TRUE REVIEW

A Personalized Yelp Recommendation Engine

Dmitriy Kats
June 2019

Table of Contents

1. Introduction and Executive Summary

Many times, we find ourselves on Yelp, looking for a new restaurant in the area with each one having thousands of reviews. There is no way we can process that information to understand how people rate and what they value. Our recommendation system will take a given user's reviews and compare to others'. Based on content and dining style, we'll present more relevant restaurants with a better fit. A content / collaborative based recommendation engine that will also average the star ratings and predict what a given user might rate the restaurant.

2. Acquiring and Aggregating the Data

The data were obtained from the Yelp Challenge Dataset, containing:

business - Contains business data including location data, attributes, and categories.
review - Contains full review text data including the user_id that wrote the review and the business_id the review is written for.
checkins - Checkins on a business.
tip - Tips written by a user on a business. Tips are shorter than reviews and tend to convey quick suggestions.
user - User data including the user's friend mapping and all the metadata associated with the user.

All data was placed into a Google storage bucket and then loaded into BigQuery for easy access.

In order to reduce the size of the Pandas data we're working with we:

- Queried the tables only with restaurants that are not fast-food
- Joined the restaurant data with review data
- Removed any users with no friends and less than 200 reviews
- Limited the analysis to one city

3. Cleaning the data

The data are relatively clean, given this is Yelp's official public dataset. However, in order to analyze and model, we'll need to significantly reduce the focus and scope of the data.
https://github.com/dmitriykats1/true_review/blob/master/true_review/notebooks/1.0-dyk-Processing_Data.ipynb

3.1. Data Query

Since the data was loaded into Google BigQuery, we'll first query the entire business data set, which contains all business, in addition to our desired restaurant data. Merging the entire business and review datasets would take too long on a local machine, so we'll let Google handle it. We'll then save the resulting dataframe as a CSV for easy access locally.
As stated in section 2, we'll focus on only the described attributes of the dataset.
Resulting csv file will contain the following:
City Name: Scottsdale

- Number of Users: 76,011 → after filtering → 3,119 (4%)
- Number of Restaurants: 1,322 → after filtering → 1,210 (92%)
- Number of Reviews: 173,062 → after filtering → 24,738 (14%)

4. Exploratory Data Analysis (EDA)

Yelp has been around since 2004, helping people find great local businesses by presenting an easy to read star rating along with more detailed users' reviews. By the beginning of 2019, the number of users and reviews has exploded to over 100M and 184M, respectively. We are going to look at a small subset of the data in order to draw insights in review trends and restaurant trends and use these insights to guide users to a better experience by predicting their personal rating of a specific restaurant. We'll first look at the businesses that are tagged as a non-fast-food restaurant. We have data from 2004 through the middle of 2017. Notebook for this section along with all code is located here: https://github.com/dmitriykats1/true_review/blob/master/true_review/notebooks/2.0-dyk-Exploratory_Data_Analysis.ipynb

4.1. Ratings Over Time

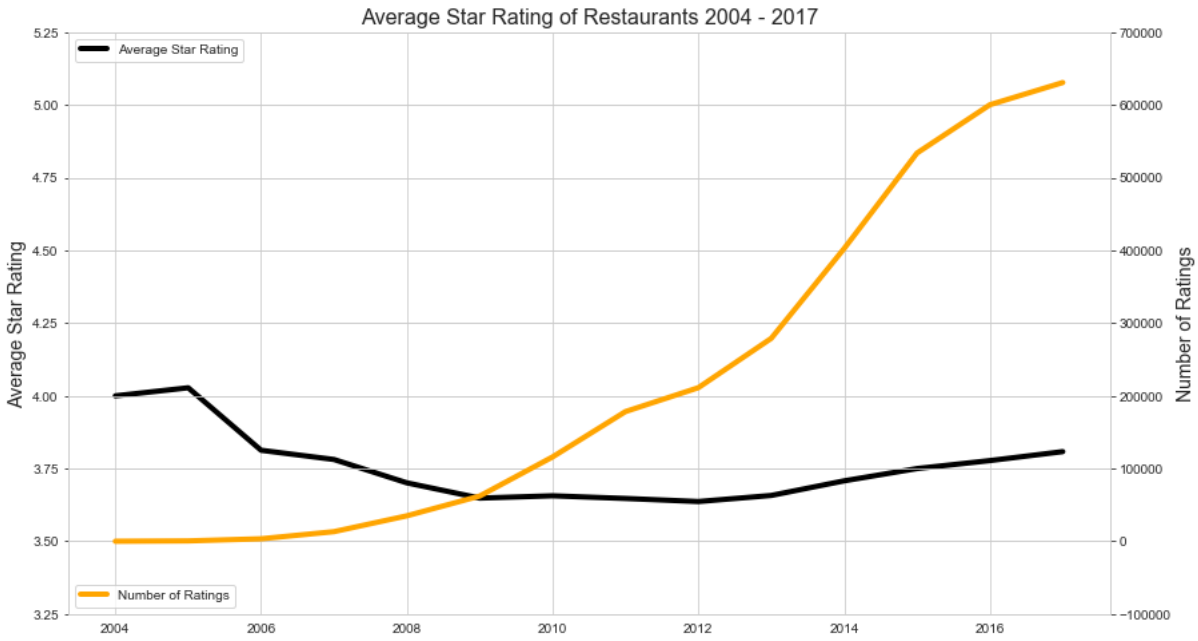


Figure 1- Average Restaurant Rating Over Time

****NOTE THE Y-AXIS SCALE FOR STAR RATINGS DOES NOT START AT A MINIMUM****

We can see from above that the average restaurant rating has fluctuated over time, starting out at around 4 stars, dipping down to 3.6, and currently on an uptrend at around 3.8 stars. We also note that the number of reviews has sky-rocketed starting at about the same time as the downward trend on the average star rating. We'll take a look at the relationship between number of reviews and average star rating later in the EDA, but for now, let's look at what the distribution of ratings looks like over this same time-frame.

