CJ Johanson

Machine Learning Analysis for Beer Recommendations

The goal of this project is to create a recommendation engine using machine learning. This particular system is going to work with data about beer. Although recommendation systems can easily have an impact on many different businesses, for the purpose of the project the client will be an online alcohol sales company whose sales mainly consist of beer that is delivered to the customer's home. This report will explain why recommendation systems are useful, information about the dataset used, details about the exploratory data analysis, and an analysis of both the implemented machine learning techniques and the subsequent findings.

Recommendation systems are so common in people's everyday experience, yet many may not even notice it. The majority of human interactions with websites include, in some way, a recommendation engine. This could include making an online purchase, enjoying content via an entertainment streaming service, or browsing social media. What is most valuable about recommendation systems is, by design, the power of suggestions. The website from which you bought a coat could also suggest some gloves to go with it, an entertainment streaming service could suggest a show that you might enjoy, and a social network could suggest a new connection based on people you already know. A reasonable goal of most websites, and possibly all businesses that also exist in the online space, is to make it as easy as possible for your customers to either spend your money (if you sell goods) or interact more with your product (if you provide a service). Recommendation systems do exactly that. Regarding the examples above, buying those gloves means an increase in sales. For the other two examples, there is not a direct increase in profit, but there is an increase in interaction with the products in question. Such increase in interaction is equally beneficial because it can, for example, create targeted advertisements for the user or collect more detailed data about users that can later be monetized. All of this opportunity for growth is driven by recommendations.

As this is being written, there is a global pandemic due to coronavirus. It is impossible to dispute the gravity of the situation and the impact it has had on the majority of people's lives. One particular trend, which is a direct result of the pandemic, is the overall rise in alcohol sales. This trend is very much worth exploring, especially when it comes to beer. Beginning around the end of March 2020, there have been restrictions in place in terms of where people can go and the size of groups that can gather, for obvious health and safety reasons. Essentially, these restrictions meant that bars and breweries were forced to temporarily close down. However, people continued to purchase beer, maybe even more than before as a way to cope with the new stress and fear from the virus.

As mentioned before, a business should make it as easy as possible for a customer to either spend more money or interact more with your product. Beer is a perfect example of this. If a customer wanted to try something new, all they had to do was go to the bartender and ask for, you guessed it, a recommendation. The bartender would likely ask a couple questions about your

general preferences and then give you a suggestion. Now that the bartender has been temporarily removed from the equation, there is a space in which machine learning can fill the gap, providing users with recommendations. There are two major benefits to this system for the client in question. The first is an increase in sales, a direct monetary value. The second, which may not be as obvious to the reader, is the increased amount of data gained from a recommendation system. That data can be used to look for insights, such as which beers get suggested the most, which style of beer gets suggested the most, and trends in general that can help a business make more intelligent decisions. Another idea, although a little more complicated but potentially beneficial, is to assemble the data in an organized way and monetize it.

The data used for this particular project went through a lengthy process for collection. It is commonplace for people to comment about how much data is out in the open and is easily accessible to anything with an internet connection, all of which is true. What was needed for the recommendation system was a large amount of beer reviews. With the reviews being the most important piece of data to collect, it was also important to collect data about particular beer styles, the individual beers themselves, and even the percentage of alcohol by volume (ABV%). There was not a dataset already in existence that included all of these details. The solution was to write a web scraper in Python to pull this data from one of the most popular beer review websites.

In terms of size, the dataset would be relatively large and the collection process would take days, if not weeks, so performing such a task on an individual computer was not an option. The best solution was cloud computing and storage, which ended up playing a vital role in the process. The Python scripts, which heavily relied on the BeautifulSoup package, were written to collect data about beer styles, beers, breweries, and ratings, with a separate script being written for each. Said scripts were uploaded to an Amazon Web Services (AWS) EC2 instance and executed automatically using cron. As the data was collected, it was periodically written to an AWS Relational Database System. The four resulting tables were “beer_styles”, “beers”, “breweries”, and “ratings.”

The first table, “beer_styles” contained 111 rows. Each row contained a beer style, a beer substyle, and a link to all beers related to said substyle. There were 14 unique beer styles and 111 unique beers styles, so each beer style has multiple substyles. An illustrative example would be the “India Pale Ales” beer style, which contains multiple substyles such as “American IPA,” “Belgian IPA”, or “New England IPA”. The “beer_styles” table was the starting point for the scraping process, which went to the link associated with each substyle to see the beers associated with said substyle.

