Capstone Project 2 Report: Beer Recommendation System

The goal of this project is to create a collaborative filtering based beer recommendation system based on 1.5 million reviews from beeradvocate. This system will return 10 new beer choices to the user based on similarity to other user's tastes. I will also examine trends in the data to determine which style of beer is most popular, what the most highly rated breweries are, and what features are most important in highly rated reviews.

My target audience for this project is first and foremost craft beer lovers. The recommendation system will be an easy way to discover new beers to try. It is also targeted for beer review sites such as beeradvocate, untappd, and ratebeer, all which (in my opinion) could benefit from a recommendation system that allows their users to discover other beers similar to ones they enjoyed.

This project was also partially inspired by Tanya Cashorali's great article "How to Hire and Test For Data Skills: a One Size Fits all Interview" (https://www.linkedin.com/pulse/how-hire-test-data-skills-one-size-fits-all-interview-tanya-cashorali/). I incorporated some elements of the questions asked in the article into my analysis.

Data Cleaning:

My dataset for this project consists of reviews web scraped from BeerAdvocate, originally hosted on the Stanford SNAP web data page. It has since been removed from that location and rehosted on Kaggle. The dataset contains 13 columns:

1. Brewery ID - an integer identifier for each unique brewery
2. Brewery Name - a string containing the name of the brewery
3. Review Time - an integer containing the date the review was submitted
4. Review Overall - a float of the complete review score. Ratings in this dataset are from 1-5
5. Review Aroma - a float of the review score for the aroma of the beer
6. Review Palate - a float of the review score for the mouthfeel of the beer
7. Review Appearance - a float of the review score for the appearance of the beer
8. Review Taste - a float of the review score for the taste of the beer
9. Review Profilename - a string containing the reviewer's username
10. Beer Name - a string of the reviewed beer's name
11. Beer ID - an integer identifier for each unique beer
12. Beer Style - a string containing the style of the beer
13. Beer ABV - a float denoting the reviewed beers percent alcohol by volume

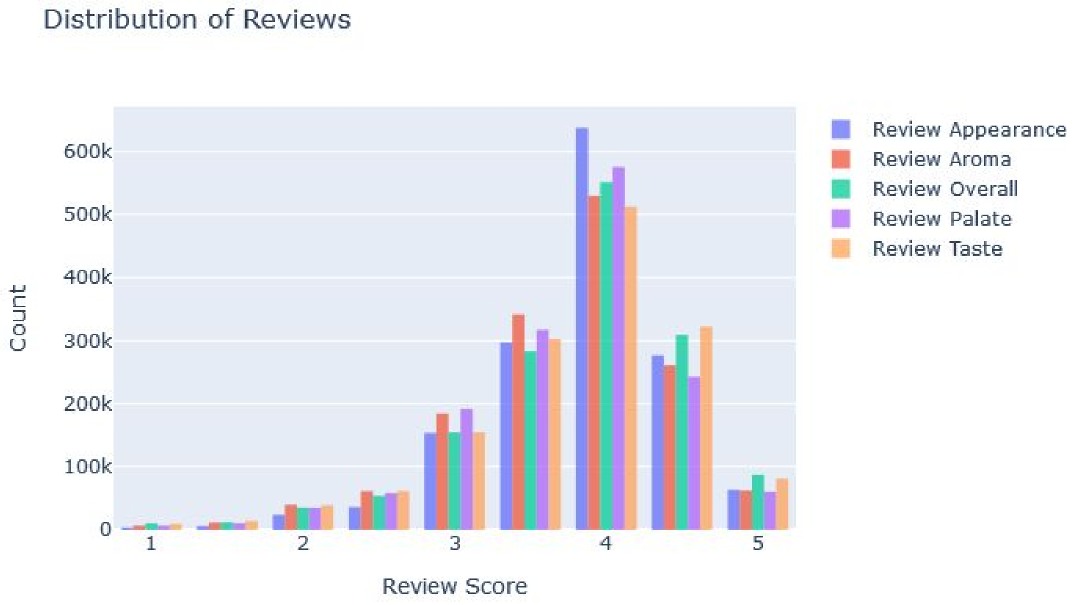
My first step in cleaning the data was to convert the review time column into a datetime object, as the original integer values were hard to interpret. After this, I inspected the data using the pandas .info() method. There are 1,586,614 total reviews in this dataset, but this inspection showed around 68,000 missing values, mostly from the Beer ABV column. Since the dataset is so large, I decided to just drop these missing values as that still left over 1.5 million reviews to work with.

The next step in cleaning the data was to drop all duplicate reviews. There were a few instances where one reviewer had multiple reviews for the same beer. I dropped the duplicates, keeping the most recent review in the dataset.