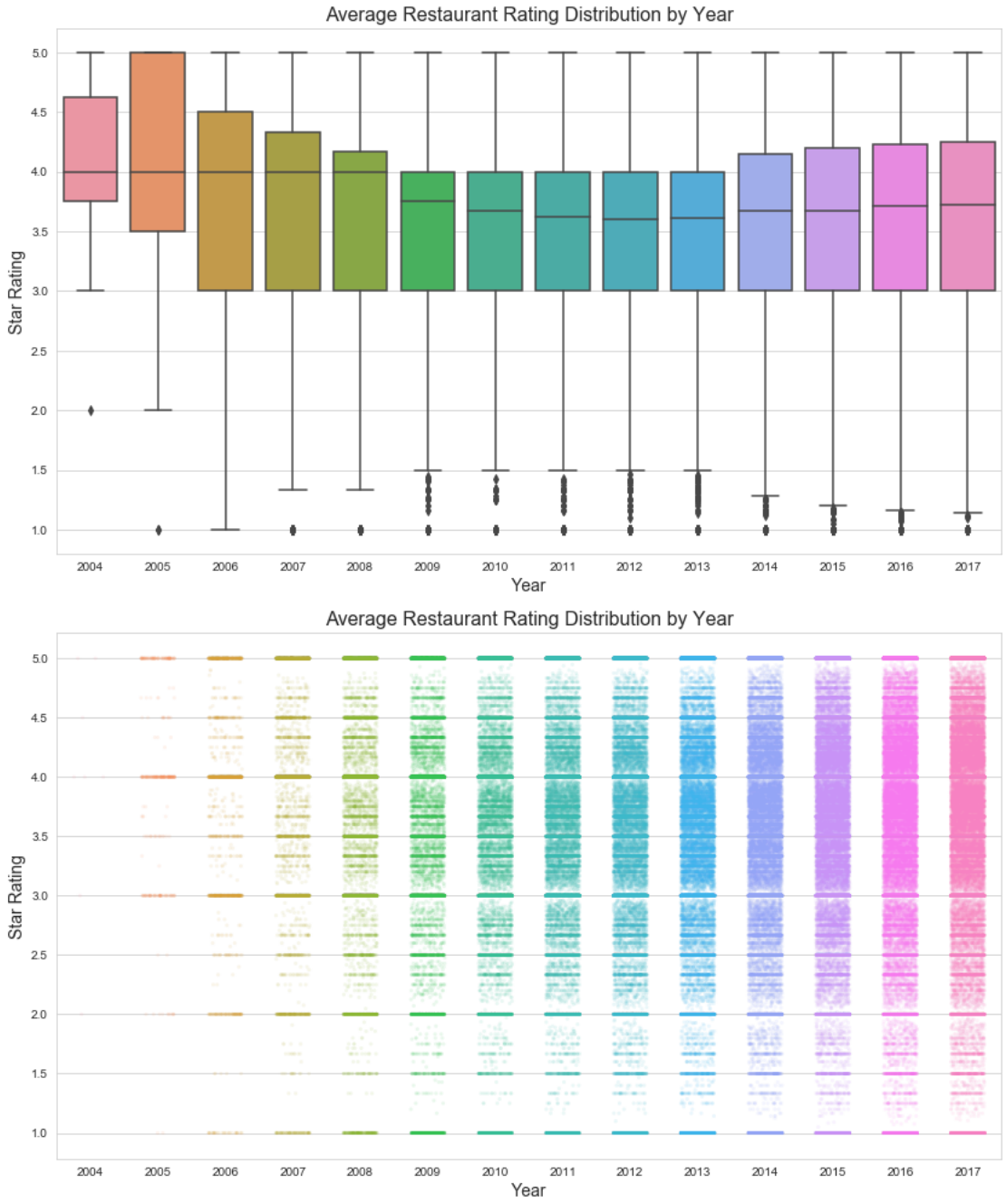


Figure 2 - Distribution of Star Ratings Over Time

Above figures show a distribution of average reviews over the years. The two different figures help us visualize how the spread has increased and how the median / mean shifted over the years.

4.2. Restaurant Categories

In this section we'll take a look at the category breakdown from all the restaurants in the dataset. It should be noted that these tags are manually entered by Yelp users and there are some discrepancies. Additionally, each restaurant is tagged with multiple tags as such the number of occurrences does not add up to number of restaurants. In general, we should get a good idea of the types of restaurants we have.
We took out the words: "Restaurant"," Food", and "Nightlife" since these don't typically contribute to actual food category. We kept "Bar" since some restaurants offer bar food, which can be considered a food category.

	business_id	categories	bus_stars
0	--6MefnULPED_I942VcFNA	Chinese;Restaurants	3.0
1	--9e1ONYQuAa-CB_Rrw7Tw	Cajun/Creole;Steakhouses;Restaurants	4.0
2	--DaPTJW3-tB1vP-PfdTEg	Restaurants;Breakfast & Brunch	3.5
3	--FBCX-N37CMYDfs790Bnw	Food;American (New);Nightlife;Bars;Beer;Wine &...	3.5
4	--GM_ORV2cYS-h38DSaCLw	Pizza;Chicken Wings;Salad;Restaurants	4.0
5	--I7YYLada0tSLkORTHb5Q	Restaurants;Sports Bars;American (Traditional)...	3.5
6	--KCl2FvVQpvjzmZSPyviA	Restaurants;Sandwiches;Pizza	3.0
7	--S62v0QgkqQaVUhFnNHrw	Breakfast & Brunch;American (Traditional);Rest...	2.0
8	--SrzpvFLwP_YFwB_Cetow	Restaurants;Chinese	3.5
9	--U98MNlDym2cLn36BBPgQ	Pizza;Restaurants	3.0

Table 1 - Example of Restaurant Category Labels from the DataFrame

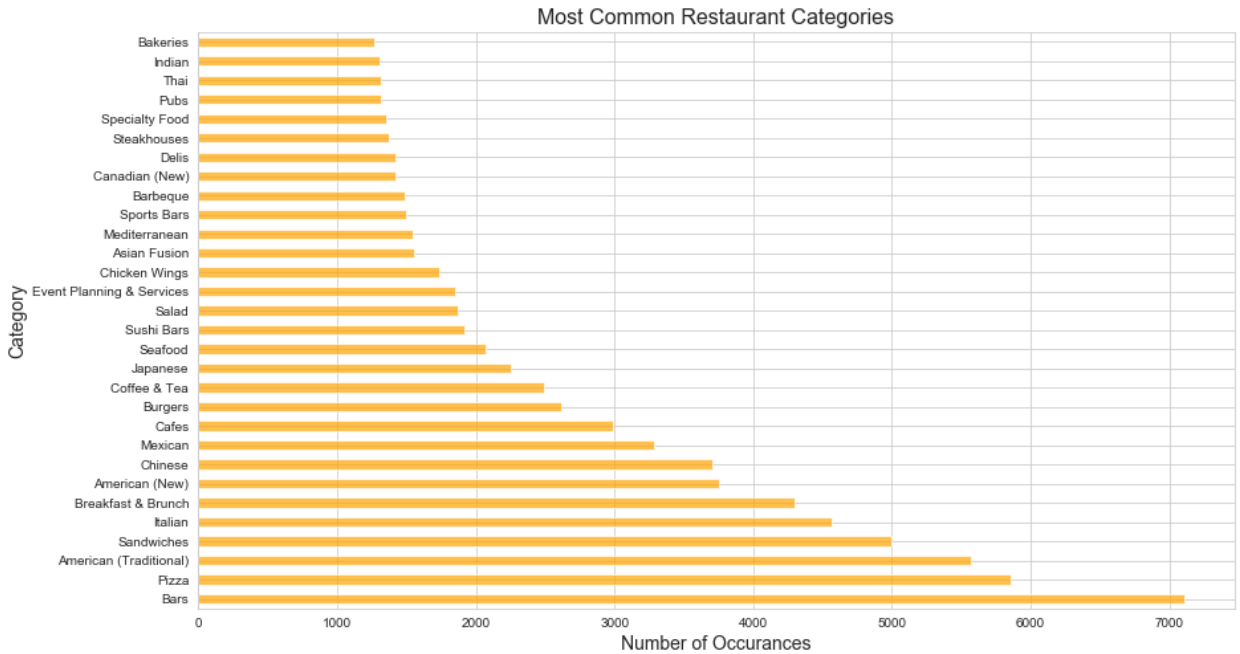


Figure 3 - Number of Category Labels

From the figure above we can see how the categories breakdown among all the restaurants. It's interesting that top category is "bar", followed by mostly junk food / lunch places. We'll use these categories in our feature engineering section, so it's important to understand the breakdown and the fact that same label may be shared among a group of different restaurants.

4.3. Ratings in Different Cities

Yelp operates all over the world; however, their public dataset is for a limited number of cities. We want to select one city for our analysis. Let's now look at what cities we have in our dataset and the distribution of reviews.