Logically, the second table was the “beers” table. After going to each substyle beer link from the “beer_substyles” table, the top 50 beers (ranked in descending order by number of ratings) were added to the table. For each beer in the table, the following information was created or scraped: a unique id, beer name, brewery name, beer style, beer substyle, number of reviews, average rating, ABV, link to the individual beer, and link to the brewery. The “beers” table

resulted in around 5,500 unique beers. The “beers” table, in turn, was the starting point to record information about the breweries.

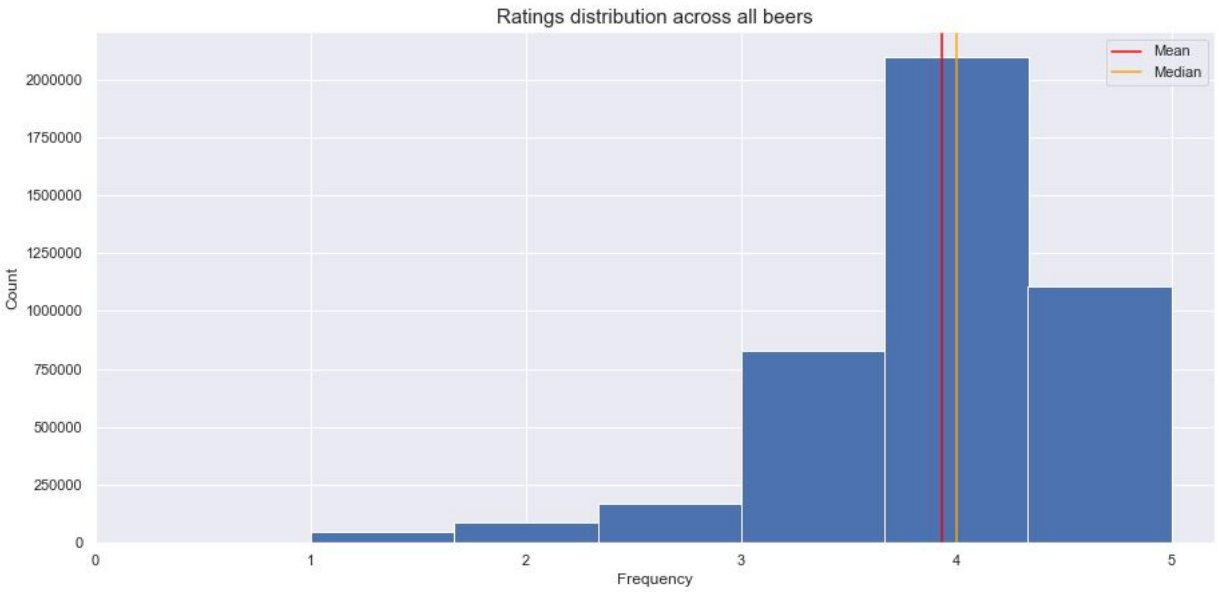
The third table, of course, was the “breweries” table. In order to access information about breweries, each brewery link in the “beers” table was visited and the information was recorded. An empty list was created where, after visiting a brewery page, the brewery’s name was added and the list was used to avoid creating duplicate rows of the same brewery. The data collected included the following: a unique id, brewery name, address, statistics about their beers, and, when applicable, statistics about the brewery location itself. While not exactly relevant to the recommendation system at hand, the “breweries” information could certainly be useful to the client in the future so it was worth including. In the end, the table had around 1,500 unique breweries.

The fourth, final, and most important table was “ratings.” For each beer link in the “beers” table, the web scraper went to each link and recorded all ratings. The data written to the table included the following: a unique id, username, rating date, rating (out of 5), beer name, brewery name, beer style, beer substyle, and ABV. The table resulted in about 5,360,000 reviews. When creating the recommendation system, this table will almost exclusively be relied upon. The large number of reviews will likely prove to be beneficial at the end of the project.

At this point, it is importantly to acknowledge the constructed database as a whole. “Ratings” is certainly the most important table for the current use case, but the beers table will be of some importance, as well. Most of the “beers” table, as well as the “beer_styles” and “breweries” tables, were, essentially, written and recorded with the main goal being the completion of “ratings.” What should not be overlooked is the value found in the last three tables. Although they may not be relevant to the project at hand, there is information that can easily be of value to the client, which could include a time series analysis of beer substyles or breweries, as well as a geographic analysis of where the most popular breweries are physically located. This kind of exploration is absolutely worth pursuing for a later project.

