very easy. Using their csv's, I combined all the relevant datasets to each other using pandas' merge

method. I combined products, aisles, and departments to make a data frame that could be referenced for item information. I then combined the order_products_train data frame with the items data frame (only used 'product_id', 'aisle_id', and 'department_id' columns) so as to create a dataframe that would be useful to run a model on.

My plan was to use the features 'user_id', 'order_id', and 'product_id' to formulate a predictive model. What I intended was to group each product in a user's order. From there I would run a model on predicting what items (product_id) would appear most in every user's orders.

## 3. Analyses

My general framework for running a model to predict a user's next Instacart order began by first creating a function to extract only the user's order_id and the products in that order. The order_id is specific to each order placed by a user. For instance, if I placed an order this morning and then placed a separate order tonight, Instacart would assign separate order_id's to each of my orders (given that they were entirely separate orders and regardless of what I placed in each order).

I then ran a K-fold cross-validation function on my data to generate predictions as to what would be placed in each future order. Using these predictions, I then was able to obtain an F1 score.

### 3.1 Understanding K-fold Cross-Validation

Before I explain how I ran my model, I feel it is important to understand how a K-fold Cross-Validation works and what it's pros and cons are. A K-fold cross-validation (K-fold) randomly splits a training dataset into *k* folds without replacement. The algorithm then uses *k-1* folds as the training data and one fold as the test data. This procedure is repeated *k* times so that every fold has an opportunity to be the model's test-set. The picture may be able to provide some visual relief.

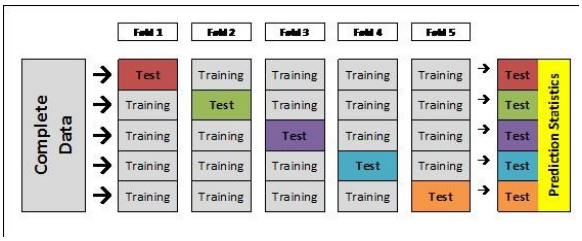


*Figure 5. You will see that each fold (5 folds) has a set amount of data as training and test. Since k=5, we divide the data into 5 portions, making one of those a test-set each time. In the end, they are combined to create average predictions about the data.*

The result of a K-fold is the estimate of each fold's performance. K-fold is a great algorithm for finding the optimal hyper-parameters that produce the best accuracy score. K-fold is also great because it allows each fold to be a portion of the test-set, ultimately reducing the variance estimate of the model. The unfortunate side of K-fold is the time it takes to run the model.

### 3.2 Steps to Obtaining an Accuracy Score

The first function accepts a data frame as an argument. It will then search through this data frame to see whether the person has reordered. For this prediction, we only want to view users that have reordered since the model will be able to learn from prior orders. The function will return a data frame of order_id's with a list of products for each order_id. Using KFold, I created a model that generated three folds and returned a predicted product list for each order_id.

| order_id | products |
|---|---|
| 1 | 43633 30881 5707 14947 |
| 36 | 35939 24964 26629 581 44359 47734 16759 |
| 38 | 33731 8012 |
| 96 | 24489 27966 |
| 98 | 4357 43654 34065 19731 1939 45204 33686 40986 ... |

*Figure 6. This screenshot portrays the dataframe that the first function returns. The left column is the order_id which can be linked to a user_id. The right column displays what product was selected in that order. The number of the product corresponds to a certain product in the main dataframe.*

To obtain an F1 score for each predicted order, I created a function that would analyze the precision and recall of each order.

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Figure 7. These formulas are the basis to obtaining a F1 score. I implemented the use of each of these in my second function.

If the order was True Positive, it would score it as a +1. If it was a False Positive or False Negative then the score would be a -1. I then had to create a third function that would search through my list of F1 scores and return a mean F1score.

## 4. Results

I demonstrated a successful application of machine learning to predict an Instacart user's next shopping cart order. My model generated a score of 32.56% accuracy. Though this may seem low, the Leader Board on Kaggle is 40.91%.

## 5. Moving Forward, Conclusions

Moving forward I believe it would be in the best interest to run some sort of boosting model. I have heard of a Python library called XGBoost in which it runs a form of adaBoosting in greater speed. Given the size of this dataset being 3million+ rows, a fast model would perform well. I also feel that the addition of features such as location or time of year would greatly increase a model's prediction.

**References**

"Instacart Market Basket Analysis." *Instacart Market Basket Analysis | Kaggle,* Kaggle, 16 May 2017, www.kaggle.com/c/instacart-market-basket-analysis.