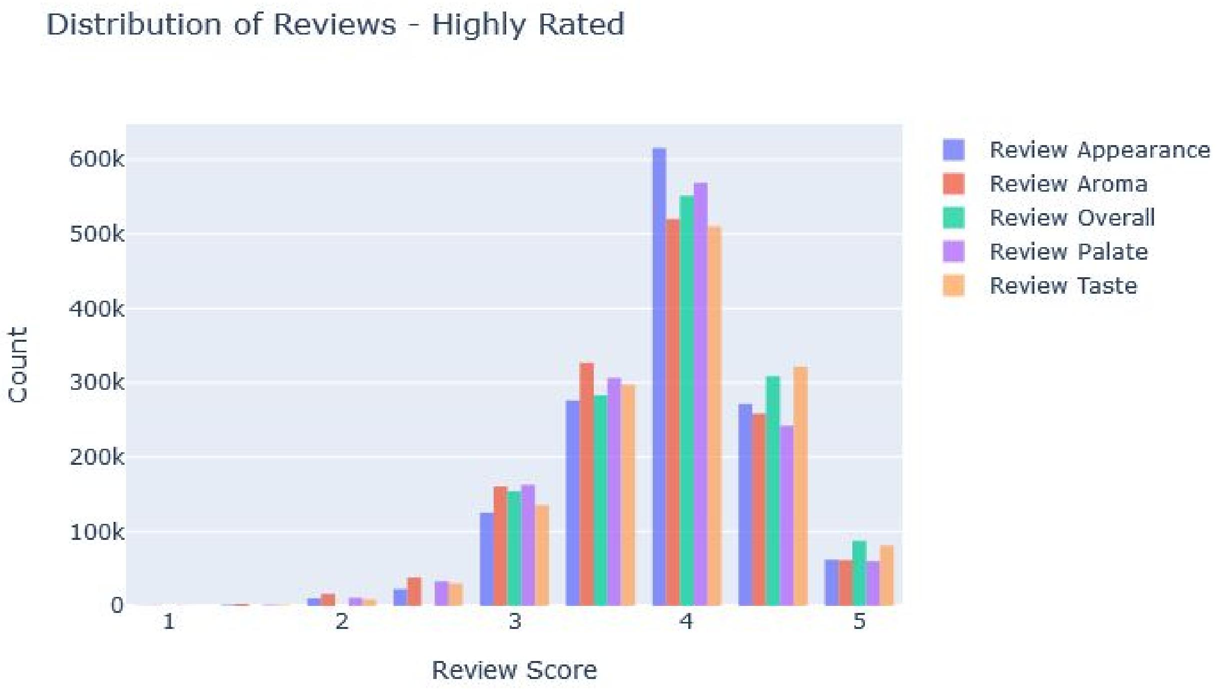
The final cleaning I performed was to drop all reviews of less than 1. The beer advocate system uses 1 as the lowest possible review score. I decided to drop these reviews altogether. After performing these steps the dataset contains 1,496,256 total reviews.

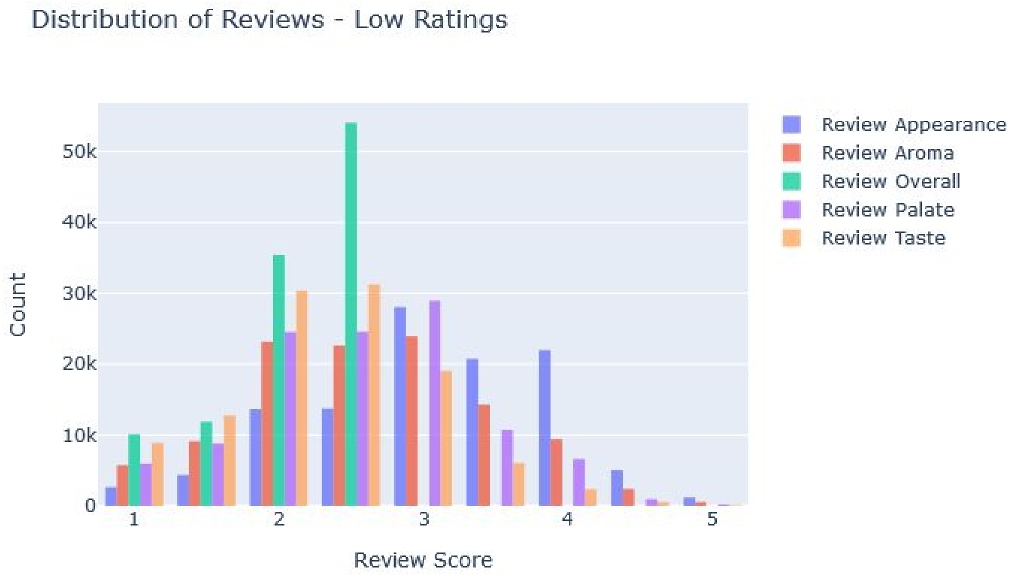
Exploratory Data Analysis

I began exploratory data analysis by checking the number of unique styles, breweries, and beers represented in the dataset. There are 5,135 breweries, 104 beer styles, and 44,0775 beers reviewed.

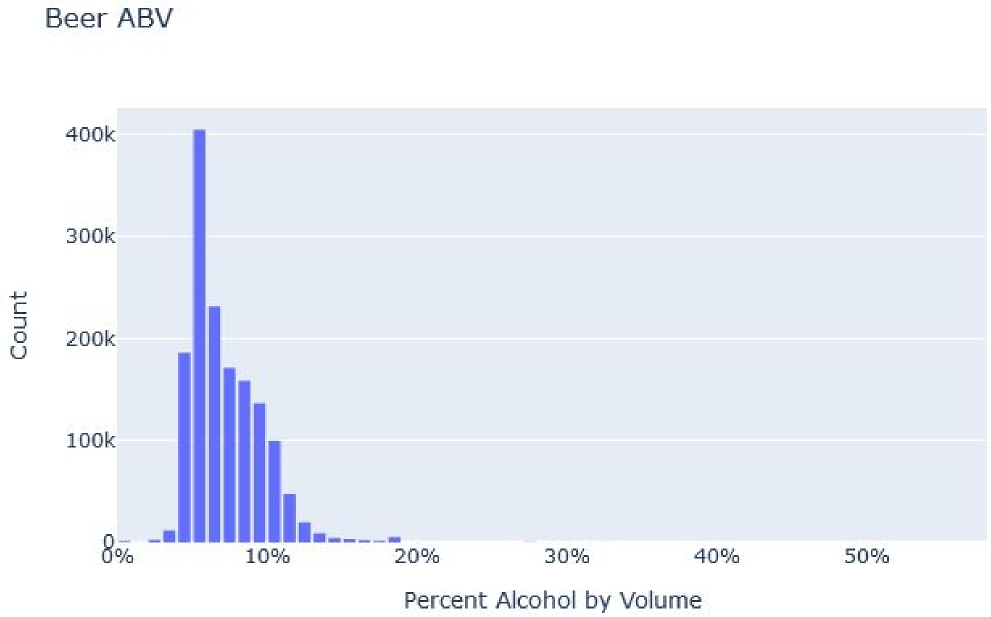


The most common review score across this dataset is 4 in all review categories. Overall there are far more positive than negative review scores exhibited.