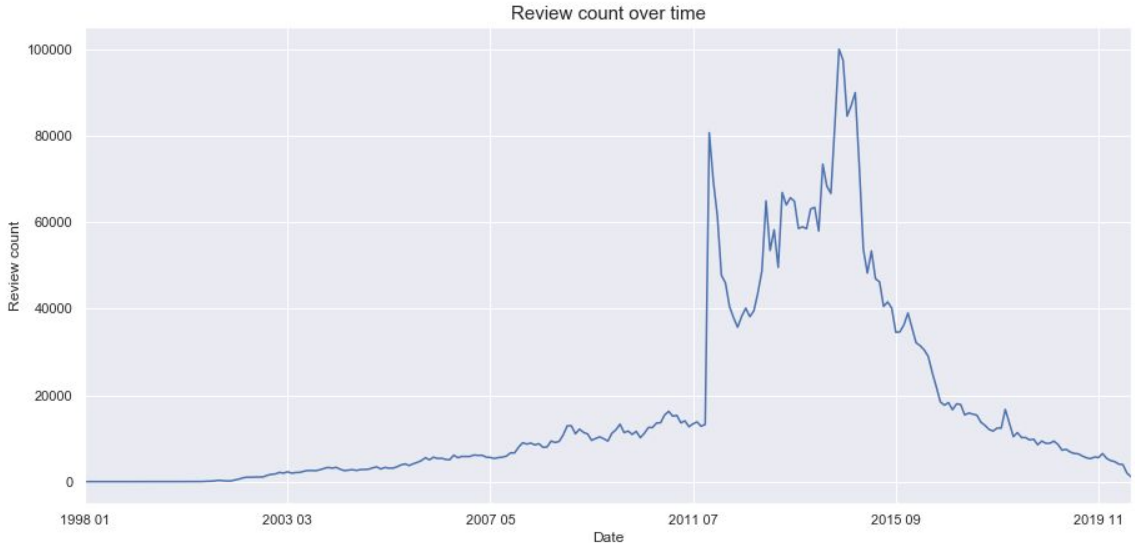
Next, the focus will shift towards cleaning the data. With “ratings” mainly being the only table used for the recommendation system itself, the data loaded into a Pandas DataFrame and explored. Overall, the data was relatively clean and easy to work with because the methodology for scraping and loading in the database was carefully considered. The only step taken in the cleaning process was to change the usernames to an integer user_id in order to be as ethical as possible and avoid displaying the usernames.

The next section of this report will be centered on the summary of findings during the exploratory data analysis process, supported by visuals and statistics. To start, the first finding of interest was the distribution of ratings for all reviews, as seen in the plot below:



Interestingly, the distribution of ratings across all beers is left skewed, with the mean being less than the median. This is a logical finding given that people who participate in a community about beer are likely to enjoy beer, hence the trend towards slightly high ratings.

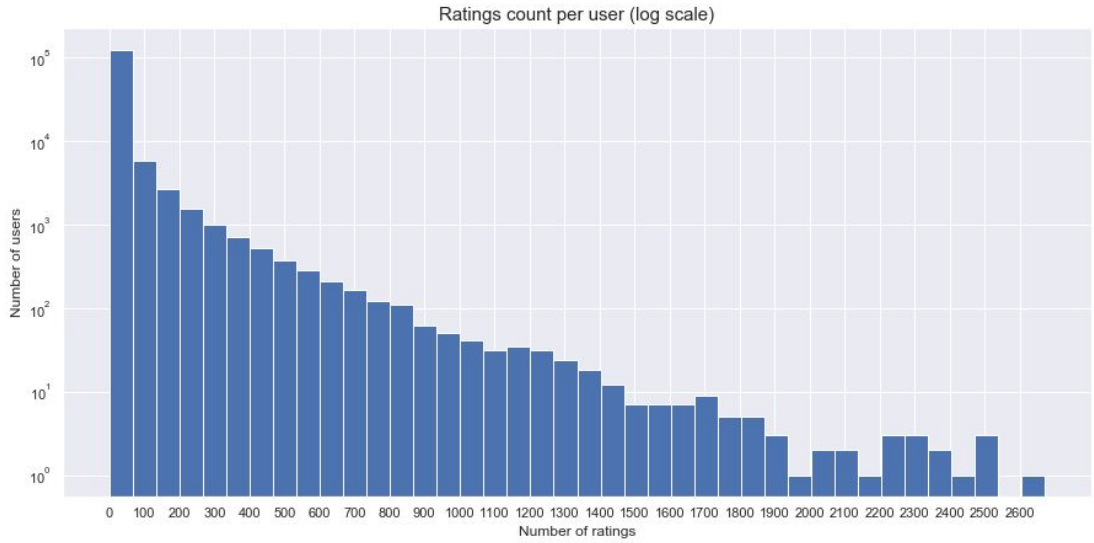
Another intriguing finding came from a quick time series analysis of review counts with the graphic below:



