Beer Recommendation Engine Project Report

Justin Bates
March 1, 2021

Problem Identification Overview

Context

According to the US Alcohol and Tobacco Tax and Trade Bureau and US Commerce Department, in 2019 the US beer industry distributed 203.1 million barrels, or 6.296 billion gallons, of beer. At this time there were approximately 329.5 million people in the US. That's about 19.1 gallons of beer per person. The US Alcohol and Tobacco Tax and Trade Bureau also reported that there were about 6,400 reporting brewery facilities in the United States in 2019.

Source: U.S. TTB and U.S. Commerce Department, 2020

The table below shows the beer markets share data from Beer Marketer's Insights, for the years 2009 and 2019. The 10 year difference helps show the current direction of the market. We see obvious giants that hold more than 25% of the market fall between 8.9% and 6.9% in 10 years. We see that there is big growth and more growth potential for the smaller breweries with a 9.2% for the other category, containing more than 6,000 breweries.

Brewer/Importer	**2009 Market Share**	**2019 Market Share**
Anheuser-Busch Inbev	*48.8%*	*39.9%*
MillerCoors, LLC	*29.5%*	*22.6%*
Constellation	*5.0%*	*10.6%*
Heineken USA	*4.0%*	*3.3%*
Boston Beer	*0.9%*	*2.5%*
All Other Domestic and Imports	*11.8%*	*21.1%*
Total	*100.0%*	*100.0%*

Figure 1: 2009 and 2019 Brewer/Importer Market Share

Source: Beer Marketer's Insights, 2020

We bring up these two facts to help drive the future of finding new beers that you may never thought of trying.

Problem

How can we make it easier to try new and interesting beers you might never have thought to seek out?

We will build a beer recommendation engine that takes in beer titles and outputs 20 recommendations. This will use data collected from beer advocates. We will use beers that have 50 ratings from other users.

Problem Solution

Our beer recommendation engine uses one of the known recommender system techniques called collaborative filtering. Collaborative filtering can filter products or some element that users might want or like from the basis of actions or ratings similar users might have taken or liked. The main reason we chose collaborative filtering was because we wanted beers that other similar users might find good, not just similar beers. We want users to explore outside their normal taste to find something new that a similar user might have liked. Collaborative filterings most common form can be simplified to two steps. The first step being, find users with similar patterns to the user you are trying to find products or some element for. The second step is to make a prediction with the user data collected in step one.

We will talk about two different approaches we have taken to solve this recommendation system problem. The first is with k nearest neighbors, KNN, a neighborhood approach to finding like users. The other is singular value decomposition, SVD, a latent factor approach where variables aren't directly discovered but are the reason for other variables that are discovered.

Data

I started by web scraping beer advocates for potential useful beer data. The data I scraped included the beers' style, substyle, name, brewery name, beer advocate beer id, beer advocate brewery id, review count, rating count, rating, ba score from beer advocates, abv and few other things.

The data included:

35,552 Users
10,465 Beers (Reduced down to 8413 for quality over quantity)
4,026,233 Review Ratings

Deliverables

- Jupyter notebooks outlining the data science method used to produce models from dataset to solve the defined problem best
- A simple program that collects data builds the final recommendation engine models and generates weights with optimizer models if you don't want to run through the Jupyter notebooks
- Project Report
- High-Level Shareholder Summary Presentation

Data Preprocessing: Notable Steps

During my preprocessing, I pulled the scraped data in AWS S3 which I stored in different partitions depending on the beers categories and subcategories. I then used an AWS Glue crawler to crawl the stored CSV files. This generated two tables that I would go on to use, beer meta and beer rating. Beer meta was used to store category name, subcategory name, beer name, brewery name, review count, rating count, ba score. Beer review was used to store beer name, user name, and rating. With the help of AWS Athena I pulled the data into Quicksight and performed some EDA and really got to know the data. Below is an image of the final dashboard.

