Restaurant Recommender

Final Report

Viraj P Modak

April 12 2020

EXECUTIVE SUMMARY

The purpose of this project is to develop a restaurant recommender system. The final product will be of interest to general internet or food services based apps and websites such as Google, Yelp, Grubhub, Doordash etc. The raw data (collected for multiple cities over the past eight years), was freely available on the web. Due to memory issues, the system was developed for the city of Phoenix, which is also logically correct since users will prefer the recommendations to be limited to a certain city. However, the model can easily be extended to any other city for which similar data is available. Exploratory data analysis and statistical tests were conducted to identify key trends in reviews and restaurant/user characteristics. However, the results from EDA add only academic value and are not critical to the final product. Two approaches were tested to build the recommender system: first which used a hybrid approach of unsupervised learning and content based filtering, and the second one which consisted of collaborative filtering. Both systems were then adapted into the final product which took in a set of user preferences and built a pool of "similar" restaurants. Recommendations can then made from this pool until the user is satisfied. Some key technical and business insights are also discussed.

CONTENTS

1. **Problem Statement..... 4**
2. **Data to be used..... 4**
3. **Problem Solving Approach..... 5**
4. **Data wrangling/clean-up..... 6**
5. **Exploratory Data Analysis..... 7**
6. **Statistical Analysis..... 16**
7. **Building Recommender System..... 19**
8. **Final Product and Discussion..... 25**
9. **Summary..... 28**

Appendix (links to relevant code/data files)..... 29

1. Problem statement:

Recommender systems have become extremely popular in today's world in social media and ecommerce industries. Some examples include Facebook recommending friends, Netflix recommending movies or Amazon recommending products. A similar approach can be used to match restaurants to consumers. Typically when users look for restaurants, they go by star ratings alone. This approach is fraught with multiple shortcomings. The reviews can be skewed toward users with different preferences than the user who is looking for restaurant options. Furthermore, given the sheer number of restaurants in a large city, it is possible some really good options may never receive any consideration. The proposed recommender system will aim to tackle these issues. The problem statement can be defined as follows: Design a recommender system based on existing star ratings, reviews and user profiles, which will recommend a set of restaurants to a user. It is assumed here that the user will not have visited these restaurants before. ***Owing to memory issues which were encountered while running the code, the product was limited to the city of Phoenix - however, it can easily be altered for any other city of choice.***

The obvious target client in this case is the entire population of a big city as they will be the direct users of this product. It can also serve as an advertisement platform for new restaurants. In addition, it can also be marketed to restaurant review apps or app based companies such as Yelp and Google.

2. Data to be used:

For this purpose, the Yelp restaurant review database will be used. The raw data is in the form of multiple json files including user, business and review information. The dataset can be found at the following location:

https://www.kaggle.com/yelp-dataset/yelp-dataset

This data will need to be cleaned up and represented in a matrix form where the rows are users and the columns are restaurants.

3. Problem solving approach:

The problem was tackled to two distinct ways - essentially we will have two final products. Both of them are summarized as follows:

1. Hybrid system (content-based filtering + K-Means clustering):
    - Content-based filtering will recommend restaurants based on similarities between restaurants; the similarities will be calculated using vectorized review text
    - The vectorized review texts can also be subjected to a K-Means clustering algorithm to find centroids or "clusters" of similar restaurants
    - The final recommendation set is essentially an overlap between the two recommendation sets
2. Collaborative filtering (user-based + item-based)
    - The database will be reduced to a rating matrix where the rows depict users and the columns depict restaurants and the cells will be the corresponding star ratings
    - This matrix will then be subjected to a standard collaborative filtering algorithm. More details on how this algorithm works, are provided in section 7.

4. Data Wrangling/Clean-up

Since this is not a traditional X vs Y machine learning problem, data wrangling was not a one-time effort. In particular, there were subsequent clean-up steps involved which are described as part of Exploratory Data Analysis. Following is a brief summary of the preliminary data wrangling process.

- The data consisted of separate json files for user info, review info and business info. The json files were converted into Pandas dataframes.
- The file for business info included information about other services as well. This was filtered to include only Restaurants
- The dataframes were then saved as csv files and these csv files will then be used for all subsequent analysis
- The info files consisted of data from multiple metropolitan cities. Based on the problem statement, it made sense to develop a system for a city and the same methodology can be implemented for other cities as well
- The rating matrix with users as rows and restaurants as columns with the star ratings as cells was extracted which will be the starting point for training the recommender system based on collaborative/content-based/hybrid filtering.
- However, the data in the info (csv) files will be used to perform Exploratory Data Analysis and Inferential Statistical Analysis

5. Exploratory Data Analysis

The EDA approach is described in a fully executed notebook and the link for the same is provided in the Appendix. Key highlights and plots are shown here.

5.1 Distribution of ratings - raw data

The star ratings from the review_info dataset were plotted to explore how the ratings are distributed. It is shown in Figure 1. It can be seen that:

- 5-star reviews are more than 1-4 star reviews
- More than half of the reviews imply a positive experience (4-5) stars

This shows that in most of the cases, the reviewers have been satisfied because of the service. However, this might mean that only the "good" restaurants get reviewed often and may not mean the general quality of the restaurants in the city of choice is good. For that how the restaurants distributed across the star-rating metric needs to be explored.

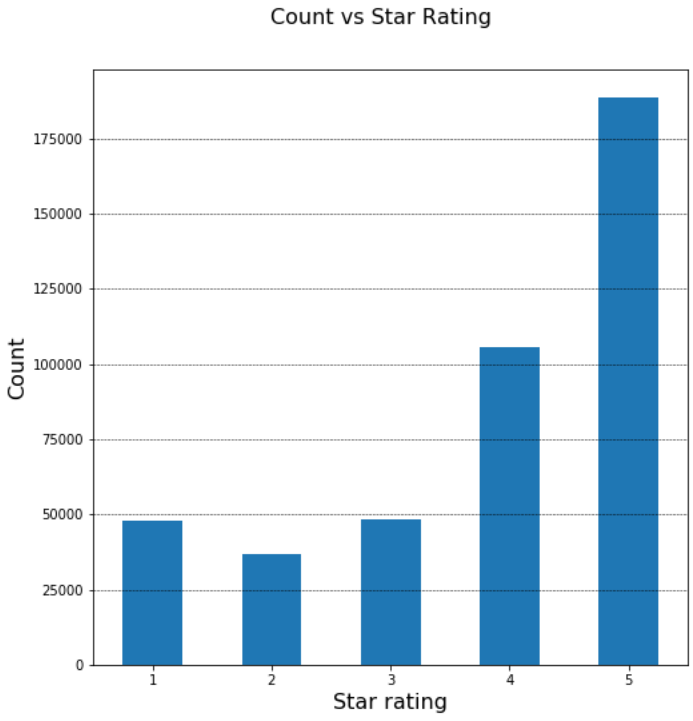

Figure 1: Count of reviews plotted as a function of the review star rating

5.2 Distribution of restaurants by star reviews

The restaurant count can be plotted as a function of the average star rating for those restaurants - shown in Figure 2. It sheds more light on what was seen in the previous plot. Overall in the city of Phoenix, there are a little over 500-600 restaurants which have an average rating of above 4. However, the majority of the restaurants are rated 3-4. This in a way confirms what was hypothesized earlier, that only "good" restaurants are reviewed more often. This can mean either people don't visit the not-so-good restaurants as often or that after people visit these restaurants, they do not feel like writing a review. Overall though, it can be concluded that in Phoenix, people are satisfied with their restaurant experience - (assumption: above 3 star is satisfied).