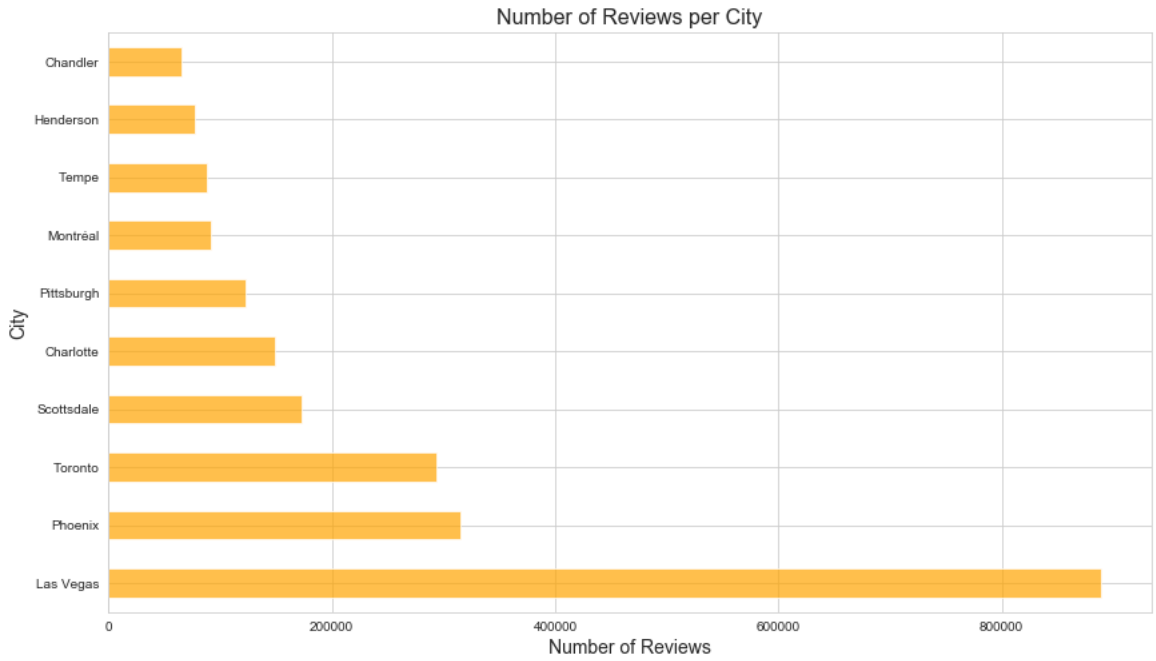


Figure 4 - Number of Reviews for Top Cities

Las Vegas has the most reviews, by far. However, we should look at other city statistics to decide which city to choose. We ultimately want to create a user-restaurant matrix that is not sparse. For this we'll need to look at a ratio between the number of reviews for a given city and the total possible combination of reviews (users x restaurants). Minimizing the sparsity will produce a better collaborative recommendation engine.

```
('Henderson', 0.145)
('Chandler', 0.14)
('Tempe', 0.12)
('Scottsdale', 0.076)
('Pittsburgh', 0.049)
('Charlotte', 0.048)
('Phoenix', 0.032)
('Montréal', 0.031)
('Las Vegas', 0.02)
('Toronto', 0.015)
```

Although Henderson, Chandler, and Tempe have higher sparsity, we'll pick the next city on the list, Scottsdale; it has double the reviews available for analysis.

4.4. Highest Rated and Reviewed Restaurant

Before we look into an individual city, let's take a deeper dive into one of the highest rated and reviewed restaurants in our dataset: Gordon Ramsay BurGR in Las Vegas, with over 2000 5-star ratings. We'll take a look at user checkins and reviews for this restaurant and compare to the other highly rated restaurants in Vegas located on The Strip.

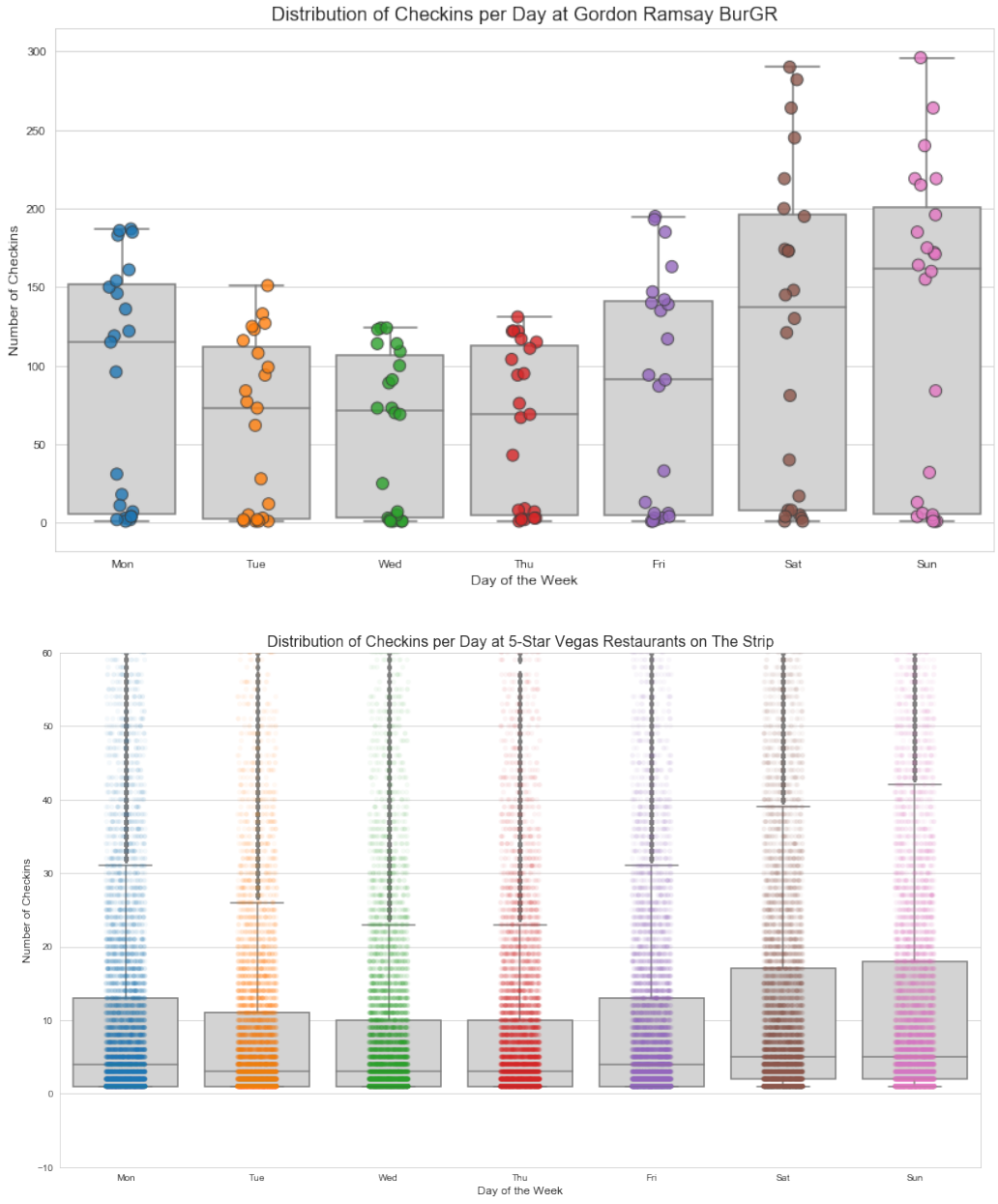


Figure 5 - Number of Checkins Each Day for Gordon Ramsay BurGR (TOP) and Other High Rated Restaurants on The Strip (BOTTOM)

We see a similar pattern for the rest of The Strip restaurants as we do for Gordon Ramsay BurGR, where the middle of the week tends to be slower and the checkins pick up during the weekend and into Monday.