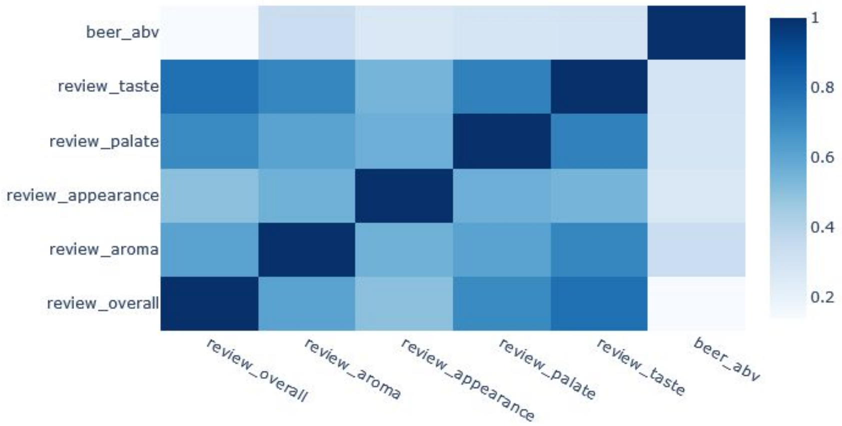

Distribution of Reviews - Highly Rated


Distribution of Reviews - Low Ratings

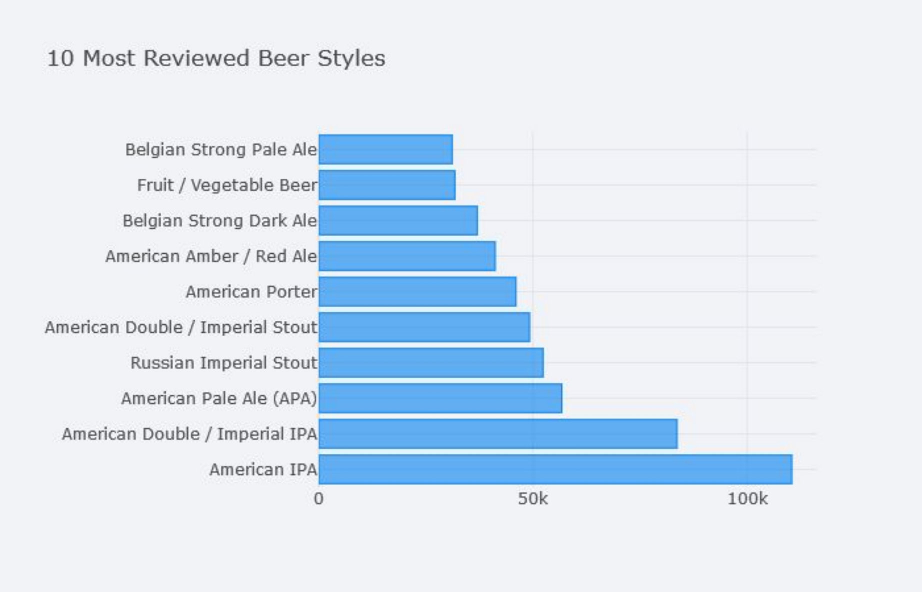
These are the same distributions separated by reviews over 3 as highly rated and reviews below 3 as low ratings. It appears that review_taste tracks most closely with review_overall, suggesting that taste may be the most important factor to a good review. Review_appearance and review_aroma appear to be more independent of review overall.



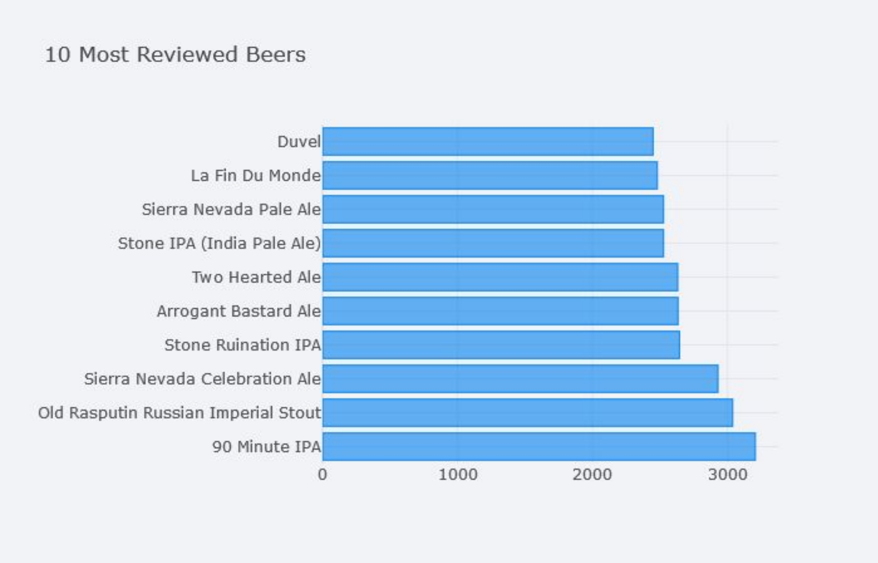
The distribution of ABV is centered from 5 - 5.99%, with most of the data falling between 4-12%. Beer typically ranges from 4-13% ABV.



