EatsEncore!

A Food Recommendation Engine

Project Overview

We created a recommendation engine to increase sales. In short, our work applies to the following recommendation scenario:
"Customers who have purchased item X, have also purchased item Y."

- Analyzed food purchases at a real-world restaurant to identify items that are commonly bought together.
- Used this information to recommend additional items that customers may want to add to their order (i.e., upsell).
- Employed a constrained version of Market Basket Analysis (implemented with the efficient Apriori algorithm), which is an unsupervised machine learning method based on item associations.

Constrained Version of Market Basket Analysis

Model is constrained in two important ways:

- Age:
    - Kids = 0–14
    - Teens / Young Adults = 15–20
    - Adults = 21+
- Categorical rules wherein a user may only have 0 or 1 item per category in each category set:
    - Kids = {kids,side,beverage,dessert}
    - Teens / Young Adults = {entree,side,beverage,dessert}
    - Adults = {entree,side,beverage,dessert,alcoholic beverage}

Constraints result in a need for three separate models—one for each age group.

Data Engineering

Before feeding the data into our ML model or the SQLite3 product database used by Flask, extensive data engineering was necessary to modify the following.

- Multiple Item IDs
- Miscategorized Items
- Lengthy Item Descriptions
- Multiple IDs in an Order

Adjustments to Age Groups:

Kids

- Used orders with only kids items, no entrées
- Removed all Kids orders that included alcoholic beverages
- Reduced the number of Kids orders to work with to about 2,000

Teens / Young Adults

- Kept all orders, but removed alcoholic beverages and kids items.
- Gave us about 14,000 orders to work with for those aged 15–20.

Adults

- Removed all kids items from our orders.
- Worked with 17,500 orders.

Market Basket Analysis: Combinations

In market basket analysis, for a combination of items {X, Y, ...}, we are interested in its *support, confidence,* and *lift*.

Support is simply the fraction of the combination (for simplicity we consider just pairs here) {X,Y} of all N orders.

$$\text{Support} = \frac{\sum\{X,Y\}}{N}$$

Confidence is a measure of how likely Y is purchased if X is purchased.

$$\text{Confidence} = \frac{\sum\{X,Y\}}{\sum\{X\}}$$

Lift is how likely the pair {X,Y} is, controlling for the popularity of {Y}.

$$\text{Lift} = \frac{\text{Confidence}(\{X,Y\})}{\text{Support}(\{Y\})}$$

For the {X,Y} association to be significant, Lift should be > 1.

Market Basket Analysis: Our Dataset

After building a two-dimensional list of all possible combinations of categorized items for our order data set, we employ the Apriori algorithm as implemented in the apyori Python package.

- Approximately ~18,000 orders utilized for this project.
- Large product list allows for approximately 1,000 pairs. *We limited to item pairs—maximum item group length of two—since larger groups would not have statistically significant data.*
- Because of our lax limits on support (typically 0.001) and confidence (typically 0.01)—again, because many combinations are possible and our dataset is limited. Thus, we always require a Lift that is greater than 1.

Predictive Model

Iterated through the apyori output to build a dictionary of dictionaries *(the keys are hashed and access is fast)*. The primary keys are the item IDs, for which each has a dictionary attached that contains all categories as keys. The keys values are:

- Item's own category has a value of -1, since we never recommend items of the same category.
- If apyroi finds significant associations, the value assigned to the category is a list of associations—each being a dictionary containing the item id—plus significance, confidence, and lift for the association.
- If apyori did not find any significant association with an item of the category, the value is set to -2 for that category. If we need to propose an item of that category to accompany this item, we use the popularity model *(see next slide)*.

For proposed additions on an order, we loop over all open categories *(i.e., categories without item)* and generate a recommendation for each category.

Popularity Model

As a fallback recommendation model, we choose the simple *popularity model*—meaning if we don’t have market basket predictions for a given category, the recommendation is the most popular item *(i.e. the most ordered item)* in that category.

Limitations & Future Improvements

- Due to time constraints, our analysis has been very preliminary. Work on hyperparameter tuning and additional data engineering work could improve our results.
- Because of rules of combinatorics, the size of our dataset, and the size of our product list (menu), we realize that we stand little chance to produce highly significant associations.
- Did not create separate training and test sets—originally planned to do so, yet limited size of the available dataset made it imperative that we use all the available data to train our model.
- While we have taken great care and done in-situ testing of our code, there is, of course, always room for systematic error. This is something that should be checked in follow-up work.

Meet the Team

Christian Ott

A computational scientist with a PhD in Theoretical Astrophysics and 15+ years of experience in scientific computing, Christian has a strong understanding of mathematics, numerical methods, and statistics. Additionally he loves cats!

Charla Myers

A content-focused creative director, Charla sees data as a reflection of human behavior. Her goal is to combine her skills for creative development and user experience with analytics to become a data storyteller. Plus she's lived in 6 major American cities.

Eric Jung

A digital product manager who focuses on data analytics & strategy. Eric loves building products that enhance the magic of a customer's experience with data-driven insights. And he's a huge Disney fan.