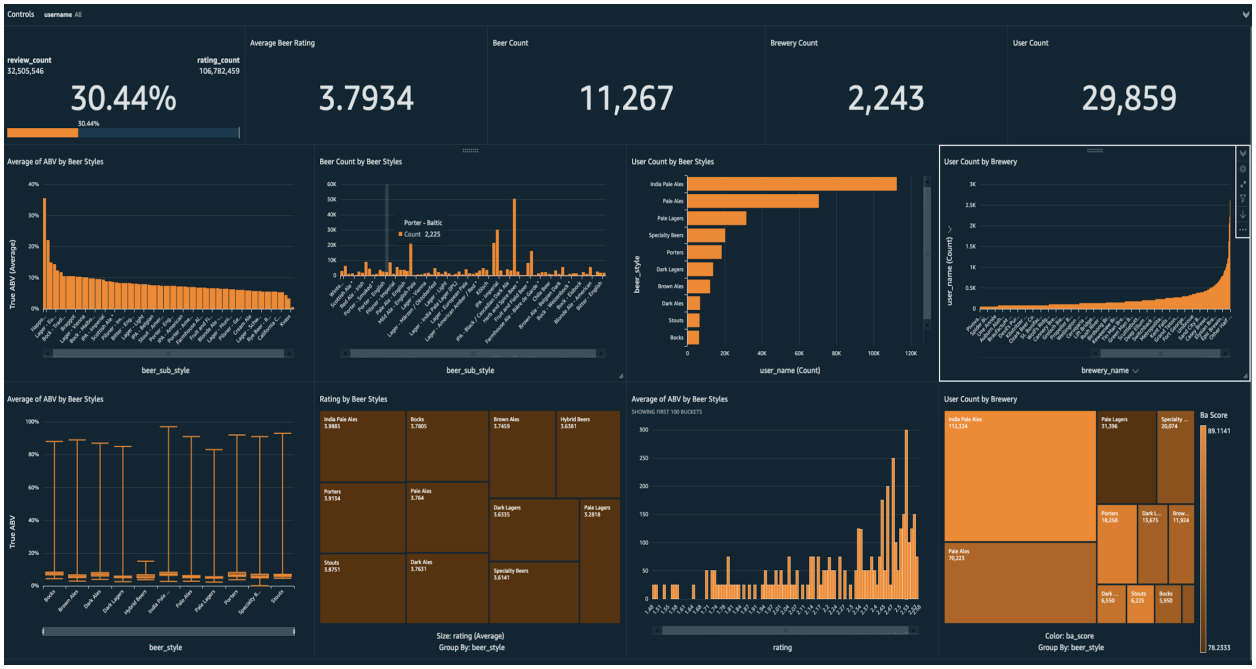

Figure 2: Quicksight Beer Dashboard

After building the dashboard, I used a jupyter notebook to look at the data a little differently. I started by checking out the distribution of number of reviews per unique user. This distribution had a right tail as the number of users that had more than 5000 reviews was small. Less than 500 users out of the >35,000 unique users. The distribution is shown in the logarithmic scaled histogram below.

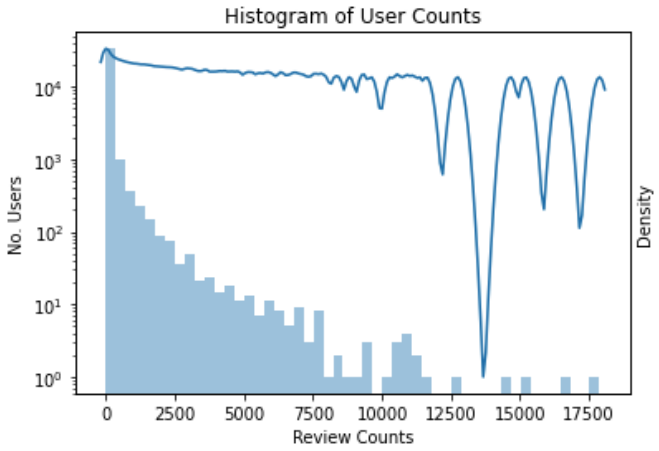

Figure 3: Review Count by User

Another distribution I looked at was of rating which had a pretty normal distribution, slight tailing left. This showed a mean rating of about 3.835. This distribution can be found below.