Since Gordon Ramsay BurGR is a highly rated restaurant, we can see how the average rating looks over time and if there any spikes in distribution of ratings to check for any potential influx of "fabricated" ratings.

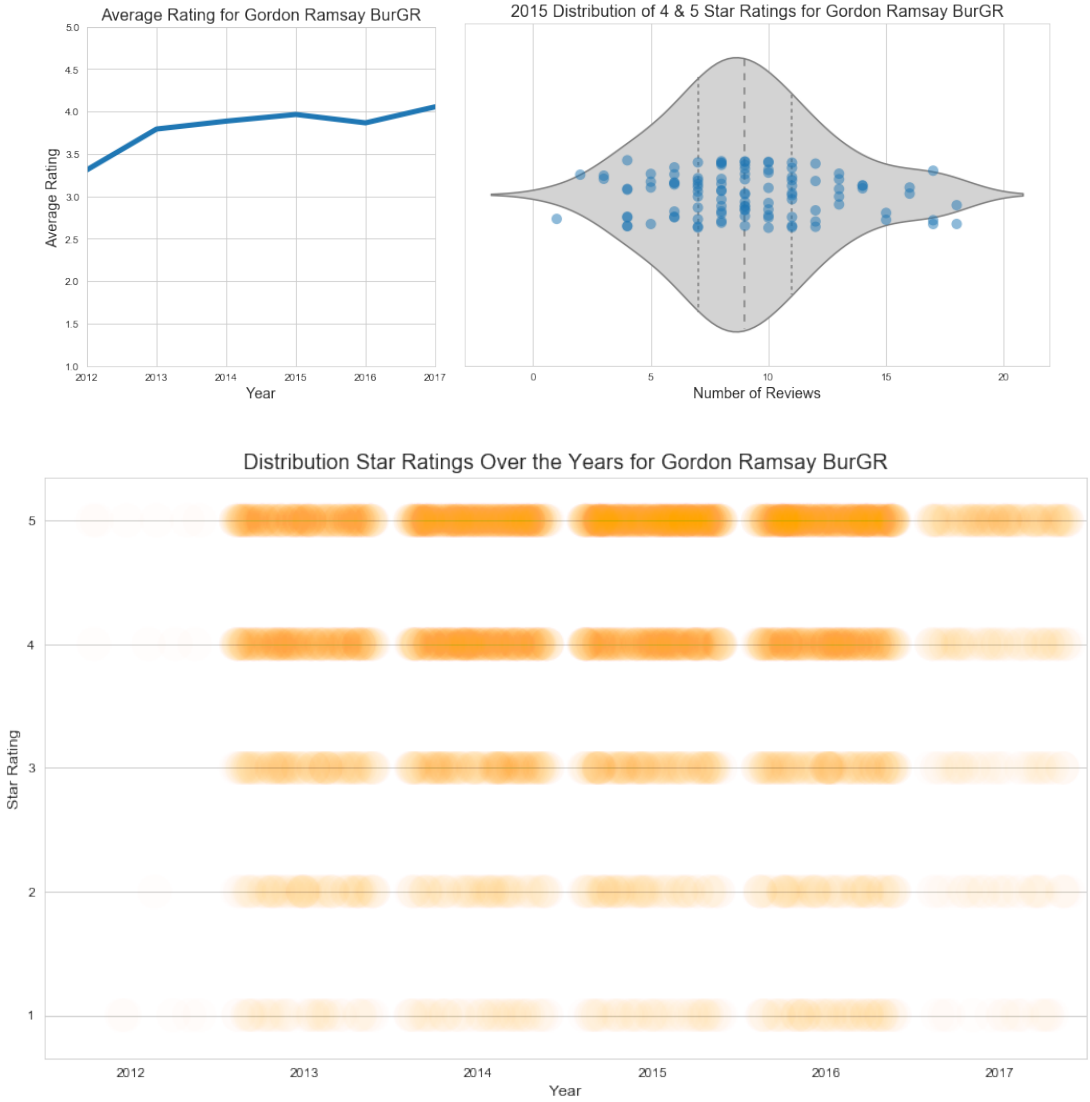


Figure 6 - Detailed Figures for Gordon Ramsay BurGR

In general, we don't see any spikes in number of reviews and the average remains relatively flat over time.

4.5. Users and Their Friend

Yelp also provided us with user data, along with who they are connected to on the platform. Let's take a look at how different users review. Let's first see how review based on how many friends they have.

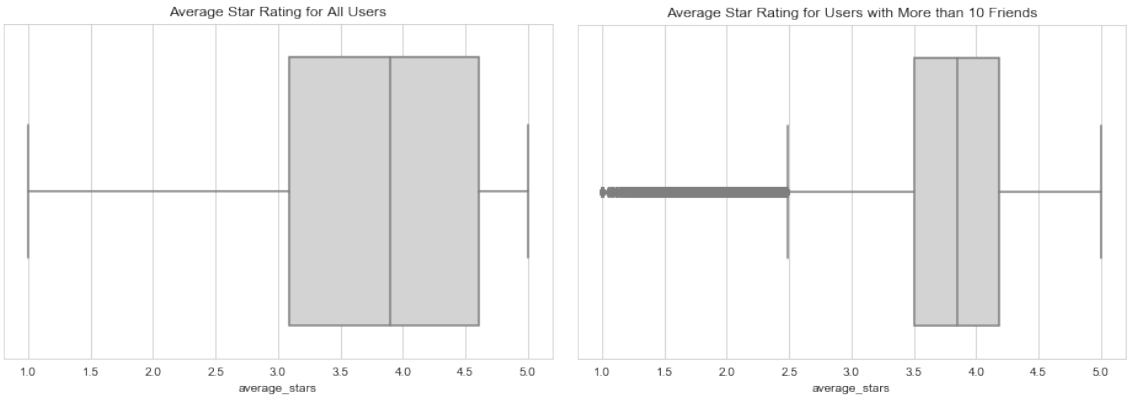
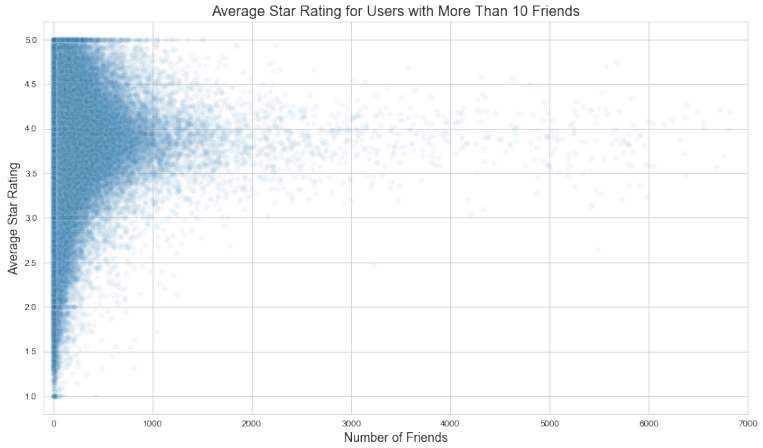
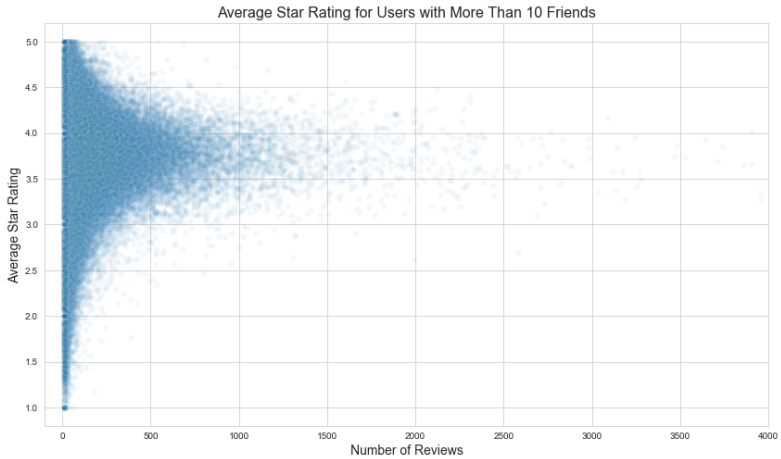


Figure 7 - Rating Distribution for All Users (Left) and Users with more than 10 Friends (Right)

Next let's see what the actual scatter of ratings looks like.