The number of reviews clearly starts to increase around March of 2003, then has a significant spike in July of 2011. During the next four years, there is an overall increase with the peak occurring some time during 2014/15. Since then, the trend has been a steady decrease. Regarding

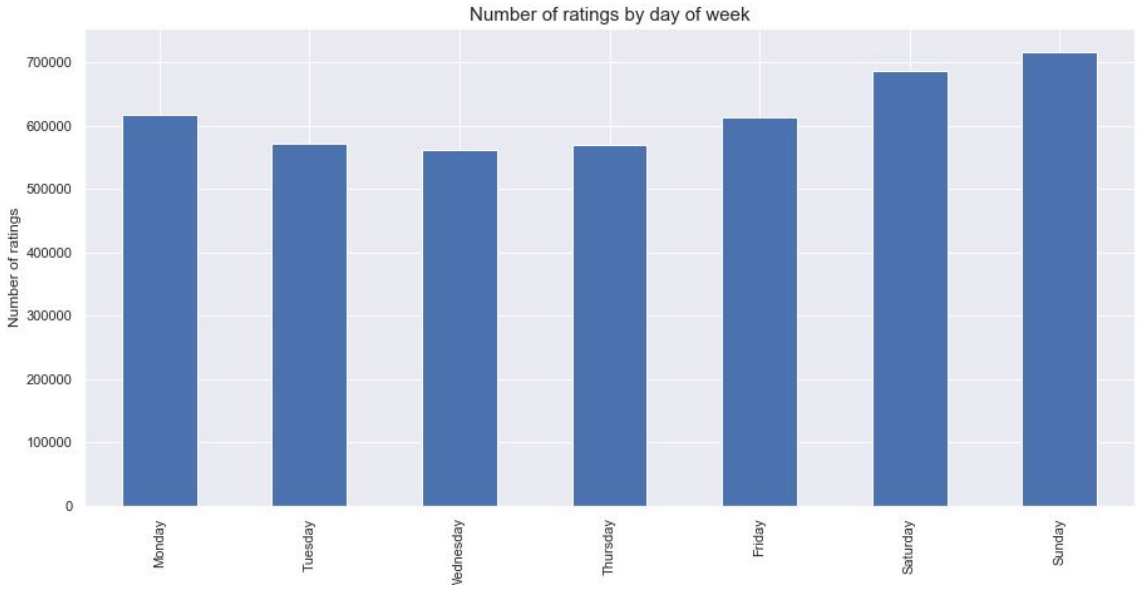
a recommendation system, this should not have an impact because the reviews themselves are important, not when they were written.

The next visualization focuses on how often users write reviews:



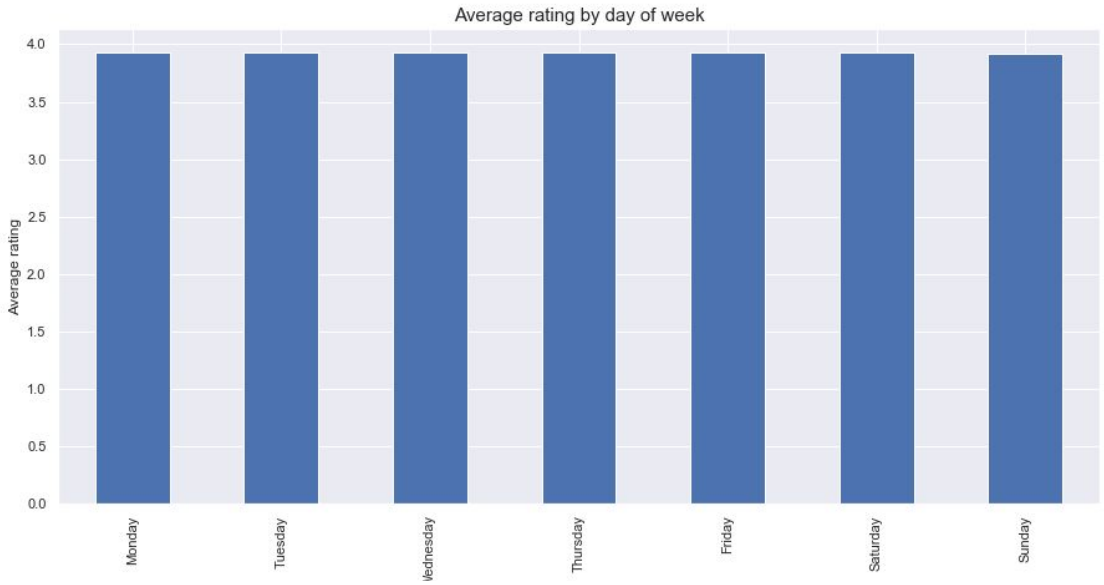
Overall, the vast majority of users write very few ratings, certainly below 100. However, the opposite side of the graph is even more interesting. There is a sizable group of users that write over 2,000 reviews each! Of course, the group only makes a small percentage of the population, but that level of engagement is impressive.