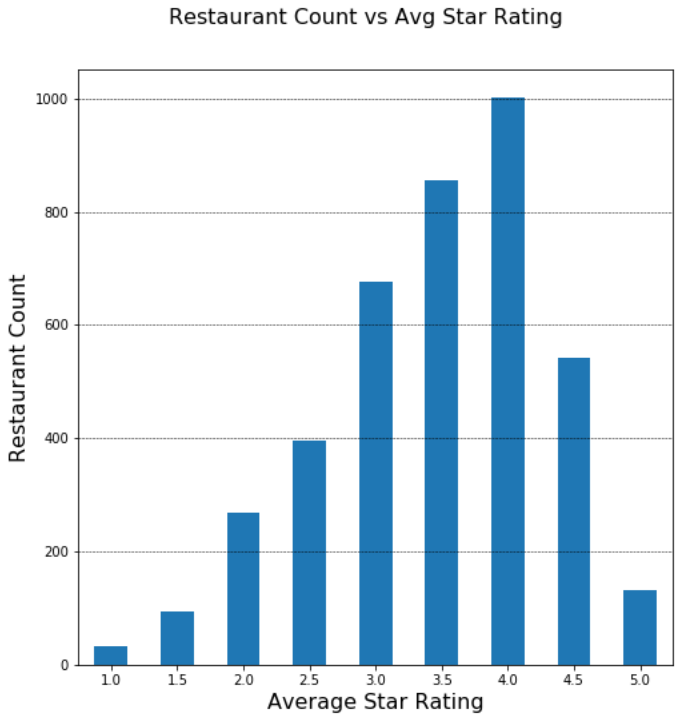

Figure 2: Restaurant count as a function of average star rating

5.3 Ratings per user

This analysis will help identify how active a user is while reviewing and shown in Figure 3. This makes for an interesting viewing. It is interpreted as follows:

Almost 100000 users have written only ONE review for restaurants in Phoenix. On the other end there are only 20 reviewers who have written more than 100 reviews for restaurants in Phoenix.

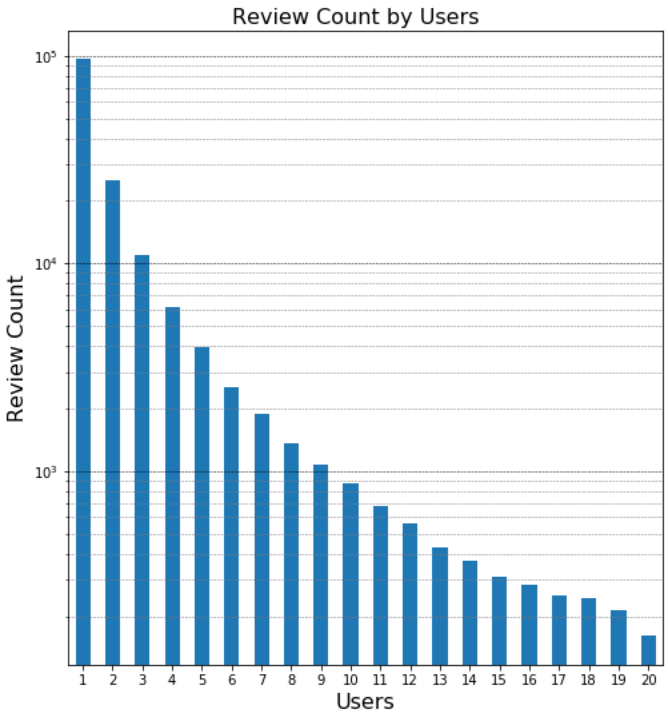

Figure 3: Review count per bin of user count

This analysis will be useful in the future because collaborative filtering is based on user characteristics and reliable characteristics about likes and dislikes can be extracted only if a user frequently writes reviews. Getting information from people who have reviewed only once or twice does not make much sense

5.4 Ratings per year and per month

Time trends in review writing habits can also be explored to answer questions like is there a particular year(s) or period of the year when the reviews are more frequent? The frequency of reviews by month and by year are shown in Figure 4 and Figure 5 respectively. We can see that the number of reviews submitted has definitely increased over time. This can of course be explained by an increases in connectivity, reliance on

online reviews and their prominence. As far as review month is considered, we do not see a large variation between months, but indeed, during the Nov/Dec holiday season, the frequency of reviews decreases, possibly because people prefer to stay indoors with family.