We notice an interesting phenomenon, the more friends a user has the closer the average rating are to the mean. This will presumably hold for a scatter plot looking at number of reviews vs average star rating.



4.6. NLP – Natural Language Processing

In this section we are going to be looking at the user review text and see if we can extract any meaningful information about the restaurants and users. We'll continue to focus on Scottsdale dataset, as we'll eventually build a model based on this data. As with any NLP model we'll need to pre-process the data first, taking out any stop-words, punctuation, and any other non-useful words. We'll then assign topics to each review utilizing gensim's LDA model. This will give us more insight into not only each user but also the restaurant in a numeric form. Notebook for this section along with all code is located here: https://github.com/dmitriykats1/true_review/blob/master/true_review/notebooks/3.0-dyk-Natural_Language_Processing.ipynb

Let's start by looking at a sample review from our dataset:
Random review for Eddie's House in Scottsdale, AZ:

```
'I really like this place. I have been numerous amt of times and I keep wanting more. The friendl
y bartenders, the exciting chef  (Eddie). The best part of this place besides the good food and c
omfort level is their ALL Night Happy Hour. Yes All night $5 dollar specialty cocktails like an e
spresso martini or wines of the day. All first courses (apps) are half off too. The apps includes
, lambchops ($19), Tuna tartar with wonton chips ($9) and so much more. Great s[ot and yearning f
or more since last night.'
```

We'll start by applying basic pre-processing, taking out unwanted characters, number, and symbols:

```
'I really like this place  I have been numerous amt of times and I keep wanting more  The friendl
y bartenders  the exciting chef   Eddie   The best part of this place besides the good food and c
omfort level is their ALL Night Happy Hour  Yes All night    dollar specialty cocktails like an e
spresso martini or wines of the day  All first courses  apps  are half off too  The apps includes
lambchops        Tuna tartar with wonton chips      and so much more  Great s ot and yearning for
more since last night '
```

We'll then remove any short words (less than 3 characters), convert all words into lowercase, and take out stop-words. For stop-words, we'll use nltk library stop-words and add more as we see fit.

```
'really like this place have been numerous amt times and keep wanting more The friendly bartender
s the exciting chef Eddie The best part this place besides the good food and comfort level their
ALL Night Happy Hour Yes All night dollar specialty cocktails like espresso martini wines the day
All first courses apps are half off too The apps includes lambchops Tuna tartar with wonton chips
and much more Great and yearning for more since last night'
```

This is almost good enough to start applying some advanced techniques to extract meaningful data from the reviews. But we can go further and try to form common phrases from the reviews. We'll do bigrams and trigrams (two- and three-word common phrases, respectively). This will help us classify commonly used phrases such as "happy hour", "ice cream", "bloody Mary", and "ice cream sundae" which are more meaningful when used in the presented combinations vs separately.

```
'numerous amt time keep want friendly bartender exciting chef good_part comfort level night dolla
r specialty_cocktail espresso_martini wine day first_course app half app include lambchop tartar
wonton_chip much great yearning last_night'
```

We have significantly reduced the text length, keeping only meaningful information. This is an important step in the process as we're going to use the cleaned-up reviews to produce our topics and eventually user profiles.

4.7. Latent Dirichlet Allocation (LDA)

We'll go into more detail on this modeling technique in the modeling section, but the main premise is to summarize textual review data into topic categories which we can use to compare user-user, restaurant-restaurant, and user-restaurant similarities to improve our recommender system.

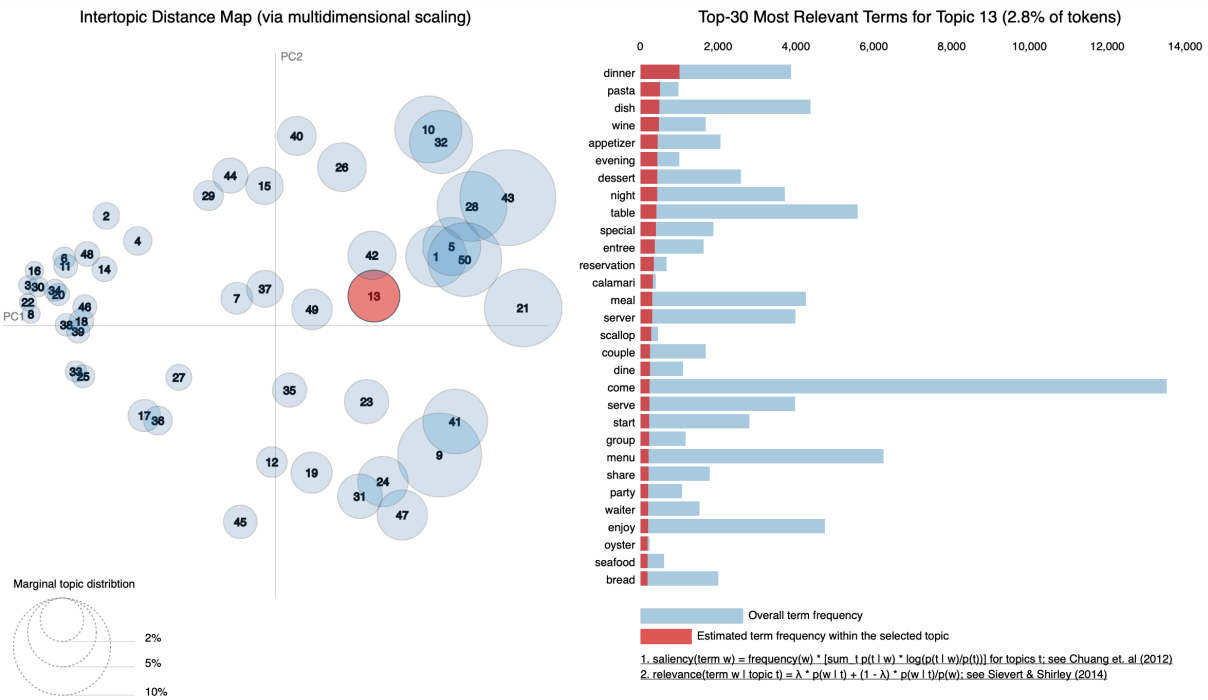
Gensim's LDA library is an unsupervised learning algorithm where topics are automatically "discovered" and presented as a grouping of words and we can then manually characterize the topic. Let's stick with our initial review from the last section:

```
'I really like this place. I have been numerous amt of times and I keep wanting more. The friendl
y bartenders, the exciting chef  (Eddie). The best part of this place besides the good food and c
omfort level is their ALL Night Happy Hour. Yes All night $5 dollar specialty cocktails like an e
spresso martini or wines of the day. All first courses (apps) are half off too. The apps includes
, lambchops ($19), Tuna tartar with wonton chips ($9) and so much more. Great s[ot and yearning f
or more since last night.'
```