Again observing patterns regarding reviews and time, this visualization is concerned with the number of reviews written in terms of the day of the week. The initial idea was that most reviews are written during the weekend, since most people work during the week and have the weekends off.



The initial idea turned out to be somewhat true. A key difference is that Monday has a slightly higher number of reviews written compared to Friday.

The last finding continues with an analysis based on days of the week. In this case, average ratings were observed according to the day of the week. An initial assumption was that weekends could have higher average ratings. Perhaps, people who drink beer wait until the weekend to drink beers they're excited about. Or, maybe, after that have had those beers, the alcohol might influence how high they rate said beer.



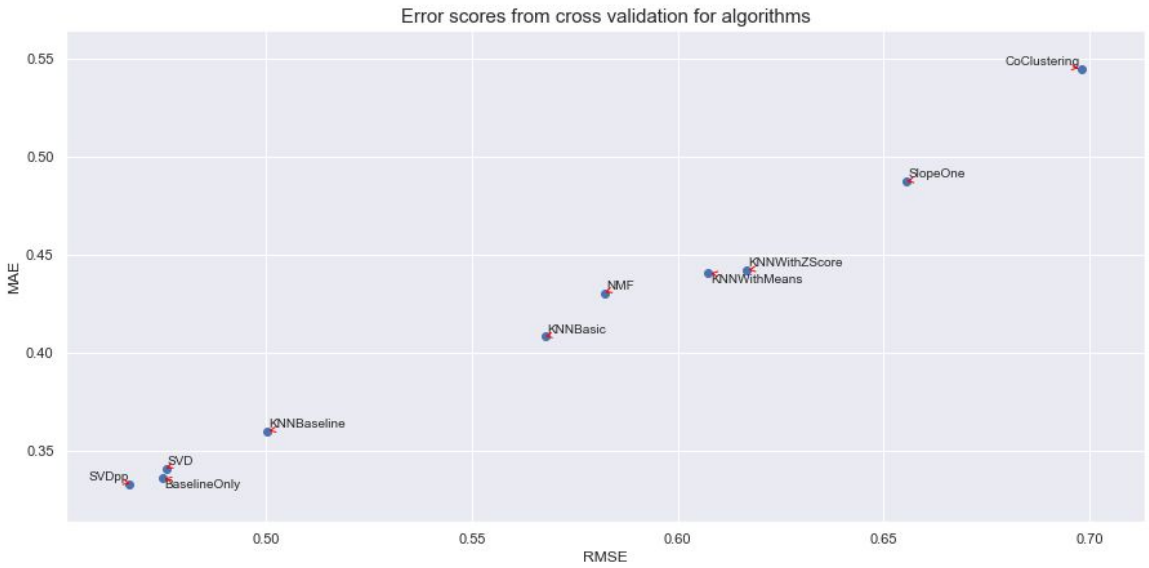
The visualization clearly proves that idea to be incorrect. Surprisingly enough, beer ratings are uniformly distributed across all days of the week.

For the machine learning portion of the project, the surprise library was heavily relied upon. Item-to-item collaborative filtering was used to build the recommendation engine, which made the surprise library a great fit. Per usual, the first thing to do was to import and preprocess the data. The data was loaded in a pandas DataFrame and preprocessing began with creating a new column that was a concatenation of the beer name and brewery name. This ensured that there would be no confusion about certain beers that share the same name, as seen with beers named "IPA." Although IPA is a style, a significant number of breweries would name a beer "IPA," so the new concatenated column was important to create.

Next, the data was randomly shuffled. Then, two smaller subsets of the data were created, the first with 500,000 ratings and the second with 100,000 ratings. The smaller subsets of the data were a bit unusual but necessary since there was limited computing power available for the project . The data was then loaded into a full length dataset set, one with 500,000 ratings, and another with 100,000 ratings.