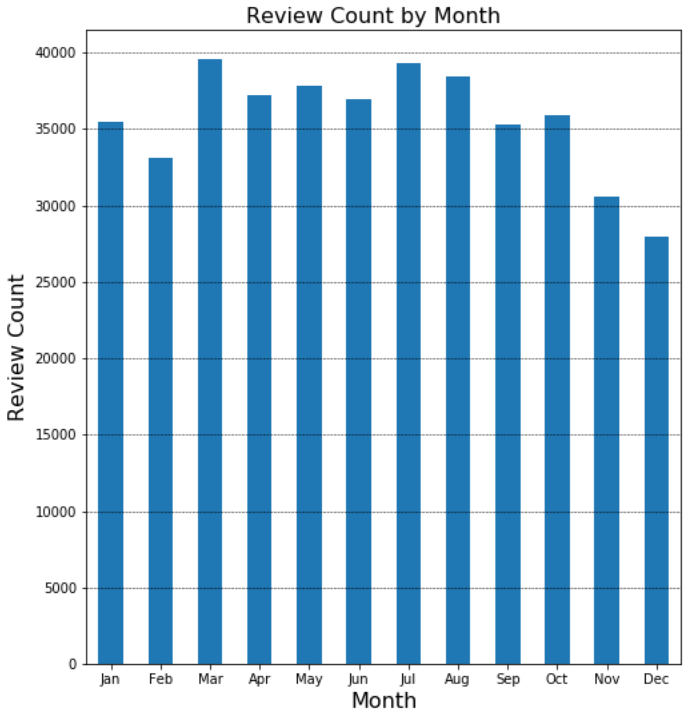

Figure 4: Review count by month

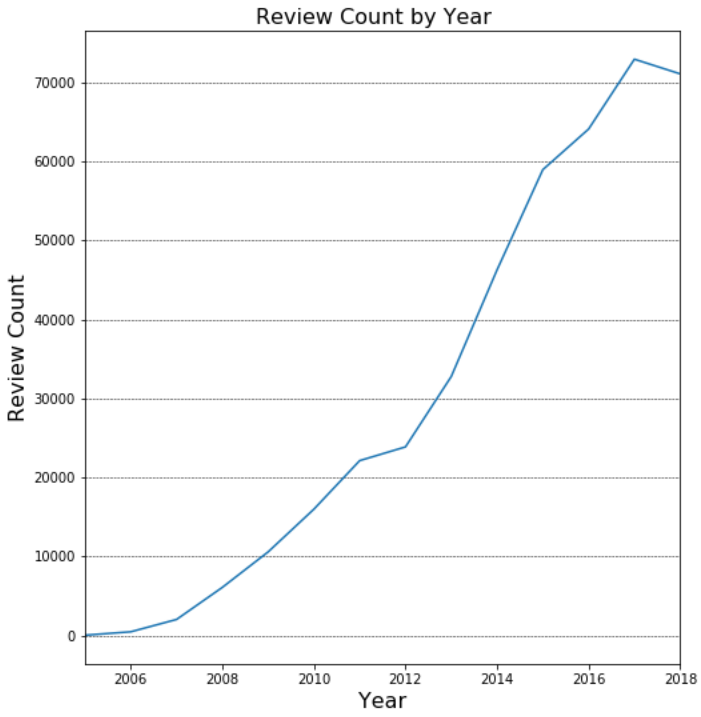

Figure 5: Review count by year

5.5 Restaurant count by category

Content based filtering is a common way to build a recommendation system. In this case, multiple objects (or in this case, restaurants), will be compared based on their characteristics - or content. In our case, one such characteristic is "category". In the next few plots, the trends between category and reviews are explored for the city of Phoenix. Figure 6 shows how many restaurants serve some typical cuisine. Mexican has the highest with >700 restaurants with Indian being the lowest with only ~50 restaurants. The data is not scrubbed for being exclusive. For example, there might be some restaurants which serve both Mexican and Italian. However, the y-axis will still reflect the correct number of restaurants serving that particular cuisine.

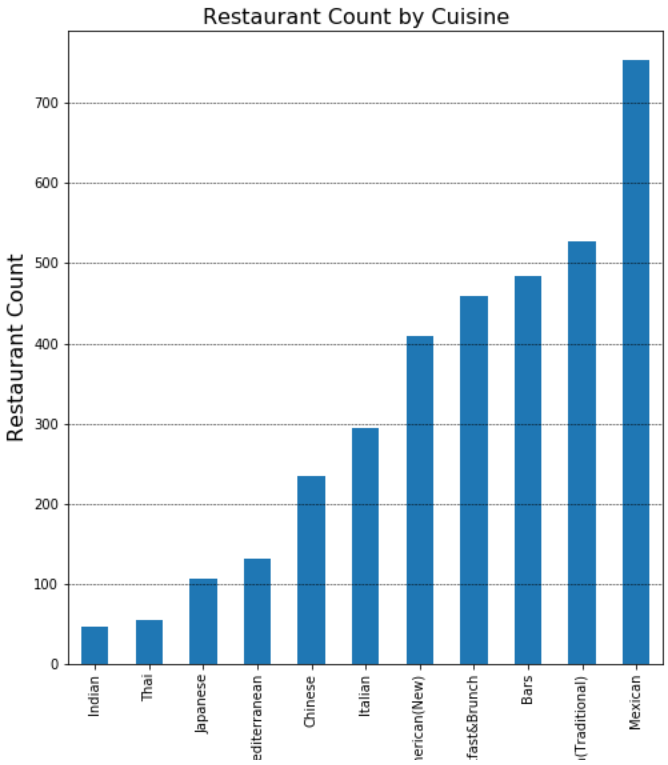

Figure 6: Restaurant count by cuisine

5.6 Restaurant count by category - segmented by star rating

We can extract how the restaurants are rated based on the category (or cuisine) shown in Figure 7. The resulting graph has a lot of information which needs to be unpacked. The general trend we see is expected i.e. a bell shaped plot of ratings - 1-5 stars. With peaks around either 3.5 or 4. We can extract an average rating by category as well, which will be seen in the next plot