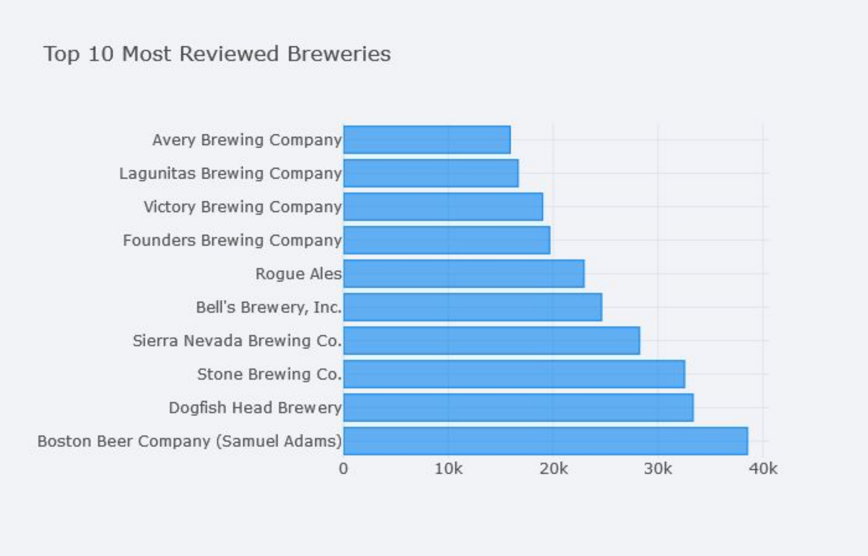
This heatmap displays the correlation between numerical columns of the datest. review_taste has the strongest correlation with review_overall, confirming what I noticed when looking at the review distribution earlier. Review_palate has the second strongest correlation, followed by review_aroma and review_appearance. Beer_abv has almost no correlation whatsoever. It seems that taste and palate (mouthfeel) are the most important factors in a beer rating.



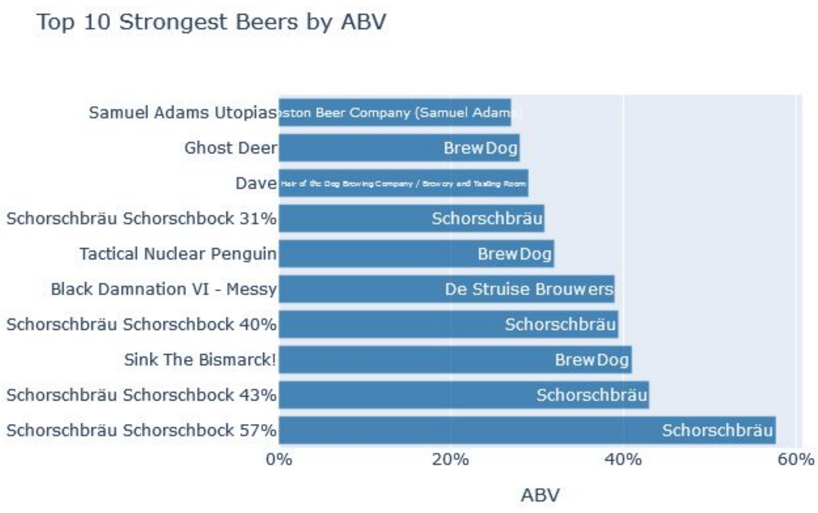
The most commonly reviewed beer is clearly the American IPA, with over 100,000 reviews. Interestingly over half of the beers in the top 10 are American styles. While we don't have data on user location, I would guess that most of the users in this dataset are American. The least reviewed beer styles are Kvass and Happoshu, both of which are beer-like beverages but arguably not actual beer. The next least reviewed style is Roggenbier, a medieval style rye beer. The average amount of reviews per style is 14,387.



The Dogfish Head 90 Minute IPA is the most frequently reviewed beer in the dataset, with over 3000 reviews. The average amount of reviews per beer is 33.9.



Samuel Adams has the highest review count out of any brewery. The average amount of reviews per brewery is 291.



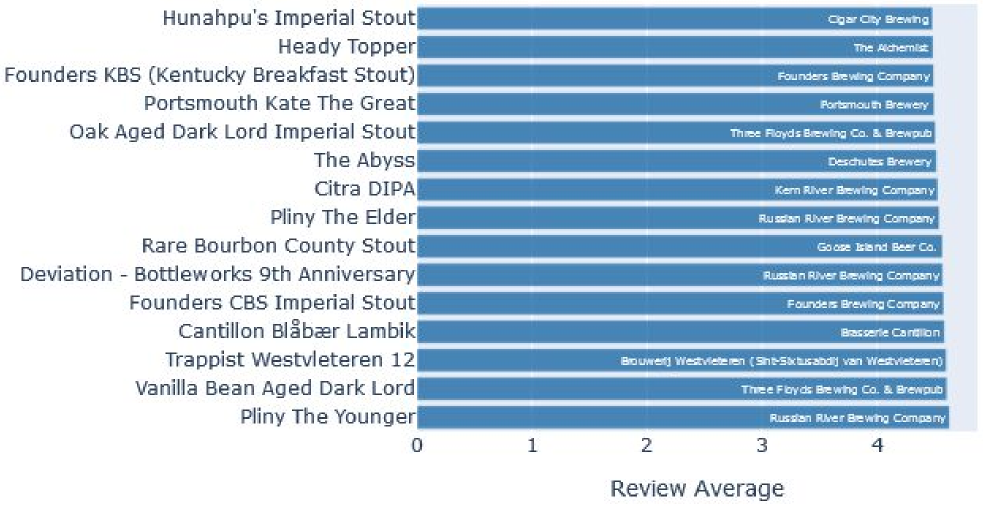
The strongest beer represented here is the Shorschbrau Shorschbock at 57% alcohol by volume, stronger than most hard liquors! The lowest ABV is LIber at 0.01%. Most beers in the dataset are around 5% ABV.