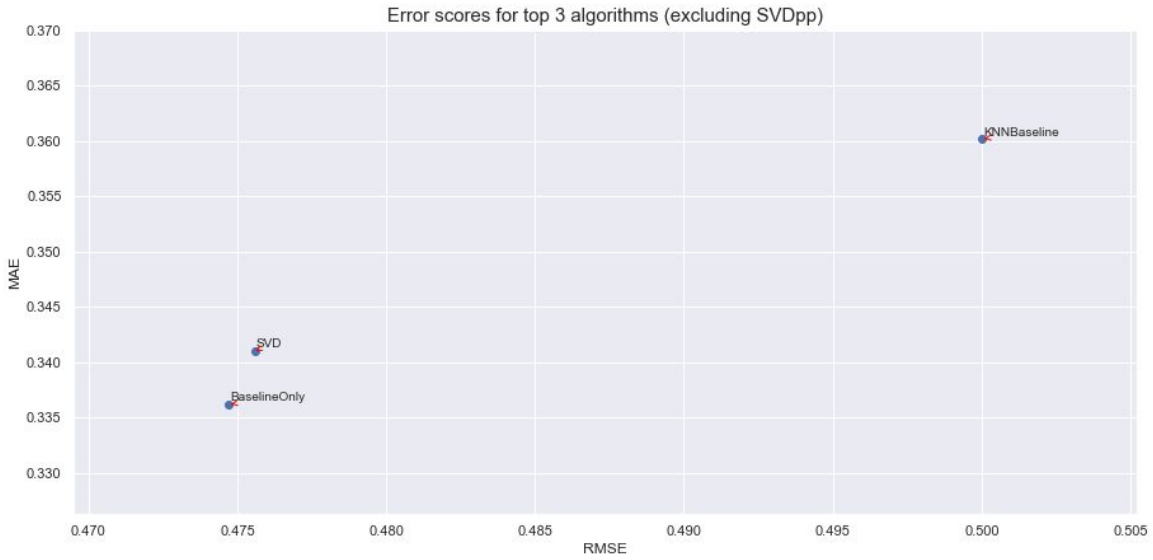
The first analysis focused on which algorithms would perform the best. There are, of course, famous algorithms like Singular Value Decomposition, but it was important to start any

machine learning by looking at the performance of all 9 available algorithms. Each algorithm was cross validated, resulting in the following visualization:



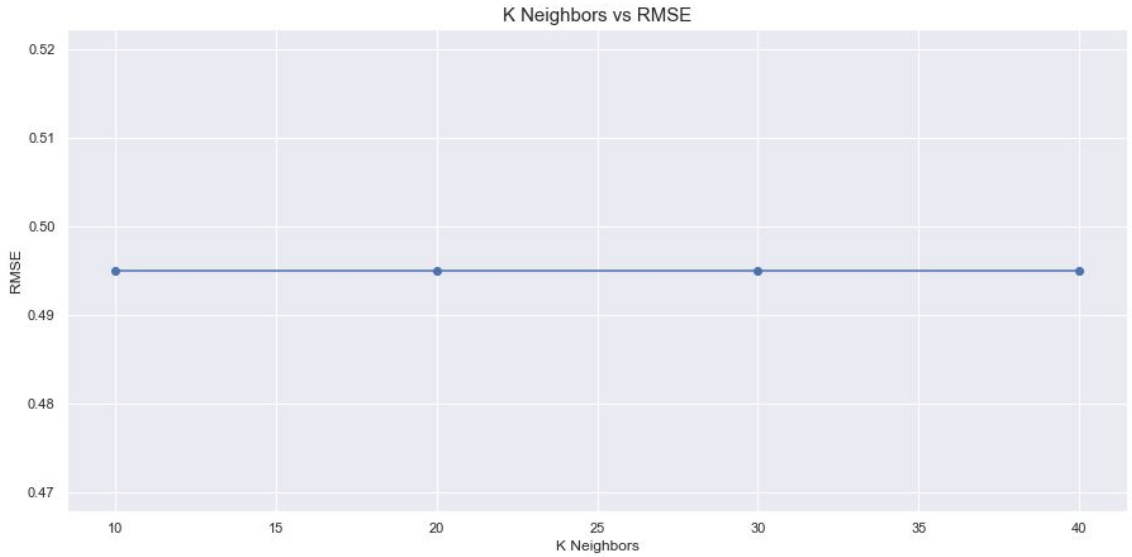
Particularly astute observers will notice that NormalPredictor is missing from the group of 11. NormalPredictor does not have any actual predictive capability and is used more for comparative purposes. BaselineOnly is also an algorithm that does not have any predictive capability, but it was included to provide a baseline estimate. With the x axis and y axis representing RMSE and MAE, respectively, the best performing algorithms are in the bottom left corner. The top 3 algorithms are the following in order of performance: SVDpp, SVD, and KNNBaseline.