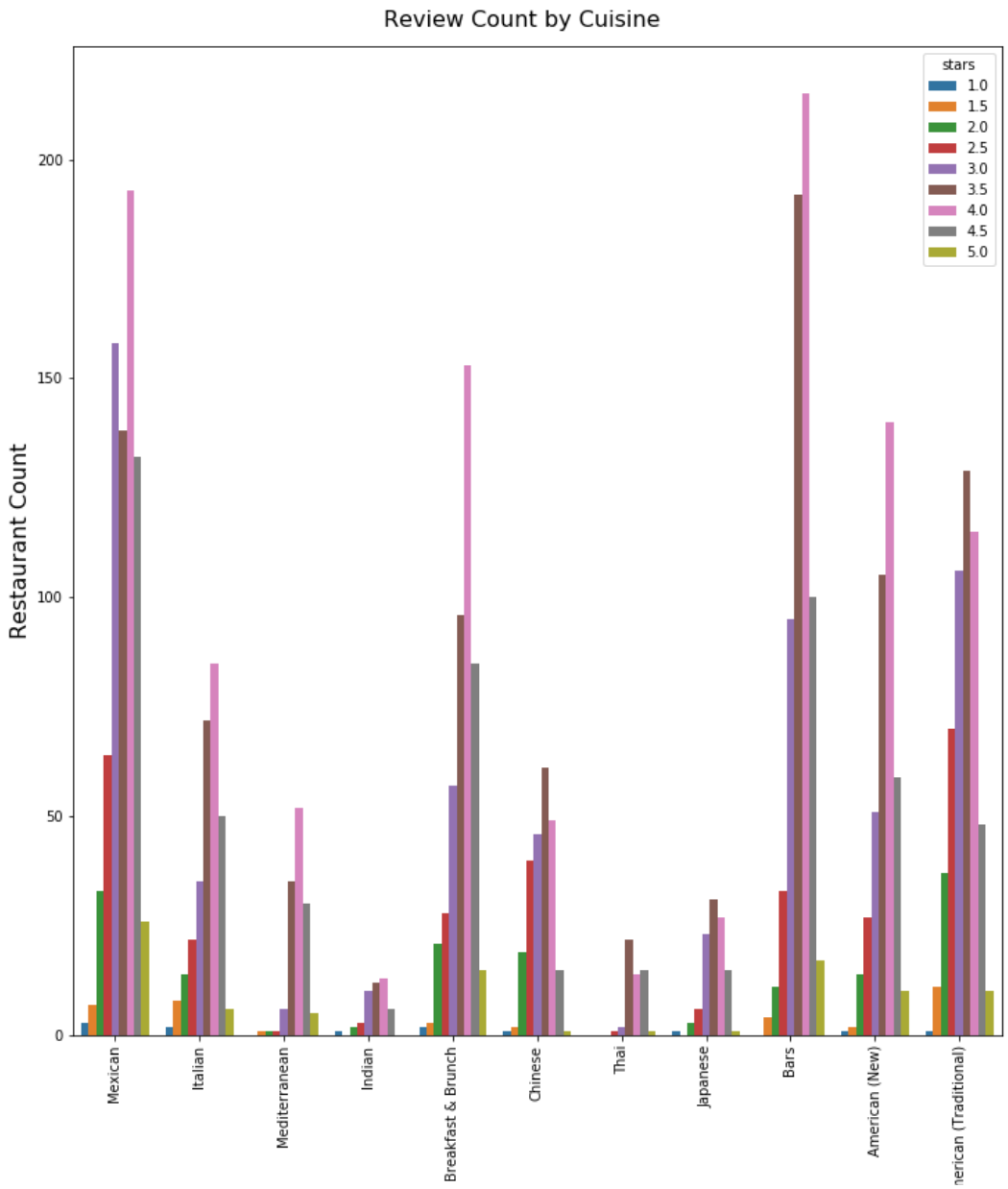

Figure 7: Restaurant count by category/segmented by star rating

5.7 Average Rating by category

As noted before, most of the categories (or cuisine) average between 3.5 and 4 stars as shown in Figure 8. An interesting thing to note here is with the Mediterranean and Thai restaurants - we saw that these two cuisines offer the least choice in terms of number of restaurants. Even so, they are two of the best rated cuisines in the city of Phoenix.

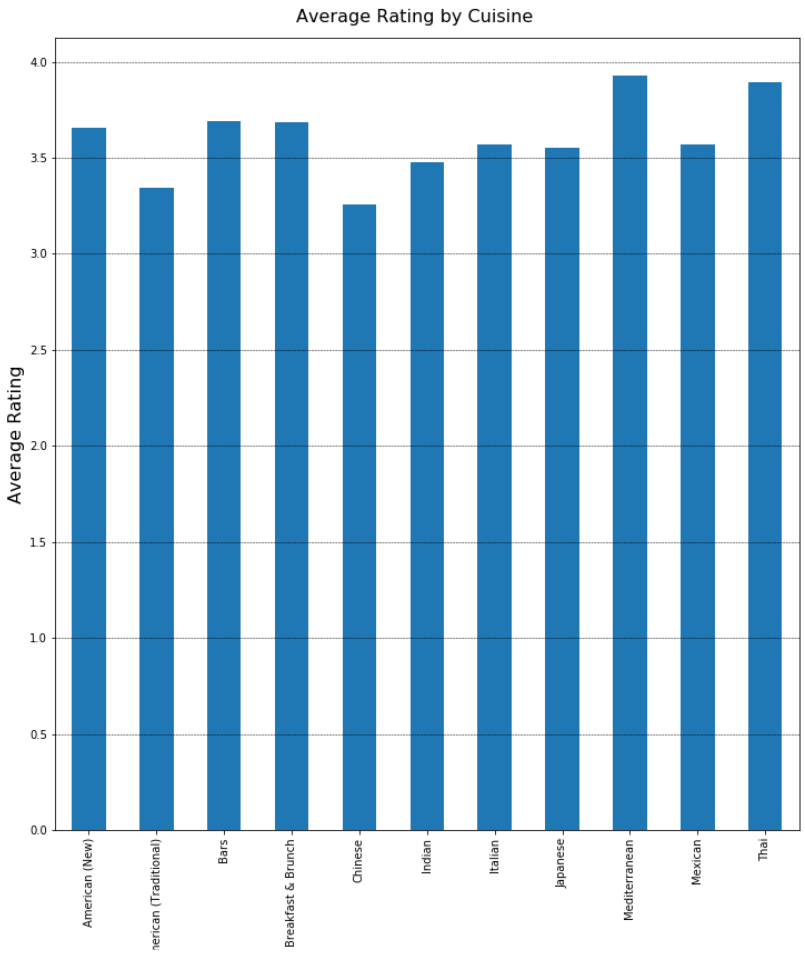

Figure 8: Average rating by Category

5.8 Review count vs useful

In the previous plots, it was explored how the number of ratings vary as a function of number of users. But how many of them are actually helpful and if there are any outliers is yet to be seen. The "useful" count vs review count can be plotted to explore that relationship - shown in Figure 9. Intuitively we expect this to be a positive correlation which is what we see.

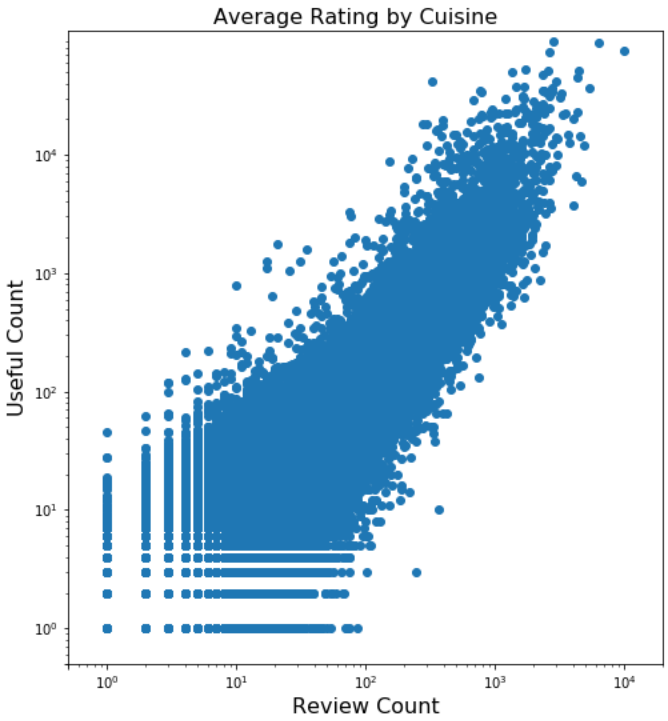

Figure 9: Usefulness of user reviews vs review count

6. Statistical Analysis

As mentioned before, this is not a standard machine learning problem with a set of independent variables and a response variable. As a result, standard inferential statistics and statistical tests may neither be relevant nor add any value to the end goal. Even so, based on the variables we have explored, we can still obtain basic statistical parameters such as mean and standard deviations wherever applicable. A few relevant cases are illustrated below.

6.1 Rating mean and stdev by cuisine

This data was shown in Figure 8 but listed along with the standard deviations in Table 1. We see no significant differences in the spread of the ratings. All of them lie between 0.5 and 0.8. Italian, is the only noticeable cuisine which has a larger spread considering the number of Italian restaurants are not as high as some of the other cuisine.