Best Beers, Styles, and Breweries

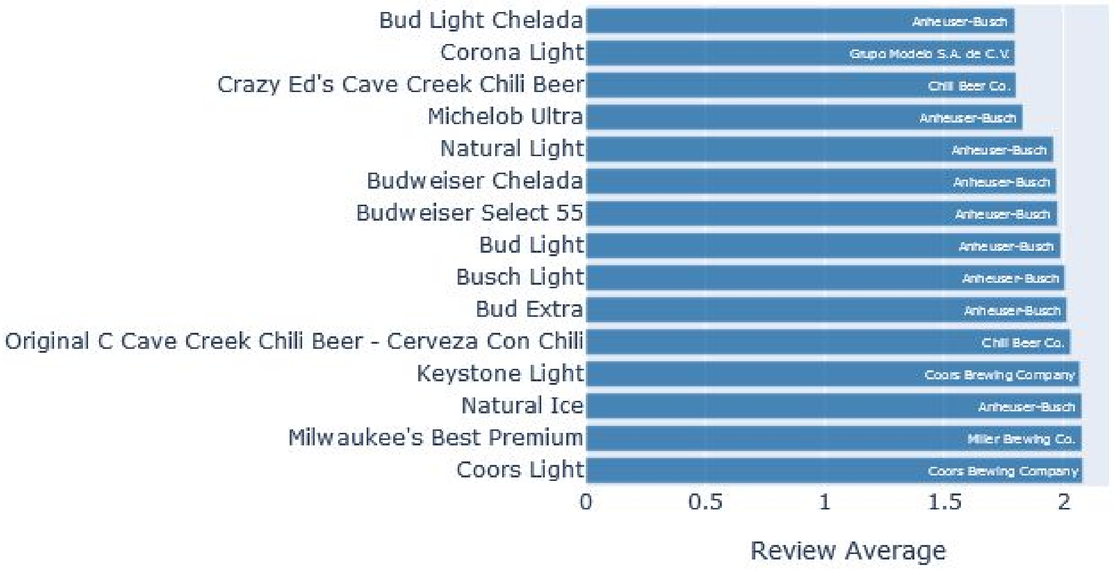
While previously I examined the most frequently reviewed beers, styles, and breweries, only looking at the number of times an item has been reviewed is not the best metric for quality, so I decided to filter by highest mean review_overall. Immediately I noticed a problem with this approach - my results were thrown off by beers, styles, and breweries with a low number of ratings. Having a mean review score of 5.0 isn't helpful if there is only one review!

To address this problem, I first created a new “review_average” column, which takes the average of review_taste, appearance, palate, aroma, and review_overall for each entry in the dataframe. I then created the “total_reviews” column, which counts the total number of times an item has been reviewed. Next I aggregated the dataframe to have the columns as unique beers, with the rows populated by the mean values for that beer. After this, I removed all entries which had been reviewed less than 100 times, and sorted by mean review_average. This created an effective way to evaluate rating quality on only beers with a significant amount of reviews.

