Beer Maven

1. Abstract

Craft beer names have become increasingly cryptic as the industry continues to grow. While some beer names still include things like “IPA” in the title, others have cryptic names such as “Curly Wolf”, “Luponic Distortion”, and “Pliny the Elder”. Whether you’re at the grocery store, restaurant, or bar, this makes it hard to pick the right beer. Additionally, with so many different choices available for consumers (especially IPA lovers), it can be hard to tell which brew is best without trying them all. Beer Maven is a recommendation app that aims to solve these issues.

Beer Maven allows users to look up beers by beer, brewery, or both, and add beers to custom Beer List. The beer list is then sorted based on user preference. Other beer recommenders like Next Can already exist, but they do not limit the beers they recommend to what is actually available. Beer Maven is more useful at a bar, restaurant, or grocery store because it will only recommend beers that are actually available to you.

2. The Data

The data used to create Beer Maven is scrapped from beeradvocate.com. I scrapped over 600,000 beer ratings made by the top 250 user on beeradvocate.com. All ratings follow the scale outlined here by beer advocate’s founders:

4.50 - 5.00 = World-Class  
4.25 - 4.49 = Outstanding  
4.00 - 4.24 = Exceptional  
3.75 - 3.99 = Very Good  
3.50 - 3.74 = Good  
3.00 - 3.49 = Okay  
2.00 - 2.99 = Poor  
1.00 - 1.99 = Awful

In addition to ratings it also became important to know what type of beer each rating is for. To accomplish this, I created a separate dataset using beeradvocate.com’s Beer Types page and joined this dataset to the original one on beer name. In the resulting dataset, each row corresponds to a review and the columns are beer name, brewery name, reviewer username, rating, and beer type. I then assigned each distinct combination of brewery and beer name an id number to deal with beer of the same name. Finally, beers were assigned to 7 larger groups such as “IPA/PA, “Dark Ale”, and “Wheat Beer” based on their beer type.

3. Methods

Scraping is a bit dull so I’ll delay speaking about it here and include details later in an appendix A.

3A. Surprise

Beer Maven’s recommendation system is built using a python module called Surprise. Surprise support multiple type of recommendation algorithms, including NMF, SVD, and KNN. Surprise has a built-in performance evaluation method that performs 5-fold cross validation. Using this method, I found that the K-Nearest Neighbors algorithm with a baseline regularly achieved the best scores with a .3532 Root Mean Squared Error and .2476 Mean Absolute Error (look up a reason I chose these other than they are the defaults). Once I selected the KNNBaseline algorithm, I optimized parameters using GridSearch. I intended split the data into a training and testing set, but the Surprise methods required to do this is broken and crashes my computer. Since cross-validation was already performed during the model evaluation step I decided to train the algorithm on the entire training set. The system takes a modified dataset with user\_id, beer\_id, and rating as columns and returns top predicted ratings for every beer\_id a user has not rated.

3B. The Cold Start Problem

Like many recommendation systems, Beer Maven was ineffective because it knew nothing about new users and therefore could not sort beer based on their preferences. Essentially, any new user just got beers sorted by their average ratings on beer advocate, which they could have just looked up in the first place. I needed to design a way for users to rapidly add preferences to the recommender. I first tried having users rate 10 common beers that represented certain beer types: Lagunitas IPA, Fat Tire Amber, Heineken, Blue Moon, etc. These ratings were then added to the database before training and predicted on. However, the result was the same: results sorted by average rating.

Instead of having users rate representative beers, I decided to have them rate beer groups themselves. I went back to beeradvocate.com and scraped beer types so that I could add them as a feature in the dataset. There were 102 different beer types on the beer advocate types pages, so I consolidated these types in 9 major groups: IPA/PA, Dark Ale, Porter/Stout, Pale Lager, Dark Lager, Wheat Beer, High ABV, Fruity/Flavored, and Non-Alcoholic. Users are asked to rate these 9 beer groups on a 1-5 scales, and these scores are used to modify ratings in the dataset before the recommendation system predicts. For example, if a user is rating the IPA/PA group, all ratings for beer in that group receive +.5 rating for a 5, +.25 rating for a 4, nothing for a 3, -.25 rating for a 2, and -.5 rating for 1. I’ve played around with the plus and minus values for different ratings, and settled on .25 and 5 as increments, but further testing is needed to optimize this parameter.