Table 1: Mean and stdev of restaurant ratings by cuisine

Cuisine	Mean Rating	Stdev
American (New)	3.656479	0.68597
American (Traditional)	3.343454	0.7755
Bars	3.691904	0.63072
Breakfast & Brunch	3.684783	0.74579
Chinese	3.25641	0.72985
Indian	3.478723	0.74424
Italian	3.568027	0.80557
Japanese	3.551402	0.67619
Mediterranean	3.927481	0.54815
Mexican	3.570955	0.78113
Thai	3.890909	0.50636

Furthermore, one-way ANOVA can be also performed on star ratings by cuisine to see if the mean star rating for different cuisines are statistically dissimilar. For this purpose the star ratings of the individual reviews from the review_info file were segment it by cuisine and the mean of those star ratings were subjected to a one-way ANOVA. This is a more robust check than the one before, because the data size is larger and TRUE mean of cuisines will be tested rather than mean of average star rating of the restaurants serving that cuisine. The F-value for this test was 323.8 and the P-value was 0, suggesting that

the null hypothesis - that all means are equal - is rejected. Details of this calculation are presented in the IPython notebook for EDA.

6.2 Quantifying review length by keywords

In the data-wrangling and EDA sections, the review text was vectorized into keywords. It will be interesting to see how long a typical review is. The following statistics were calculated for length of reviews:

- **Mean**: 51 keywords
- **Median**: 36 keywords
- **Stdev**: 47 keywords

It can be seen that there is a large spread - mean:stdev is almost 1:1 - in terms of reviews written. In other words, there are users which are succinct in their reviewing and there are users which are more verbose. However, the distribution is heavily skewed toward to reviews which are shorter, which is explained by the median value of 36. The distribution is plotted in Figure 11.

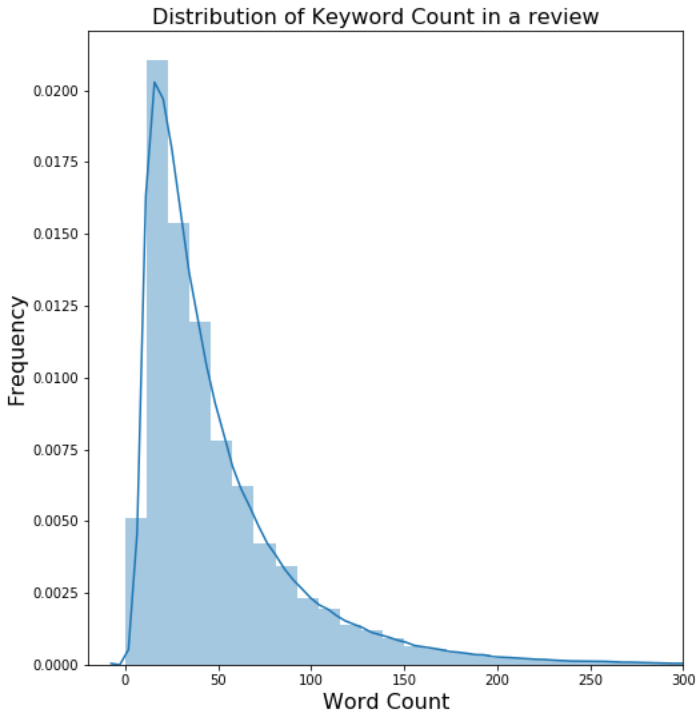

Figure 10: Distribution of review keyword count

6.3 Yelping age Statistics

In collaborative filtering, the recommendations are based on how the user is similar to previous users or reviewers. One characteristics which is worth exploring is the "Yelping Age" or how long has the user been active on Yelp. The following statistics were calculated and the distribution is shown in Figure 12. We can see that the average yelp user in Phoenix is quite mature at an age of 6-7 years

- **Mean**: 2404 days
- **Stdev**: 945 days
- **Median**: 2372 days

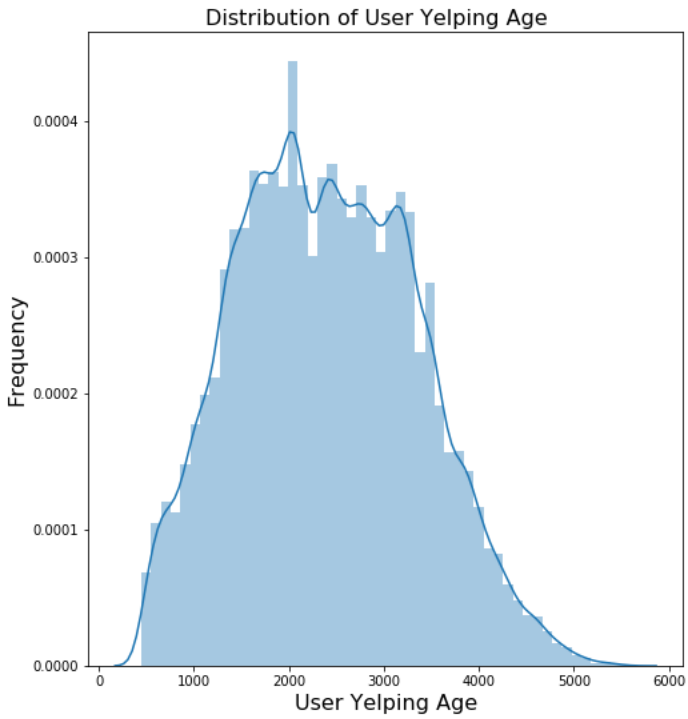

Figure 11: Distribution of Yelping Age of Users

7. Building the Recommender Systems

Now that the initial leg-work of cleaning up the data and visually analyzing the data, has been done, we can move on to actually building the recommender system model. In Section 3, two distinct strategies were proposed to solving this problem.

1. Content-based filtering coupled with a K-means clustering algorithm

2. Collaborative filtering using user-based and item-based similarities

7.1 Content-based filtering with K-means clustering

We start with a dataset which has information about the user, business and the review text. The review text in this case is the "content" on which our recommender system will be built. All the reviews for each restaurant are collated into a single text string, which is then vectorized with TfidfVectorizer. This vectorized form will then be used for two purposes. Firstly, to form clusters and second, to obtain similarities between restaurants.

The review vectors are scaled using a StandardScaler and features are extracted from the scaled vectors. Using PCA and variance analysis, the number of features can then be reduced to a value of choice. Usually this value corresponds to a maximum drop in variance in a variance vs feature plot (Figure 13), which in this case would be a total of 2-3 features. However, given the number of data points which we have, this would result in highly overlapped clusters. Hence we would like to have a significantly higher dimensionality. The number of PCA components (arbitrarily) was chosen to be 41. Because this problem is not one with a single right answer - the effect of this number on the efficacy of the final solution was not explored.

Once the principal components were available, the data was then subjected to K-means clustering. The plot of inertia vs number of clusters shows that the inertial drops significantly between 3 and 4 clusters. This suggests that our dataset can be resolved to ideally 3 or 4 distinct clusters (Figure 14). However, once again, because we needed higher complexity to our model, we chose a cluster count of 20. Post K-means clustering, each restaurant belonged to one of the 20 clusters or "centroids". Subsequently, given any restaurant of choice, its centroid-neighbors can be identified using a simple equality

based filter. The code for this approach is presented in a Jupyter notebook, the link to which is included in the Appendix. This model could be an independent recommender system in theory, but to make it more robust, can be coupled with a content-based filtering algorithm.