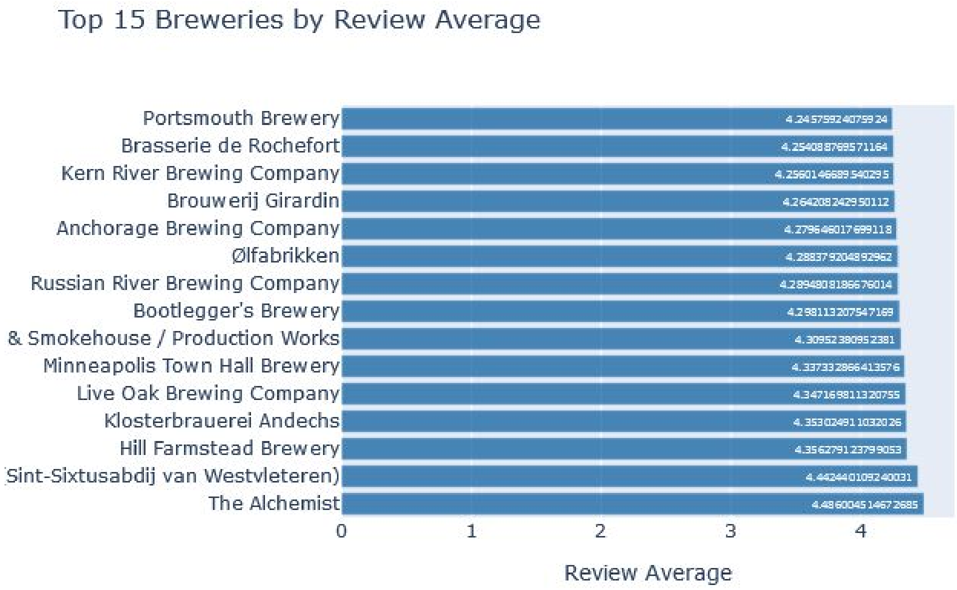
Top 15 Beers by Review Average

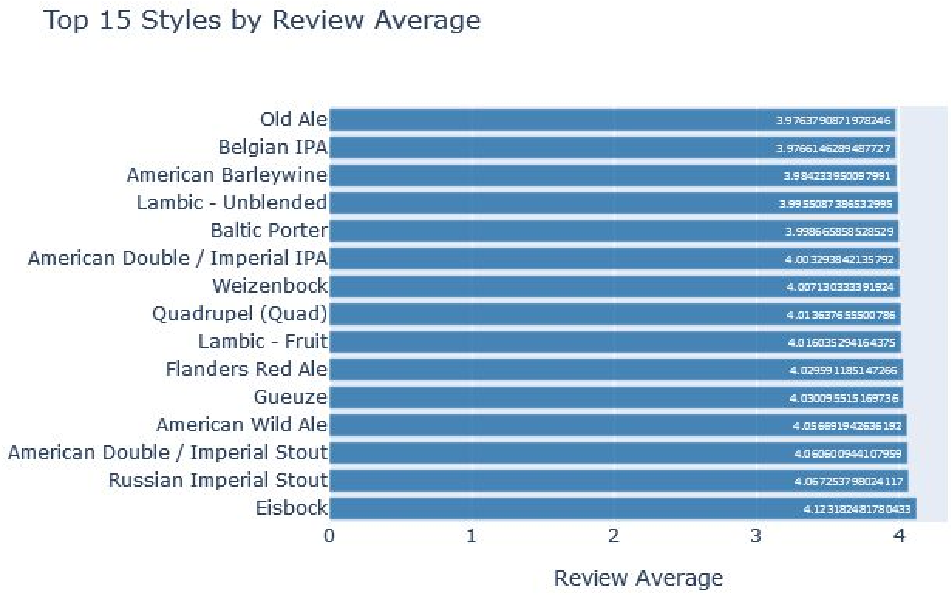


Bottom 15 Beers by Review Average



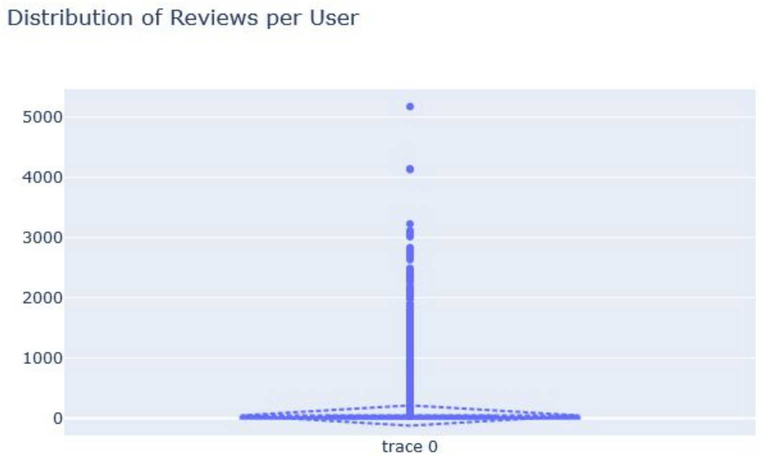
These results show that the lowest rated beers in the data set are almost all mass-produced light beers by large breweries, especially Anheuser-Busch. BeerAdvocate users definitely have a strong preference for craft beer over macro-brewed.





User Statistics

The dataset contains reviews from 32908 distinct users. The mean amount of reviews per user is 45.4, though this is a slightly misleading statistic. The median reviews per user is 3. There is an outlier subset of highly prolific users who review very frequently, while most users have reviews in the single digits.



This boxplot illustrates the disparity between the average user and high-review users. The upper fence of the interquartile range is 38 reviews, meaning that 75% percent of users in the dataset have 38 or fewer reviews. User northyorksammy has the highest number of reviews, reviewing over 5,000 beers!

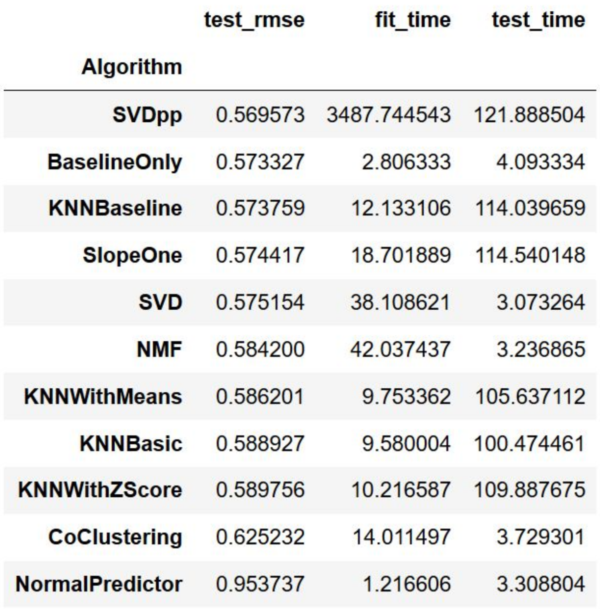
Building a Recommendation System

This recommendation system will use collaborative filtering - meaning it recommends items to users based on other users with similar tastes. As I discovered in exploratory data analysis, the dataset contains many beers with a low amount of ratings. For the recommender system I decided to only include beers which have been rated at least 50 times. This will help avoid the algorithm recommending too many obscure or hard to find beers. This filtering reduced the number of reviews by around 420,000, which still left over 1,000,000 reviews.

To create the first version of the recommendation system, I made a new dataframe from the filtered data containing only the user name, beer name, and review overall columns. I then ran an initial fit using an out of the box, untuned SVD (singular value decomposition) model from the Surprise recommender library, splitting the data with a test size of 0.25. This model achieved an RMSE of 0.575 without tuning any hyperparameters. I then examined the first 5 predictions on the test set to make sure everything looked good:

```
[Prediction(uid='Long813', iid='Hophead Double India Pale Ale', r_ui=4.0,
est=3.585667317417661, details={'was_impossible': False}),
 Prediction(uid='beerwolf77', iid='Orchard White', r_ui=4.0, est=4.007106415545197,
details={'was_impossible': False}),
 Prediction(uid='SShelly', iid='Green Flash Le Freak', r_ui=4.0,
est=4.131120724067257, details={'was_impossible': False}),
 Prediction(uid='maxpower', iid='Lump Of Coal', r_ui=3.5, est=3.539782794389326,
details={'was_impossible': False}),
 Prediction(uid='Cs1987', iid='König Pilsener', r_ui=4.0, est=3.6295640690908293,
details={'was_impossible': False})]
```

I was surprised at the out of the box accuracy of this model. The next step was to iterate over a list of possible algorithms to see which one performs best on the data. I tested the SVD, SVD++, SlopeOne, NMF, NormalPredictor, KNNBaseline, KNNWithMeans, KNNWithZScore, BaselineOnly, and CoClustering algorithms with 3-fold cross-validation. Inside the loop, I created a dataframe to store the results of each algorithm for evaluation.