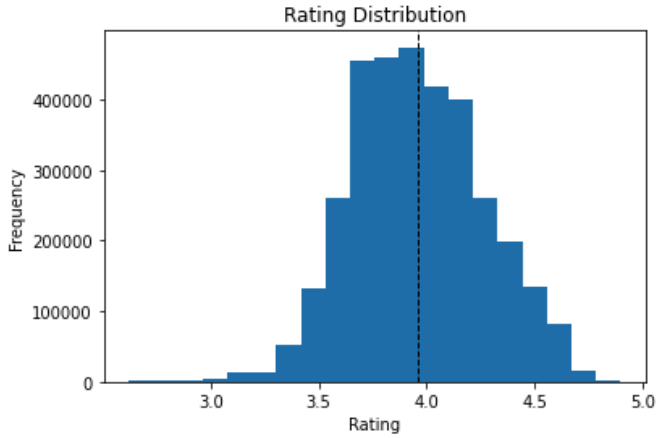

Figure 4: User's Beer Rating Distribution

Model Description, Results, & Insights

Model Description

Once my data was ready for processing I created two models. One was a KNN and the other was a SVD. I used surprise and sklearn to build my models. I used two different libraries because I wanted a little exposure to both. Surprise builds on top of sklearn so using them should be fairly similar.

Surprise has many different types of models and many different KNN models to choose from but because of the scope of the project and time I kept it simple with surprises KNNBaseline models. The KNNBaseline model is a basic collaborative filtering algorithm which uses a baseline rating. Within the model I implemented the default k = 40 neighbors and used Pearson correlation between all pairs of users with the baselines rather than means. I chose these settings not because of their obvious superiority but since they were advised default parameters by surprise.

Sklearn also has many different types of models and many different SVD models to choose from. I decided to use a TruncatedSVD because it performs linear dimensionality reduction by means of truncated singular value decomposition. So basically the data doesn't get centered before computing the SVD. Allowing it to work with sparse matrices.

Model Results & Insights

The KNN model was validated with 5-fold cross validation using both RMSE and MAE as measurements of success. The RMSE of 0.1317 and the MAE of 0.0547 shows a low error for such an unreliable rating system of 1-5. The results are in the figure below.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE	0.1319	0.1321	0.1314	0.1316	0.1315	0.1317	0.0003
MAE	0.0548	0.0549	0.0546	0.0547	0.0547	0.0547	0.0001

Figure 5: 5-Fold Cross Validation

The KNN Baseline takes a sparse matrix and associates missing values with a baseline estimate. This is basically a road map matrix to see how every user will rate every beer. After the road map is drawn, the algorithm is fast because it's now just a simple look up and measure the distance between ratings to find the nearest neighbors.

As for the SVD, I needed to find the number of components. I ended up running a quick explained variance test to see what n_components I should probably use to best explain my variance. The test results are below but I ended up continuing with 900. The scatter plot below shows the percentage of explained variance by component count.