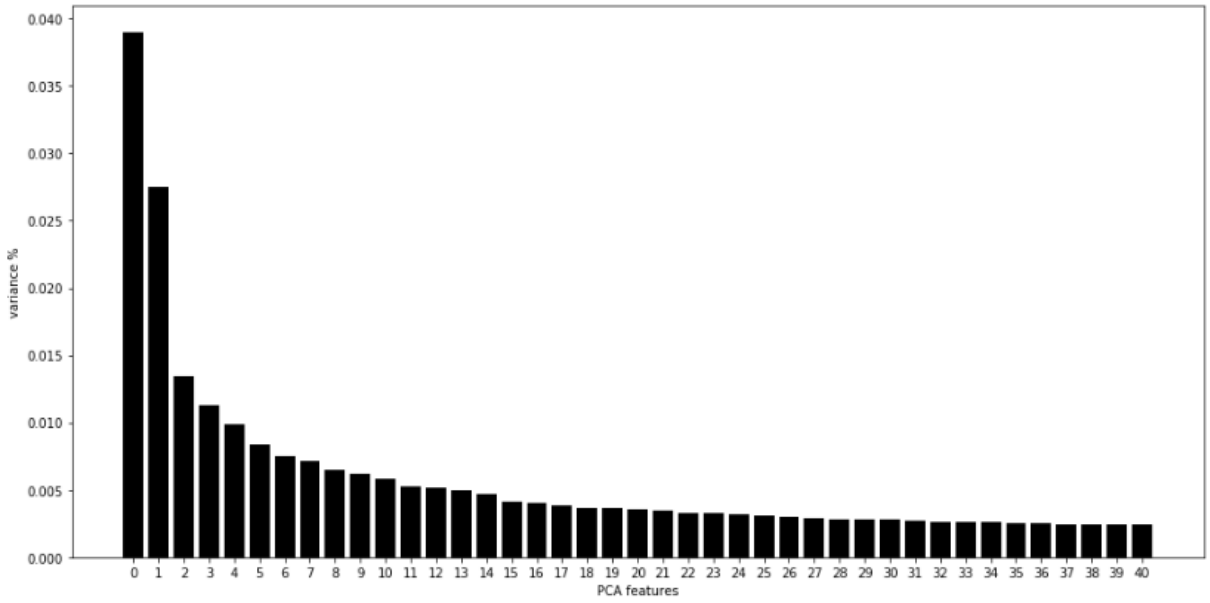

Figure 12: Variance% as a function of PCA Features

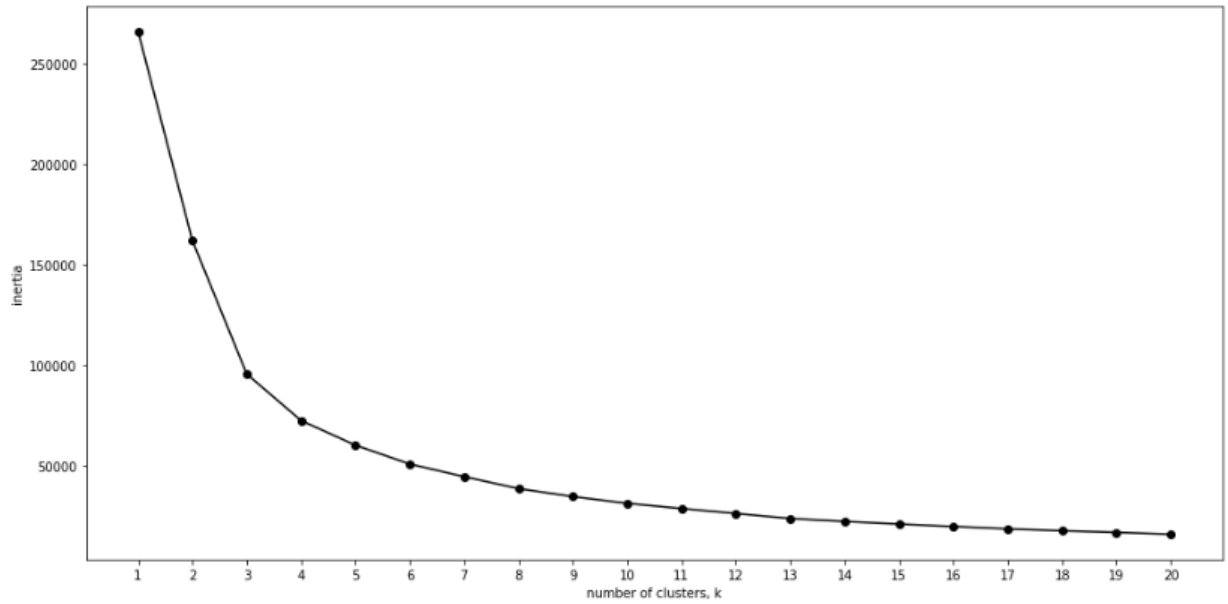

Figure 13: Inertia as a function of n_clusters

The starting point for content-based filtering is the same vectorized review text which was used to generate the K-Means clustered system. The vectorized review text essentially is represented in the form of a Pandas series where the index is the restaurant-id and the value is the actual collated review text. Pair-wise cosine similarities are then calculated using a "linear-kernel function" which result in a "symmetric matrix" where all the diagonal elements are unity. Each row essentially represents the similarity value of restaurant-id of the index with each of the other restaurants, including itself. Most similar restaurants for each index can then be extracted by a simple value-based ordering or a filter of the entire row. This approach would be implemented in case this model were to be used as a stand-alone one. However, since we are coupling this with the K-Means clustering algorithm, we can extract the similarity values of the centroid neighbors. The final recommendations are the ones which belong to the same centroid ***and*** which are most "similar" to the restaurant.

7.2 Collaborative filtering

Collaborative filtering essentially uses similarity between users to make recommendations. Furthermore, there are two approaches User-based and Item-based and the approaches can be summarized using the following statements:

Item-based Collaborative Filtering: “Users who liked this item also liked …”

User-based Collaborative Filtering: “Users who are similar to you also liked …**"**

We start by splitting the data into a training and a test set. Following that, the first step in creating this model is to resolve the review dataframe into a rating matrix, where the row is the user-id and the column is the restaurant-id. The rating matrix first mentioned in Section 4, can be extracted using a simple Pandas pivot table. Once the rating matrix is available, the user and item similarities can be extracted using the standard cosine similarity formula. The corresponding item similarity formula is not shown for redundancy.

$$sim(u,u') = \sum \frac{r_{u,i} r_{u',i}}{\sqrt{\sum r_{u,i}^{\ 2}} \sqrt{\sum r_{u',i}^{\ 2}}}$$

Once the similarity is calculated, we can move ahead and predict the ratings which are missing. For user-based collaborative filtering, we predict that a user u's rating for item i

is given by the weighted sum of all other users' ratings for item i where the weighting is the cosine similarity between the each user and the input user.

$$\hat{r}_{u,i} = \sum_{u'} sim(u,u')r_{u',i}$$

This user prediction along with the test matrix, can be used to extract mean squared error. This algorithm can be enhanced by considering only fraction of similar users which are most similar to the users to make the predictions. However, this did not result in any significant improvement in the predictions. It is then straightforward to validate your recommendations:

7.3 Validation with examples

The final product will accept star ratings and reviews on a few restaurants from the user, before making recommendations using the two models which we have designed. However, before that we need a way to get some idea about the performance. This can be done by getting recommendations for each possible test subject for both models. For K-Means clustering with content-based filtering described in Section7.1, it is straightforward, where recommendations belong to the same centroid and which have a high similarity with the subject restaurant.