	test_rmse	fit_time	test_time
Algorithm			
SVDpp	0.569573	3487.744543	121.888504
BaselineOnly	0.573327	2.806333	4.093334
KNNBaseline	0.573759	12.133106	114.039659
SlopeOne	0.574417	18.701889	114.540148
SVD	0.575154	38.108621	3.073264
NMF	0.584200	42.037437	3.236865
KNNWithMeans	0.586201	9.753362	105.637112
KNNBasic	0.588927	9.580004	100.474461
KNNWithZScore	0.589756	10.216587	109.887675
CoClustering	0.625232	14.011497	3.729301
NormalPredictor	0.953737	1.216606	3.308804

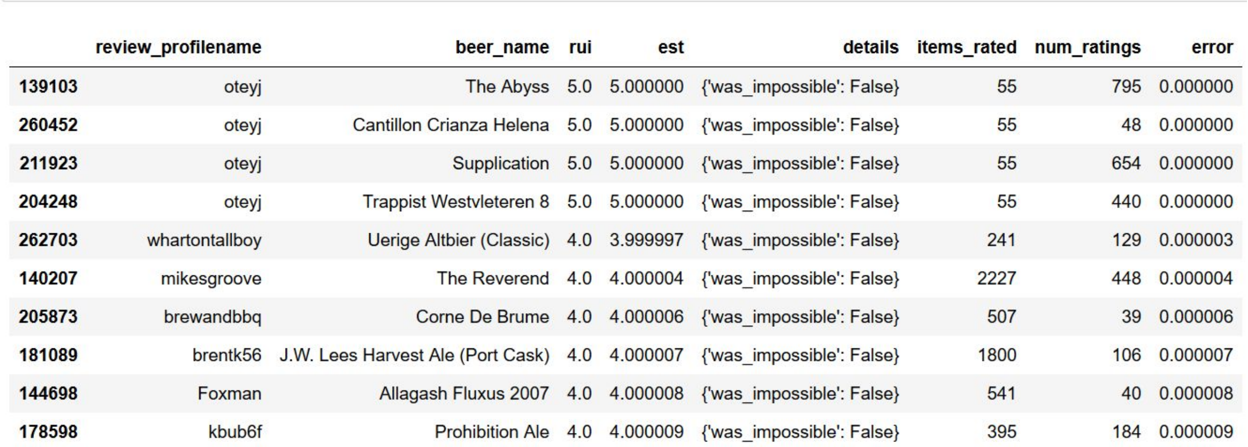
SVD++ has the lowest RMSE by .01%, but also has by far the largest fit time. The other algorithms are fairly close in performance and fit time. The NormalPredictor algorithm, which predicts a random rating based on the distribution of the data, which it assumes to be normal, had the highest RMSE as I expected, giving a good benchmark to evaluate other models by. Based on these results, I decided to move forward with hyperparameter tuning on the SVD++ algorithm.

I initially ran a grid search with 3 fold cross validation over a set of SVD++ hyperparameters to find the best model. After 3 days of fitting, I decided the potential accuracy gain wasn't worth the time it was taking. Given more computing power or time, I would eventually like to revisit this. I decided to do some manual tuning instead.

I created an instance of SVD++ with the following hyperparameters:
(n_factors=150, n_epochs=75, lr_all=0.01, reg_all=0.1)
This model achieved an RMSE of 0.571, slightly outperforming the original SVD model, and significantly outperforming the NormalPredictor.

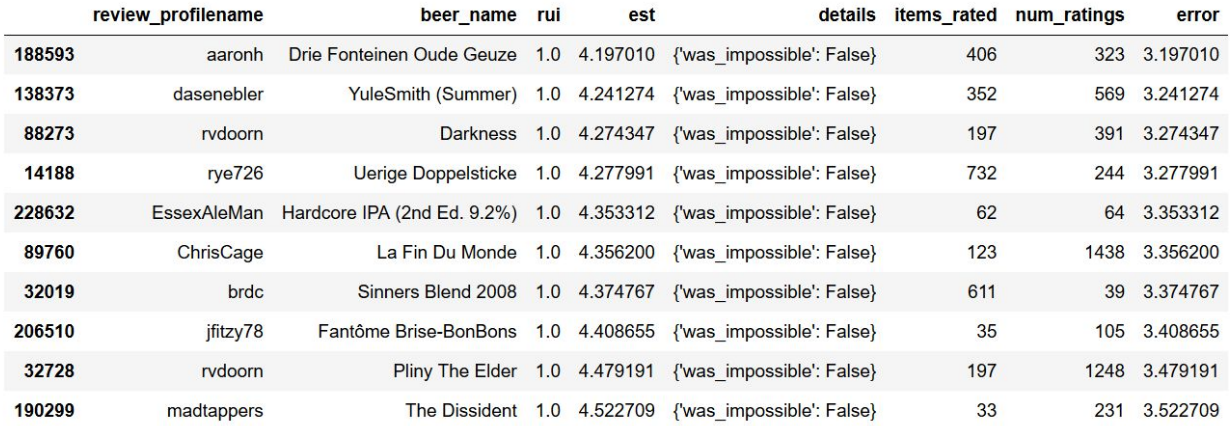
To analyze the predictions of this model, I created a function that returns a dataframe with the prediction metrics, how many times the beer has been rating, and how many items the user has rated. After applying this function to the predictions, I sorted the results by error to examine the best and worst predictions the model had made.