And with the LDA model applied, we get the following topic distribution:

```
dinner                    0.21
happy hour, drinks        0.18
cheap, good, service      0.18
lunch                     0.16
buffet                    0.11
healthy                   0.06
```

We can see that these topics are related to the review, especially the top three. The last three are not necessarily related, but they have a lower probability of occurrence. We can visualize the distribution of words for these topics using the pyLDAvis library. (note topics are incremented with starting index at 1, in lieu of python's index starting at 0)



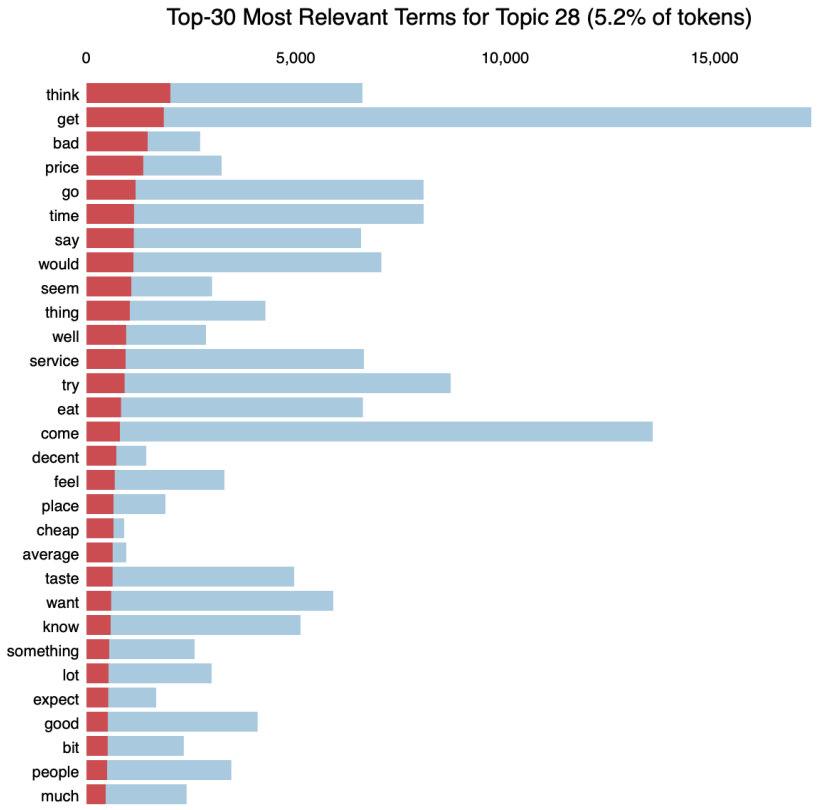
We have selected 50 topics for this analysis, and although there is some overlap, majority of topics are somewhat independent and provide a more diverse distribution for each review. Let's dive deeper and look at the words and their distribution for each topic for the selected review:

```
Topic: 12 - dinner
Words: 0.028*"dinner" + 0.014*"pasta" + 0.013*"dish" + 0.013*"wine" + 0.012*"appetizer" + 0.012*"
evening" + 0.012*"dessert" + 0.012*"night" + 0.011*"table" + 0.011*"special"

Topic: 41 - happy hour, drinks
Words: 0.076*"happy_hour" + 0.072*"drink" + 0.036*"nice" + 0.032*"great" + 0.027*"patio" + 0.024*
"service" + 0.022*"bar" + 0.019*"love" + 0.018*"price" + 0.017*"atmosphere"

Topic: 27 - cheap, good, service
Words: 0.030*"think" + 0.028*"get" + 0.022*"bad" + 0.021*"price" + 0.018*"go" + 0.017*"time" + 0.
017*"say" + 0.017*"would" + 0.016*"seem" + 0.016*"thing"
```

This is where the manual interpretation becomes important. For example, looking at topic 27, we have "bad" as a common word, but there are a lot of other positive or informative words in this topic such as: "price", "time", and "go". Full list:



We will use these topics to create additional attributes for our user profiles.

5. Modeling

The general approach to predicting user ratings will be to use collaborative filtering to initialize predictions for every user / restaurant. Then we'll use other models to boost the predictions and make our model more accurate. Furthermore, we'll add content-based filter by creating user and restaurant profiles and computing similarities to predict whether a user will a certain restaurant or not. Notebook for this section along with all code is located here: https://github.com/dmitriykats1/true_review/blob/master/true_review/notebooks/4.0-dyk-Recommender_Preprocessing.ipynb

https://github.com/dmitriykats1/true_review/blob/master/true_review/notebooks/5.0-dyk-Prediction_Modeling.ipynb

5.1. Overview

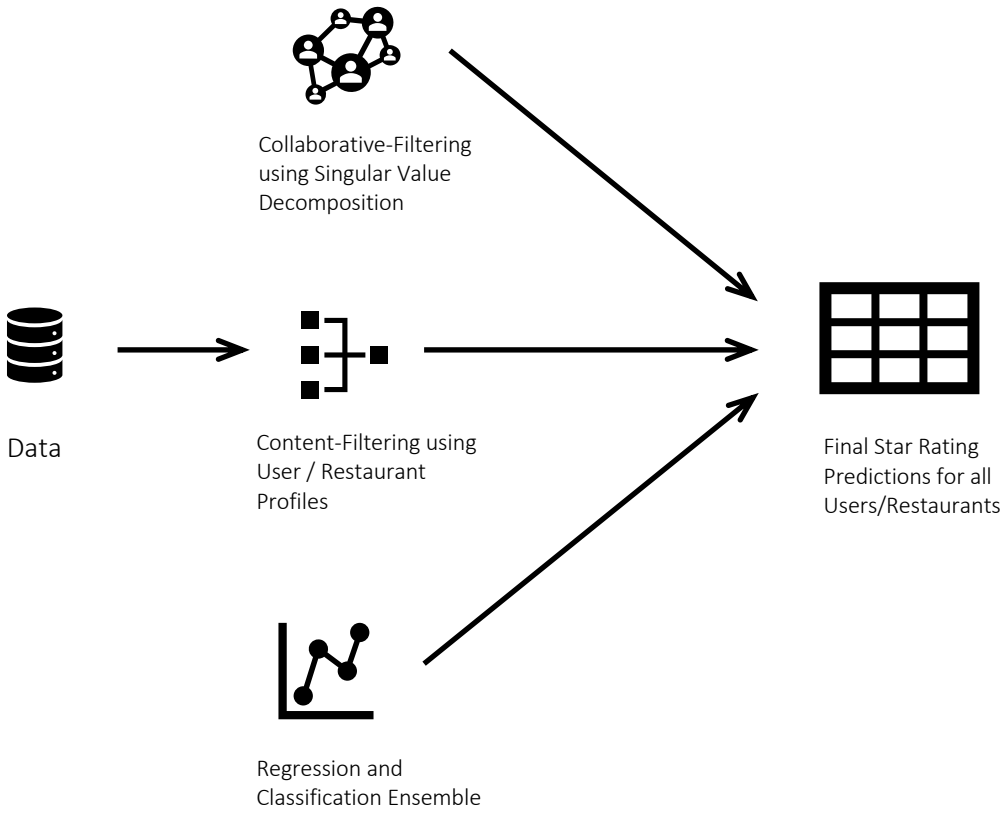
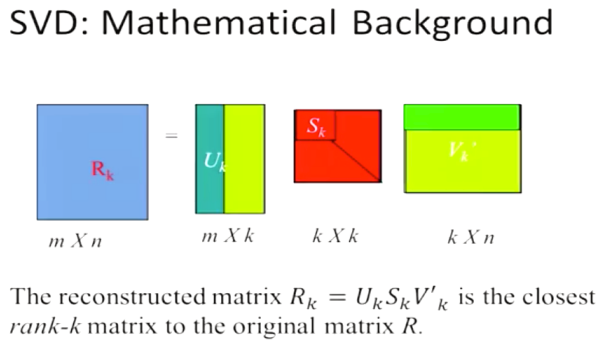