Table 2: Recommendations from Content-based Filtering

Restaurant	Recommendation1	Recommendation2	Recommendation3	Recommendation4	Recommendation5
Taco Bell	Seed Cafe	Stacy's Off Da Hook BBQ and Soul Food	Szechwan Palace	Morton's The Steakhouse	Frank's New York Style Deli
Nee House Chinese Restaurant	Huss Brewing Co - Uptown Taproom	Carl's Jr 7081	Blue Mesa Tacos	Santanas Mexican Food	Latino Restaurant
Vals Getaway DES Cafeteria	Little Shanghai	Jimmy John's	Mariscos El Dorado Sin	Little Caesars Pizza	Europa Pastry Cafe
Sushi Mocorito	Southern Rail	Cyprus Grill of Phoenix	Livin' Lite Arizona	Macayo's Mexican Table	SpoonZ Cafe
Oregano's Pizza Bistro	Waffle House	Havana Cafe	Corbin Bar & Grill	Fuzzy's Taco Shop	Sushi-to
Santanas Mexican Food	Ramiros Mexican Food	Huss Brewing Co - Uptown Taproom	Sun Asian Kitchen	Chipotle Mexican Grill	Subway
Five Guys	Uberrito	Burger King	Cork'n Cleaver	KOVO Modern Mediterranean	My Big Fat Greek Restaurant
Pizza Hut	Domino's Pizza	Pete's Fish & Chips	Guanaquito Restaurant	Pastries N Chaat	Outback Steakhouse
Mariscos El Dorado Sin	Pizza Patron	Buddyz a Chicago Pizzeria	Samurai Sam's	Papa Murphy's	Boomer's Sweet Home Chicago
12 East Cafe	Barrio Cantina & Grill	COFCO Chinese Culture Center	Thao Sandwiches	The Notorious PIG	Rimrock Bar and Grille

Some examples obtained using the content-based filtering algorithm are shown in Table 2 and the ones obtained using the hybrid approach are shown in Table 3. We can see that there is some redundancy in the hybrid approach.

Table 3: Recommendations obtained using hybrid K-Means clustering and Content-based filtering

Restaurant	Recommendation1	Recommendation2	Recommendation3	Recommendation4	Recommendation5
Taco Bell	SongbirdCoffee&Tea House	Chick-fil-A	PizzaHut	Dennys	BirrieriaObregon
Nee House Chinese Restaurant	SantanasMexicanFood	Subway	CarolinasMexicanFood	OlivesMediterraneanGrill	TacoBell
Vals Getaway DES Cafeteria	SamsGyros	Subway	TacoBell	SushiMocorito	PizzaHut
Sushi Mocorito	Subway	McDonalds	OlivesMediterraneanGrill	TacoBell	JackintheBox
Oregano's Pizza Bistro	Subway	MariscosElDoradoSin	TacoBell	SushiMocorito	PizzaHut
Santanas Mexican Food	TacoBell	FiveGuys	BirrieriaObregon	DaisyMountainCoffeeRoasters	MariscosElDoradoSin
Five Guys	Dennys	McDonalds	ArmadilloGrill	TacoBell	OnDeckSportsGrill
Pizza Hut	Subway	ChickRotisserie&WineBar	DaisyMountainCoffeeRoasters	BirrieriaObregon	Dennys
Mariscos El Dorado Sin	BirrieriaObregon	TacoBell	DaisyMountainCoffeeRoasters	PizzeriaBianco	Subway
12 East Cafe	TacoBell	BirrieriaObregon	MariscosElDoradoSin	12EastCafe	PizzaHut

In case of collaborative filtering, getting recommendations from item-based collaborative filtering is straightforward, where given a subject restaurant, we just need to sort other restaurants by similarity to the subject restaurant. Some examples are shown in Table 4. Some of them make sense like Taco Bell being similar to Filberto's Mexican and Raising Cane's and Sushi Morito being similar to Wild Tuna Sushi. Whereas some others are not exact matches such as Oregano Pizzeria being similar to a Yogurt shop. However, we cannot be dismissive of this similarity because they might be linked in some other way such as proximity, popularity among a certain age group, working hours and so on.

Table 4: Recommendations from User-based based collaborative filtering

Restaurant	Recommendation1	Recommendation2	Recommendation3	Recommendation4	Recommendation5
Taco Bell	Tropical Smoothie Cafe	Filiberto's Mexican Food	Boston Market	Raising Cane's	Bear And The Honey Specialty Bakery
Nee House Chinese Restaurant	Kwan & Wok Chinese Restaurant	SOLO Trattoria	McDonald's	Good Taste House	Generation Y Design
Vals Getaway DES Cafeteria	Spinato's Pizza	Original ChopShop	Burger King	Alida Restaurant Supply	Panda Express
Sushi Mocorito	Clarendon Kitchen + Bar C4	Wild Tuna Sushi and Spirits	Dickey's Barbecue Pit	Plaza Bonita	Wingstop

Oregano's Pizza Bistro	Yogurt Mart	Draw 10 Bar & Grill	Chelsea's Kitchen	The Ol Pizzeria & Cafe	Tariq Restaurant
Santanas Mexican Food	Palms Restaurant and Market	Giant Manhattan Pizza & Pasta	Oscar's Pizza	Firehouse Subs	Fatburger
Five Guys	Del Taco	Moon Valley Grill	O Bar & Grill	Tacos Huicho	Chick-fil-A
Pizza Hut	R.Kidd's Pizza & Wings	Baja Loco Mesquite Grill & Cantina	Jack-In-the Box Drive Thru	ThirdSpace	Panera Bread
Mariscos El Dorado Sin	Fattoush Restaurant	Taylor's Place	Casita Del Mar	Chick-fil-A	Kona Grill

User-based collaborative filtering involves recommendations based on user similarities. Hence it involves identifying users which are most similar to the subject user and then choosing restaurants which are rated highly by those users. In this case though, there might be instances where there will be no recommendations based on what filters are set - for example let's consider a user who is a sparse reviewer. He will be similar to other "sparse" reviewers. In case this set of reviewers have not given "good" reviews, then it will be hard to find good recommendations based on a user-based collaborative filtering algorithm. Examples for these are shown in Table 5.