Best Predictions:



	review_profilename	beer_name	rui	est	details	items_rated	num_ratings	error
139103	oteyj	The Abyss	5.0	5.000000	{'was_impossible': False}	55	795	0.000000
260452	oteyj	Cantillon Crianza Helena	5.0	5.000000	{'was_impossible': False}	55	48	0.000000
211923	oteyj	Supplication	5.0	5.000000	{'was_impossible': False}	55	654	0.000000
204248	oteyj	Trappist Westvleteren 8	5.0	5.000000	{'was_impossible': False}	55	440	0.000000
262703	whartontallboy	Uerige Altbier (Classic)	4.0	3.999997	{'was_impossible': False}	241	129	0.000003
140207	mikesgroove	The Reverend	4.0	4.000004	{'was_impossible': False}	2227	448	0.000004
205873	brewandbbq	Corne De Brume	4.0	4.000006	{'was_impossible': False}	507	39	0.000006
181089	brentk56	J.W. Lees Harvest Ale (Port Cask)	4.0	4.000007	{'was_impossible': False}	1800	106	0.000007
144698	Foxman	Allagash Fluxus 2007	4.0	4.000008	{'was_impossible': False}	541	40	0.000008
178598	kbub6f	Prohibition Ale	4.0	4.000009	{'was_impossible': False}	395	184	0.000009

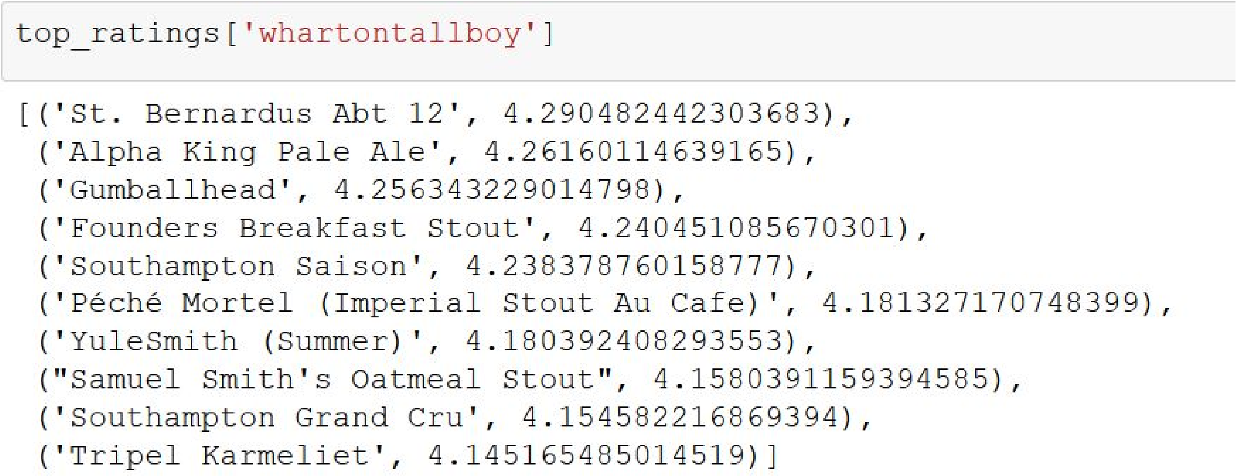
Worst Predictions:



	review_profilename	beer_name	rui	est	details	items_rated	num_ratings	error
188593	aaronh	Drie Fonteinen Oude Geuze	1.0	4.197010	{'was_impossible': False}	406	323	3.197010
138373	dasenebler	YuleSmith (Summer)	1.0	4.241274	{'was_impossible': False}	352	569	3.241274
88273	rvdoorn	Darkness	1.0	4.274347	{'was_impossible': False}	197	391	3.274347
14188	rye726	Uerige Doppelsticke	1.0	4.277991	{'was_impossible': False}	732	244	3.277991
228632	EssexAleMan	Hardcore IPA (2nd Ed. 9.2%)	1.0	4.353312	{'was_impossible': False}	62	64	3.353312
89760	ChrisCage	La Fin Du Monde	1.0	4.356200	{'was_impossible': False}	123	1438	3.356200
32019	brdc	Sinners Blend 2008	1.0	4.374767	{'was_impossible': False}	611	39	3.374767
206510	jfitzy78	Fantôme Brise-BonBons	1.0	4.408655	{'was_impossible': False}	35	105	3.408655
32728	rvdoorn	Pliny The Elder	1.0	4.479191	{'was_impossible': False}	197	1248	3.479191
190299	madtappers	The Dissident	1.0	4.522709	{'was_impossible': False}	33	231	3.522709

I initially expected the worst predictions to be due to users or beers that had low number of ratings, but that doesn't seem to be the case. All of the predictions with the highest error were users who rated a beer as a 1, where the model predicted they would rate it highly. Interestingly several of the beers on this list are present in the top reviewed beers from above. It seems the recommender system has the hardest time with users who did not like a beer which is generally reviewed highly.

While the model does have incorrect predictions like this, overall it performs very accurately. The final step was to create a function that will return the top 10 predictions for any given user in the dataset. This function splits the predictions by user, then sorts by rating. I created a dictionary top_ratings by applying this function to the predictions. This dictionary contains the users as keys, and the top 10 predictions as values. Top_ratings can then be easily filtered by user id. Let's take a look at the predictions for user whartontallboy:



```
top_ratings['whartontallboy']
```

```
[('St. Bernardus Abt 12', 4.290482442303683),
 ('Alpha King Pale Ale', 4.26160114639165),
 ('Gumballhead', 4.256343229014798),
 ('Founders Breakfast Stout', 4.240451085670301),
 ('Southampton Saison', 4.238378760158777),
 ('Péché Mortel (Imperial Stout Au Cafe)', 4.181327170748399),
 ('YuleSmith (Summer)', 4.180392408293553),
 ("Samuel Smith's Oatmeal Stout", 4.1580391159394585),
 ('Southampton Grand Cru', 4.154582216869394),
 ('Tripel Karmeliet', 4.145165485014519)]
```

Overall, the recommender performs very well on the dataset. Given some additional work it could easily be made into a customer facing feature. In the future, I would like to complete the time-intensive grid search to see if the RMSE can be lowered, and experiment with hybrid recommendation approaches to incorporate other features from the dataset into the model. Another next step would be to update the dataset to include more recent reviews from BeerAdvocate, as the data only includes reviews through 2012.