SVDpp, although it had great performance scores, has to be removed from consideration because it is extremely computationally expensive, and, as mentioned before, there was limited computing power at hand for the project. Even though it was removed, it is important to note that if computing power were not an issue, SVDpp would likely be the chosen algorithm. This left SVD, and KNNBaseline, so their results were re-plotted to give a better understanding of how the three compare:



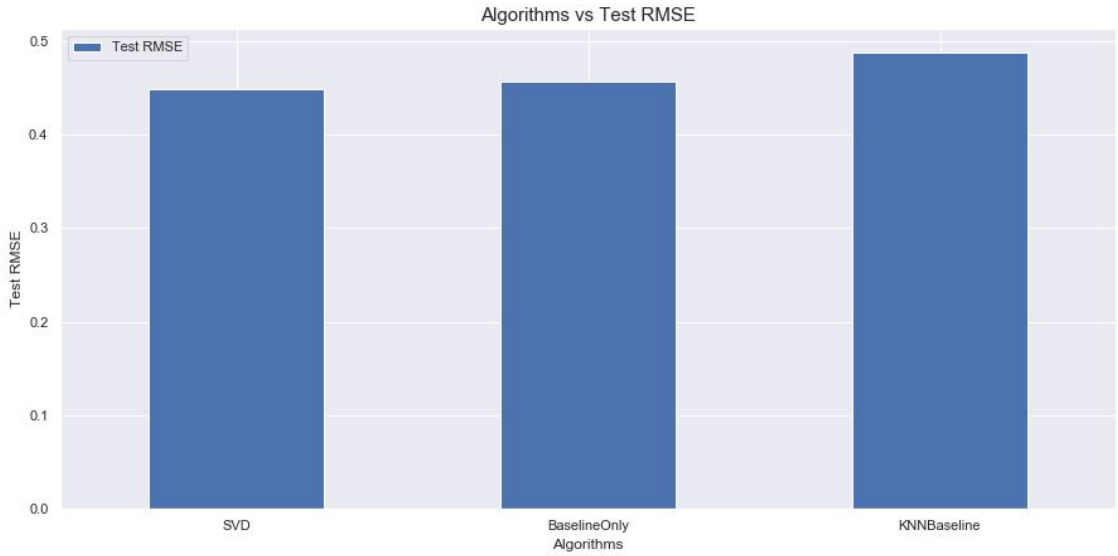
It is important to keep in mind that BaselineOnly is not a predictive algorithm and is only used for comparative purposes. With that said, there is a very clear difference between SVD and KNNBaseline, with the former having significantly better results than the latter. The two algorithms will be hypertuned to find the best estimator, with the likely result being that SVD has the best performance of the two.

GridSearchCV was used to hypertune the two remaining algorithms. For SVD, the best number of latent features was 100. For KNNBaseline, the best k number of neighbors was 10:



As seen in the visualization, the RMSE for KNNBaseline did not vary much across a varying k number of neighbors and was constant around 0.495. It is important to note that both SVD and

KNNBaseline were grid-searched using the 100,000 sized sample of ratings because of the computational cost. Although using the full dataset could have been better, the smaller sample was effectively in displaying the difference in performance, which can be seen in the following visualization:



The best performing algorithm was SVD because it had the lowest RMSE, so it was chosen as the final algorithm used to make predictions.

When it comes to making actual predictions with the recommendation engine, it is important to briefly explain how similarity was measured and to introduce a few beers that will be used to get an understanding of similarity. Cosine distance was used to determine item similarity. The range for cosine distance is 0 to 2. To get a 0, the two items being compared have to be identical, so similarity increases as cosine distance decreases, while similarity decreases as the cosine similarity increases. Some examples will help clarify this. Budweiser is a popular beer that most people have heard of. When comparing the similarity between Budweiser and itself, the cosine distance is 0, meaning the items are perfectly similar. This makes sense because two identical beers are completely similar. Bud Light is, of course, similar to Budweiser, so when the two are compared, the cosine distance is about 0.25, which also makes sense. To put the Bud Light-Budweiser comparison into perspective, the next comparison is Budweiser and Guiness. Budweiser is a pilsner and Guiness is a stout. Those two beer styles are clearly very different. When these two are compared, the cosine distance is around 0.92!