Table 5: Recommendations from User-based based collaborative filtering

User_index	Recommendations
0	'Viet Kitchen' 'Asian Cafe' 'Las Jicaras Mexican Grill' 'Bobby Q' 'The Greek Pita' 'Ocotillo' 'El Bravo Mexican Food' 'Mi Pueblo'
1	'Naked BBQ' 'Rustic Cafe' 'Paradise Valley Pizza Company' "Carolina's Mexican Food" 'Barrio CafÃ©' 'HEK Yeah BBQ' 'Los Betos Mexican Food'
2	Ollie Vaughn's
3	'La Grande Orange Pizzeria' 'Bobby Q'
4	'Hooters'
5	
6	'The Henry' 'Jobot Coffee & Diner' 'Olive & Ivy'
7	'The Vig Uptown'
8	'Doughbird' 'Great Wall Cuisine' 'Pappadeaux Seafood Kitchen' 'Jamba Juice' 'Da Vang Restaurant' 'Pizza Heaven Bistro'
9	'Adobo Dragon' 'Hana Japanese Eatery'

It can be seen that developing Recommender Systems is a challenging problem with no right or wrong answer. However, some insight into what parameters can impact the results will be added as a separate Discussion section in the final report.

8. Discussion

In this section, some key insights are presented concerning the technical as well as the business aspects of this product. Although a lot of these ideas are beyond the scope of this project, they are critical in visualizing how the system can be prodcutionizede. For ease of understanding, it is divided into three sub-sections viz. Optimization, Deployment and Success Criteria.

8.1 Optimization:

As seen before, based on user's preferences, the system will build a pool of recommendations from which values will be extracted five at a time, until the user chooses to stop. This approach is taken for both systems. As with any recommendation system, it is hard to evaluate its performance since there will not be a single right answer. More thoughts on this are presented in section 8.2.

In this section though, let's discuss some trade-offs in the system as well as some parameters which can be tuned to optimize the performance. For example, in the hybrid recommendation system, review text is used to calculate the similarities. On one hand it may seem that this metric may not be the most correct way to evaluate similarities - since it can potentially view a Mexican and an Italian restaurant as similar if their reviews are largely similar. However, in a collated dataset of all the reviews, this method will be adept at picking up subtle similarities in non-cuisine related characteristics; for example service, location, view, hygiene and so on. Other factors which can affect the behavior and can be tuned to change performance are listed as follows:

a. **Vectorizer** - tfidf vectorizer was used. Count vectorizer is another common option
b. **Similarity calculation** - a linear kernel was used to calculate cosine similarities. Other common similarities include Euclidian and Jaccard similarities
c. **Number of PCA features:** 40 was chosen as an arbitrary number
d. **Number of centroids:** 20 was chosen as an arbitrary number

Moving on to the collaborative filtering model, the current product takes in only the names of the restaurants as "user preferences" and assumes a 4-star rating. However, the final product can of course be made more sophisticated by also taking in the corresponding

star ratings. Even in this case, as with the hybrid approach, one of the factors which affect the performance is the similarity formula. A simple dot product (cosine) similarity was used but other strategies, including Jaccard, Pearson and Euclidian similarities are options. Finally, a simple mean squared error value was used to test the accuracy of the predicted star ratings. A more sophisticated approach would be as follows. Use vectorized review text features to get resolved user-features matrix (U) and business-features matrix (B). Calculate the error as the differences between the "rating matrix" and the matrix multiplication of U and B. Minimize the error using gradient descent to optimize U and B. Such an approach was tried but proved computationally intensive. The cost can be reduced by truncating data but this was relegated in favor of using a simpler calculation of user/item based similarities.

8.2 Deployment:

As with any review-based platform, the amount of data available is constantly increasing as more and more users review the subjects (restaurants in this case). Any new data can of course be used to improve the performance of our model. This leads to the following questions. How frequently should the model be updated? Should the user see updated recommendations every time they access this product? Or should they see updated recommendations every time they review a new restaurant? These questions are open ended with no right answer. Furthermore, there are computational costs associated with a model rebuild as it takes a non-trivial amount of time. A reasonable approach would be update the parent model at specified time intervals - for example every day. And use this update the recommendation pool using the new model for a user ONLY when they review some new restaurants; in other words, only when they add value to the product.

8.3 Business Value

Finally any new product design will be incomplete without understanding its business value. In this case, how do understand whether this product is making good recommendations and how do we quantify the user satisfaction? As discussed in Section 8.1, there are multiple ways in which we can tune our model. But how do we decide which are the correct parameters? Taking an iterative, manual approach to answer these questions is suboptimal and would take several man hours.

However, we can "outsource" this problem to the users themselves. For example, when we deploy a model, we can add a survey question along with the recommendations - "Did you find these recommendations helpful?" The response to this question can help provide insight in whether the recommendations make sense or not. Based on this information, we can revisit the model and tweak it using the variables discussed in Section 8.1. Performance of two models can be compared using a simple A/B analysis conducted over a period of time, may be a few weeks to a few months.

There will be an unavoidable uncertainty about the effectiveness of the very first recommendation system. But over time, user satisfaction data can be collected and used to improve model performance significantly. At that point, it will add serious value to the customer as well as to user experience.

9. Summary

We started with a problem statement of developing a Restaurant Recommender app which primarily can be used by the general population, also used as an advertising platform and finally also acquired by apps such as Google and Yelp. The raw dataset was acquired from Kaggle, which was cleaned up to a usable format. This included information about reviewers, restaurants and the actual reviews. Keeping in mind that this is not a standard ML problem with independent and response variables, EDA and Statistical Analysis were performed for informational purposes.

Two models were developed to solve the problem. The first one was a hybrid approach using an overlap between K-Means clustering and content-based filtering. Vectorized review texts were used to identify centroids and calculate the content similarities for the restaurants. The second model implemented a collaborative filtering algorithm. User and item based similarities were calculated to make recommendations.

Both models provided relevant recommendations along with some seemingly unrelated recommendations. To counter this, a repetition feature was added akin to a user scrolling down where recommendations were made from a larger pool until the user reached a decision. Towards the end, some key insights are presented on how the model can be optimized and deployed, and how user satisfaction data can be used to improve model performance and enhance the business value of the product.

Appendix

Cleaned up dataset is not included because of the large size. However, instructions for cleaning up the dataset from the raw data are available in the following notebooks.

Link to preliminary data clean-up notebook:

https://github.com/virajmodak16/Restaurant_Recommender/blob/master/Preliminary_data_cleanup.ipynb

Link to Notebook for EDA and Statistical Analysis:

https://github.com/virajmodak16/Restaurant_Recommender/blob/master/Exploratory%20Data%20Analysis.ipynb

Link to the Notebook for K-Means clustering and Content-based filtering

https://github.com/virajmodak16/Restaurant_Recommender/blob/master/Hybrid(Content%2BClustering).ipynb

Link to the Notebook for collaborative filtering algorithm

https://github.com/virajmodak16/Restaurant_Recommender/blob/master/Collaborative-Filtering.ipynb