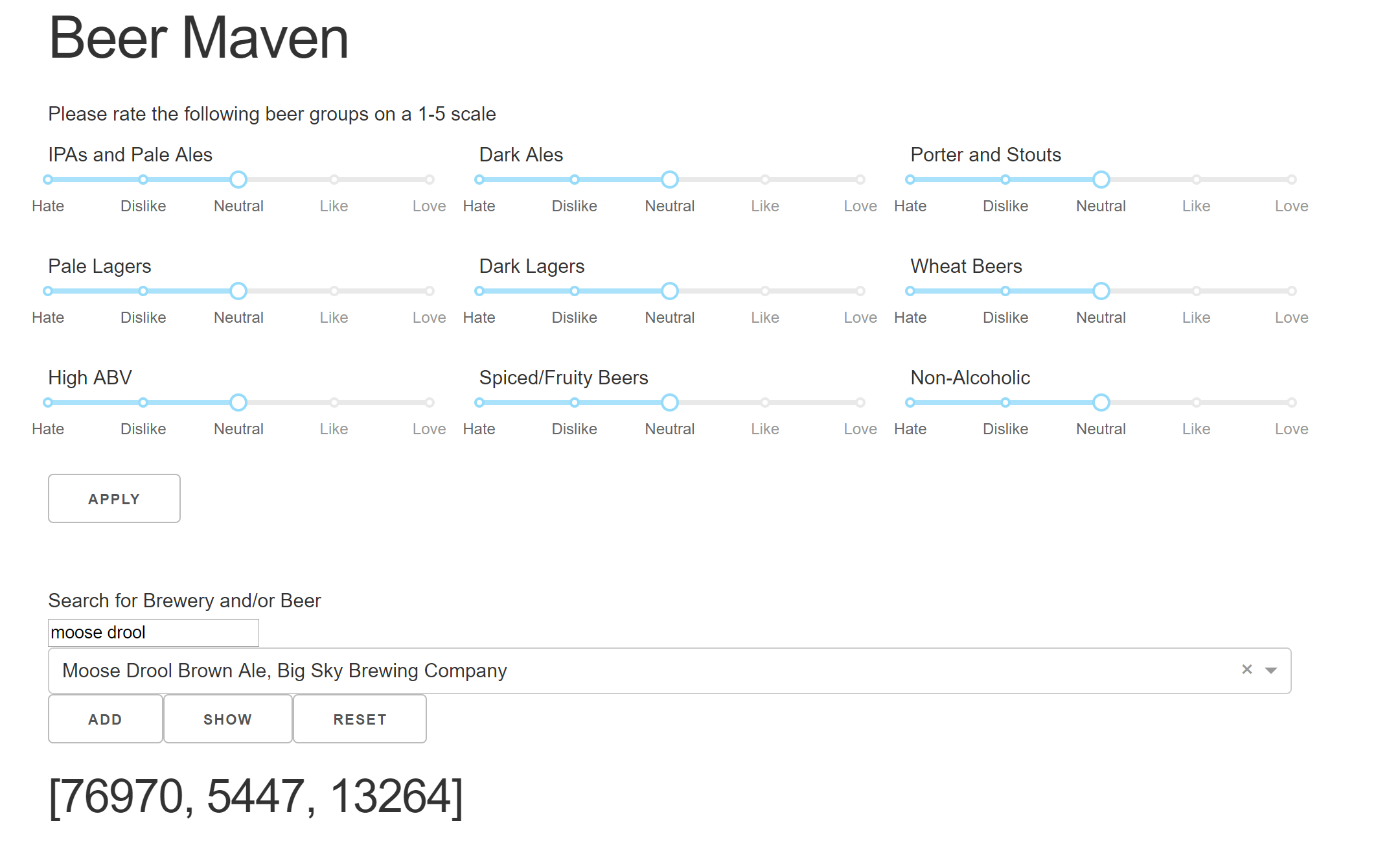
3C. Building the Beer List

Being able to quickly search for beers and add them to the beer list is crucial to Beer Maven’s function. My goal was to make using Beer Maven as convenient as possible else consumers wouldn’t want take the time to use the app. Originally I had hoped to use optical character recognition technologies to scan beer lists directly off of real life beer lists. I attempted to process beer lists using PyTesseract, but ran into many issues. Tesseract does poorly with the hand written chalk boards many beer lists are written on, and printed menus tend to have to many colors for Tesseract to be effective. Additional, the text bars and restaurants actually write on beer lists varies wildly. After some anguish I decided to table the OCR scanning aspect of Beer Maven in favor of manual addition of beers.

I used a python search engine module called Whoosh to allow users to quickly find beers. All beer names, breweries, and ids are input into the Whoosh engine. User inputs are submitted into a multi-field query that searches for beer name and brewery for the closest match, and then returns all information for hits. One the proper search hit is selected, the returned information is easily added to the beer list for sorting. The search work well and is relatively fast.

3D. The App

The Beer Maven application can be downloaded at <https://github.com/thewho14/Beer_Maven>. Follow the instructions in the readme to use it. The application is built using Plotly Dash. Dash is a module that converts python code into equivalent html, which can be used to build simple apps. Here’s a screenshot if you can’t access the app itself:



Please be patient as the app runs slowly. In the future I plan on taking steps to speed up the website.

4. Earlier Versions

Originally I tried to build a content based system instead of a collaborative one. To accomplish this, I scrapped reviews text from reviews for the top 250 most rated beers. I used spaCy and nltk to try and pull key phrases out of these reviews. The idea was to suggest beers based on the number of shared phrases. To do this I used spaCy’s build in part of speech tagger and looked for patterns in the text. For example