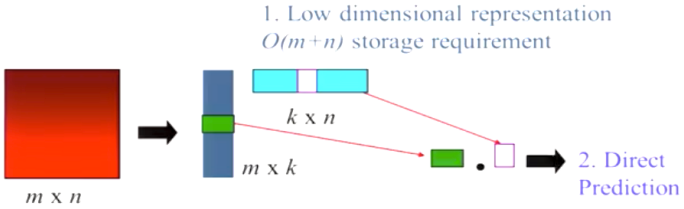
Figure 8 - Overview of Prediction Model

5.2. Collaborative Filtering using SVD

With extremely highly dimensional datasets, such as a user / item matrix, it can be easy to overfit traditional models to specific features rather than the underlying themes of the data. Using singular features to extract useful information will result in poor performance. Instead, we can use a technique called Singular Value Decomposition (SVD) to reduce a large set of features into a smaller set of themes.



Factor matrix R into u and s. The diagonal in S represents importance of features. We can then sort the features in descending order (most important to least important) and cut off the matrix at some item k. In other words, we can reduce the number of given dimensions by matrix R to a defined number of dimensions, k.



Expressing each user's taste/style in a single vector of k-taste/style values and each restaurant's description in same k-taste/style values, then we can compute a dot product to predict how much a user will like a given restaurant.

There are many ways to produce a collaborative model and, in this analysis, we'll be using Surprise SVD for the initial predictions.

5.3. Content-Based Filtering

We'll start by creating a restaurant attributes matrix from the two datasets provided by Yelp and an additional dataset that was created using NLP and LDA Topic Modeling.

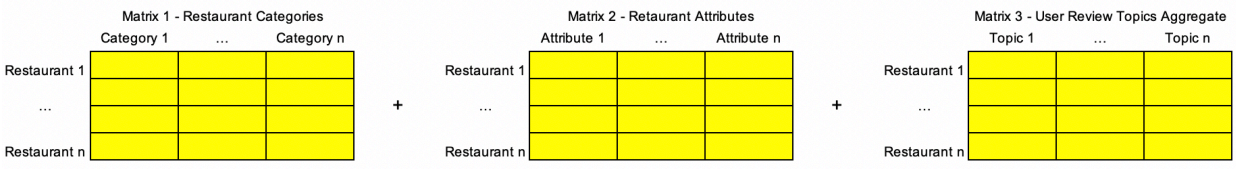


Figure 9 - Matrices Making up the Restaurant Attributes

Then, using the reviews for each user we'll construct a Users' Attributes Matrix by assigning 1 if a user has rated the restaurant 4 or 5 and -1 if the rating was below 4. If a user has not been to the restaurant or has not rated it, we'll assign a zero. Once the user matrix is formulated, we can compute a profile for each user via a dot product between user attributes and restaurant attributes:

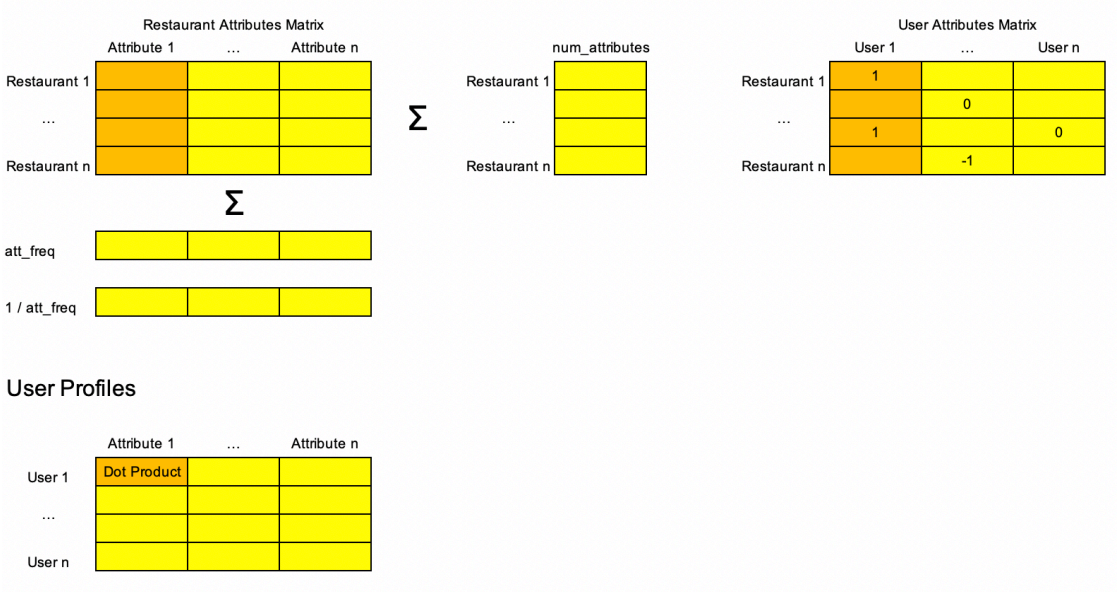


Figure 10 – Forming User Attributes and Profiles

Finally, we can make predictions by computing the dot product between restaurant attributes and user attributes for each User / Restaurant combination.