Since recommendations are based on similarity, the next logical step was to create a function that looks at a users ratings and uses those to recommend new beers to try. To best illustrate the recommendations, a user was randomly chosen and here are the beers they had rated:

	rating	beer_name	brewery_name	beer_style	beer_substyle	abv	user_id
787131	4.75	Fou' Foune	Brasserie Cantillon	Wild/Sour Beers	Belgian Fruit Lambic	5.50	10995
2180183	4.75	Heady Topper	The Alchemist	India Pale Ales	New England IPA	8.00	10995
2190470	4.50	Julius	Tree House Brewing Company	India Pale Ales	New England IPA	6.80	10995
1602982	4.25	Enjoy By IPA	Stone Brewing	India Pale Ales	American Imperial IPA	9.40	10995
3857968	4.25	Breakfast Stout	Founders Brewing Company	Stouts	English Oatmeal Stout	8.30	10995
233200	4.00	La Fin Du Monde	Unibroue	Strong Ales	Belgian Tripel	9.00	10995
1612429	4.00	Lagunitas Sucks	Lagunitas Brewing Company	India Pale Ales	American Imperial IPA	7.85	10995
1818110	4.00	Sculpin	Ballast Point Brewing Company	India Pale Ales	American IPA	7.00	10995

This particular user clearly has a preference for India Pale Ale, but also has a few other beer styles mixed for a bit of variety. The next table shows the recommendations for this particular user:

	beer_name	brewery_name	beer_style	beer_substyle	abv
0	Consecrator	Bell's Brewery - Eccentric Café & General Store	Bocks	German Doppelbock	8.0
1	Melcher Street	Trillium Brewing Company	India Pale Ales	New England IPA	7.2
2	Zoe	Maine Beer Company	Pale Ales	American Amber / Red Ale	7.2
3	3 MONTS (Originale)	Brasserie 3 MONTS	Pale Ales	French Bière de Garde	8.5
4	Trial Of Dmitri	Benchtop Brewing Company	Specialty Beers	Russian Kvass	4.5
5	La Clef Des Champs	Brasserie Dieu du Ciel!	Specialty Beers	Scottish Gruit / Ancient Herbed Ale	5.0
6	Double Or Nothing	Otter Creek Brewing	Strong Ales	American Barleywine	10.5
7	Three Philosophers	Brewery Ommegang	Strong Ales	Belgian Quadrupel (Quad)	9.7
8	Gouden Carolus Classic	Brouwerij Het Anker	Strong Ales	Belgian Strong Dark Ale	8.5
9	Blanche De Chambly	Unibroue	Wheat Beers	Belgian Witbier	5.0

The recommendations, when compared to the beers that the user has rated, are interesting. Given the preference for India Pale Ale, one could have expected India Pale Ales to be more frequent in the recommendations. While they are still in the list of recommended beers, they do not dominate the list. The most interesting insight is that the user rated 1 Strong Ale but the recommendations suggest 3 different Strong Ales. This was a bit puzzling at first glance but made more sense when analyzing the ABVs of the users rated beers which tend to be, well, strong! In short, the recommended beers based on the user's history seem to be a good fit.

Given how widely used recommendation systems are, it would be difficult to argue that they are not useful. In the case of making recommendations for beers, it is reasonable to believe that there would be multiple benefits to implementing such a system. Benefits range from increased sales, using the data to learn more about customer behavior, and treating the data as a product in order to monetize it for the use of other groups in the food and beverage industry. SVD was the best performing algorithm, harnessing the power of latent features to output recommendations.

The analysis of the recommended beers results in two important insights. First, the recommender engine works and the similarity between the user's rated beers and suggested beers is apparent. Second, the algorithm detected more similarities that influenced the suggestions, particularly where the high ABV of rated beers resulted in suggesting Strong Ales.

Regarding the client, an online alcohol sales company, the recommendation engine will be a great tool in driving sales, growth, and increasing the quantity and quality of customer interactions. In addition, there is a large amount of data that was scraped but not used for the recommendation engine. That data can be utilized to gain valuable insights to improve different facets of the business and its strategy, with a particularly intriguing area of investigation being the creation of another recommendation engine. This new engine could be geared towards recommending breweries and could also take in the user's current location, suggesting breweries that both compliment the user's rating history and are close enough to visit. The power of recommendation engines is evident and their impact on business is likely to continue playing a role in the future.