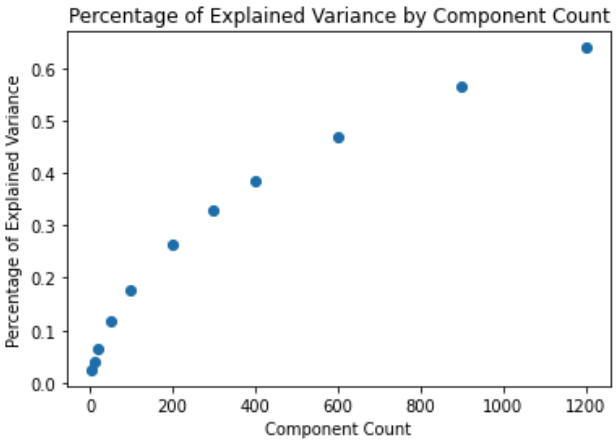

Figure 6: Percentage of explained Variance by Component Count

The last step I took was to compare and combine the two models. To do this I ranked the beers by rating and then name to kinda randomize similarly rated beers. I then took every 84 beers in this list. (This number comes from 8413 unique beers divided by 100 test beers equals 84 as my iteration rate.) After I ran each beer through both models I got outputs like in the figure below which links the common beers. The final plot is after shows a each beers results in a scatter plot.

```
1 get_top_n('Samuel Adams Boston Lager', 40)
```

```
['Nugget Nectar',
 'The Abyss',
 'Three Philosophers',
 'Punkin Ale',
 'Traditional Lager',
 "Dragon's Milk",
 'Bomb!',
 'Orval Trappist Ale',
 'Samuel Adams Summer Ale',
 '#9',
 'Ayinger Celebrator',
 'Péché Mortel',
 'Maudite',
 'Mirror Pond Pale Ale',
 'Leffe Blonde',
 'Kellerweis',
 'Raging Bitch',
 'Hop Head Red Ale',
 'Vitus',
 'Old Stock Ale',
 'Banana Bread Beer',
 'Troegenator',
 'The Sixth Glass Quadrupel Ale',
 'Best Brown Ale',
 'Hopback Amber Ale',
 'Flipside Red IPA',
 'Warlock',
 'Tap 9 Aventinus Eisbock',
 'Gonzo',
 'East IPA',
 'London Porter',
 'Red Chair NWPA',
 'Pannepot',
 'Blushing Monk',
 'La Trappe Quadrupel (Koningshoeven / Dominus)',
 'Skull Splitter',
 'Ayinger Bräuweisse',
 'Raspberry Tart',
 'Chainbreaker White IPA',
 'India Ale']
```

```
1 svd900_top_n('Samuel Adams Boston Lager', 40)
```

```
['Fat Tire Amber Ale',
 'Negra Modelo',
 'Samuel Adams Octoberfest',
 'Hoegaarden Original White Ale',
 'Goose IPA',
 'Anchor Porter',
 'Traditional Lager',
 'Dead Guy Ale',
 'Guinness Foreign Extra Stout',
 'Kalamazoo Stout',
 'Anchor Steam Beer',
 'Snowdrift Vanilla Porter',
 'Samuel Adams Cold Snap',
 'Samuel Adams Winter Lager',
 'Born Yesterday',
 'Pilsner Urquell',
 'Hofbräu Original',
 'Blue',
 'Baltimore Blonde',
 'Edmund Fitzgerald',
 'Ten FIDY',
 'Bigfoot',
 'Solid Gold',
 'Hazelnut Brown Nectar',
 "Smithwick's Imported Premium Irish Ale",
 'Sublimely Self-Righteous Black IPA',
 'Midnight Lager',
 'Boddingtons Pub Ale',
 "Dragon's Milk",
 'Erdinger Weissbier (Hefe-Weizen)',
 'Xocoveza',
 'Blue Moon Harvest Pumpkin Wheat',
 'Samuel Adams Summer Ale',
 'Taddy Porter',
 'Black Butte Porter',
 'Samuel Adams Alpine Lager',
 'Hefe-Weissbier Naturtrüb',
 'Dirty Bastard',
 'Grolsch Premium Lager',
 'Prima Pils']
```

Figure 7: KNN Results on the Left vs SVD results on the Right

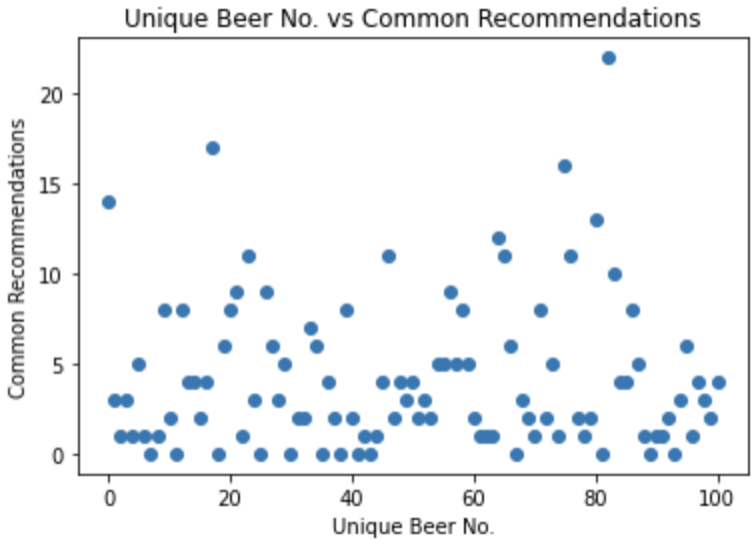

Figure: Unique Beer No. vs Common Recommendations

Conclusion

The surprise and sklearn libraries make building a recommendation system quick and easy. There a ton of different algorithms and all have their own use cases. We used KNN for a nearest neighbor approach and SVD for a latent factor approach. As individuals that did okay with recommending beers to users. The real power was when I combined the two to find common beers. These beers were normally more relevant to the users preference.

Next Steps

1. Build models using more advanced techniques.
2. Build a user interface.
3. Automate data collection on a schedule for new data being added daily.
4. Incorporate other ranking systems and NLP review data from breweries, beer advocates, and Twitter.