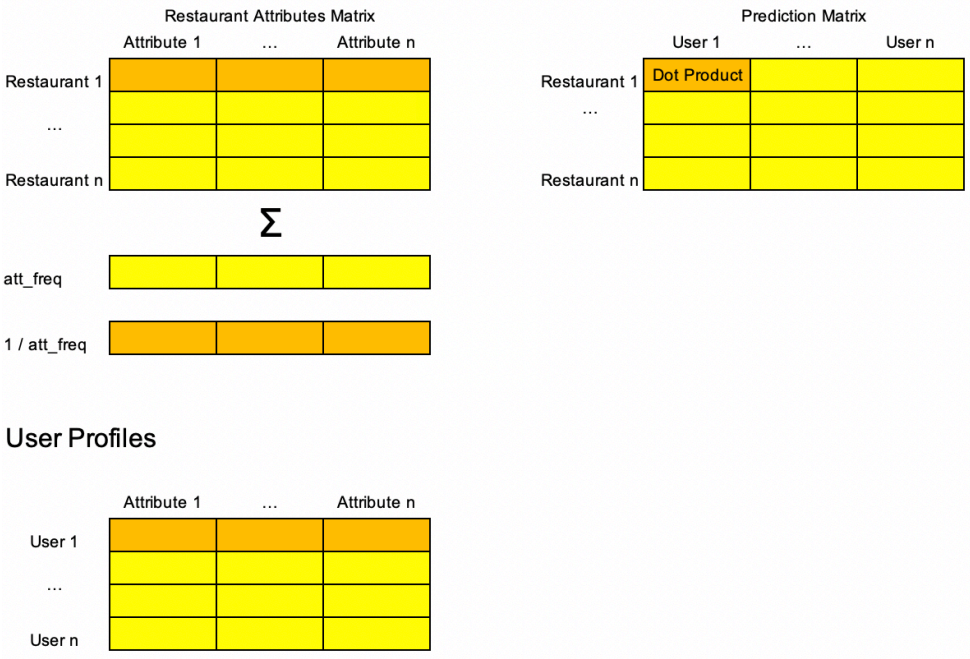
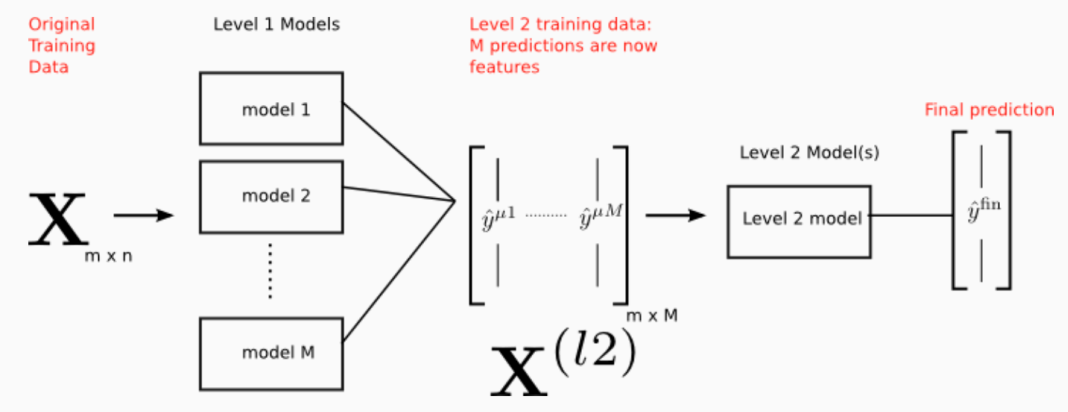


Figure 11 - Final User Prediction Matrix

5.4. Classification and Regression Models

In this section we'll discuss using multiple regression models to predict a user's star rating. There are a few popular techniques which allow for combining different models into a single predictor. This is known as Ensemble learning. Although, boosting models such as XGBoost, AdaBoost, and Random Forest provide boosting internally, we can combine these models and improve our overall performance. We'll be using Stacking and Weighted Average Ensembles in our models.



Below are just some of the algorithms available for classification and regression modeling.

	Advantages	Disadvantages
Linear Regression	Very easy to implement and comprehend. No tuning required. Can be used in an Ensemble model	No tuning. Won't perform well on non-linear models.
LightGBM	Fast training, low memory usage, better accuracy. Parallel learning supported.	Not advisable on small datasets and can easily overfit.
Random Forest	Easy to implement, use, and tune. Very easy to interpret and explain as this is an ensemble of decision trees. Not a lot of parameters to tune. Very fast and proven to be accurate.	Although not necessary, some tuning is required to achieve optimal performance.
XGBoost	High model performance and execution speed.	A lot of parameters to tune.

Table 2 Modeling Techniques

We are going to use XGBoost, LightGBM, and AdaBoost in our Level 1 models and then Linear Regression as our Level 2 model.

We'll also deploy the Random Forest Classification model and compare it to our content-based filtering model.

5.4.1. Regression Models

Using XGBoost, LightGBM, and AdaBoost to predict users' star ratings we get the following results:

	RMSE - Train	RMSE - Test
XGBoost	0.86	0.89
LightGBM	0.91	0.94
AdaBoost	0.93	0.95

Using the train predictions from these three models, we'll fit them to our linear regression model:

	RMSE - Train	RMSE - Test
XGBoost	0.86	0.89
LightGBM	0.91	0.94
AdaBoost	0.93	0.95
Linear Regression	0.85	0.88

Figure 12 - Stacked Ensemble Results

5.4.2. Surprise SVD Model

The SVD algorithm, as discussed in section 6.1 relies on matrix factorization methods and will be used here as part of collaborative filtering, to predict initial ratings for restaurants. We'll use RMSE as our performance metric to stay consistent with the Random Forest model.
Surprise has an API available to get help us started with our collaborative model.

Initial run is, again, untuned and results in RMSE of 0.961 which is on par with our other regression models. Again, utilizing GridSearchCV, we improve the RMSE to 0.947. And after running the model on the entire dataset, we can add the results to our predictions table:

	RMSE
SVD - Initial	0.96
SVD - Tuned	0.95
SVD – Entire Dataset	0.99
SVD + Stacked Ensemble	0.85

Figure 13 - Stacked and Weight Balanced Ensemble

5.4.3. Random Forest Classifier

Random Forest Classifier was used to predict if a user will like or dislike a given restaurant. We'll use the same basis as content-based filtering to create a target column. We'll use 1 if a user has rated the restaurant 4 or 5 and 0 if the rating was below 4.

```
Average precision-recall score: 0.78
Precision score: 0.72
Recall score: 0.67
```

5.4.4. Content-Based Filtering

Our content-based predictions range from negative numbers to positive numbers. The negative values represent restaurants that a given user will not like or will rate <4 stars and positive numbers correspond with restaurants that a given user will 4 or 5 stars.

```
Average precision-recall score: 0.74
Precision score: 0.70
Recall score: 0.72
```

5.5. Final Model

In order to combine our Content-filter and RF Classifier, we'll apply a simple averaging algorithm, and round the results up in order to capture all liked predictions.

```
Average precision-recall score: 0.77
Precision score: 0.73
Recall score: 0.73
```

In order to combine our average classified model of likes and dislikes we need to decide on an algorithm. Since we chose a 4-star rating as a divide between likes and dislikes, we can split the user reviews into two categories. Likes will only have 4 and 5-star ratings and dislikes will have 1, 2, and 3-star ratings. We can average the star ratings by adding the averages of each group to our average ensemble rating and taking a weighted final average to come up with a prediction.

for avg_ensemble:
    if avg_prediction == 1:
        *final_rating = avg_ensemble * 0.8 + liked_average * 0.2*
    else:
        *final_rating = avg_ensemble * 0.8 + dislike_average * 0.2*

where:

liked_average = average of the 4- and 5-star reviews in the dataset
disliked_average = average of the 1, 2, and 3-star reviews in the dataset

Equation 1 - Averaging Algorithm

	RMSE
SVD - Initial	0.96
SVD - Tuned	0.95
SVD – Entire Dataset	0.99
SVD + Stacked Ensemble	0.85
Avg. Ensemble + Avg. Classifier	0.85

Figure 14 - Final Results

6. Conclusions and Next Steps

Due to the nature of the dataset, many features used for our modeling were readily available from Yelp. However, given the strong correlation between a users' review text and their rating, we needed to extract as much information from the reviews as possible to improve our modeling techniques. Using LDA Topic modeling we were able to generalize 25,000 reviews into just 50 topics that are easy to understand. Using these additional "features" we predicted a given user's rating and whether they would like a specific restaurant.

Numerous models were used on the dataset to make predictions. Alone, these models performed relatively well, but combining them resulted in a much better prediction as measure by total RMSE of 0.85.

Although the final goal of combining collaborative and content-based models has been achieved, there are many ways to improve the performance of this particular application.

- LDA Topic model can include reviews from entire dataset to provide a wider range of language dynamic and regional dialects
- Extract latent features from the SVD model to better understand feature creation and improve overall results
- Derive a better averaging algorithm for content-based filter and stacked ensemble