Yelp Review Analysis

An extended analysis of Yelp restaurant reviews by Xiangyu Chen/Springboard Data Scientist Career Track 2017-18

Table of contents:

1. Motivation of this project and how the analysis would be useful to many people.

Yelp is one of the largest online crowd-sourced rating platforms for many different businesses where consumers can post their experience with the business as well as giving the business a star rating between 1 and 5 with 1 being the worst and 5 being the best. Below you will find a picture showing different categories of businesses being reviewed and rated on Yelp.

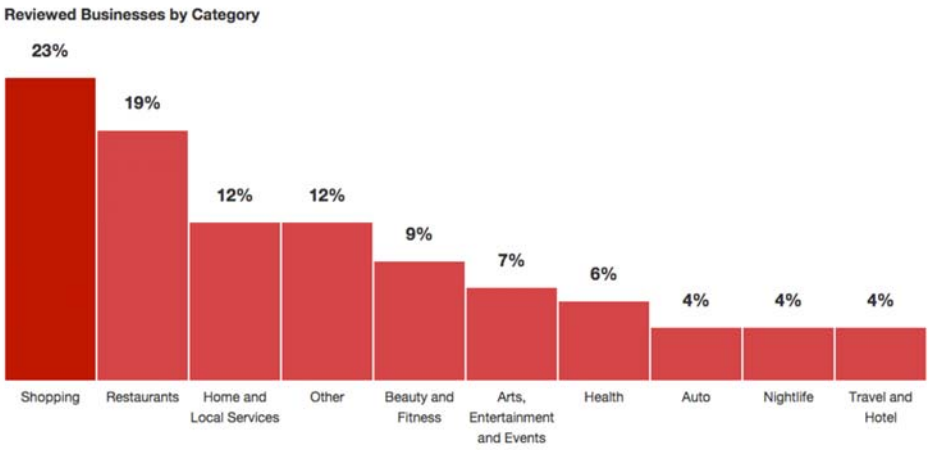

Figure 1: Business distribution on Yelp (Source)

This platform benefits many people because before they choose service from a business, they can check the reviews and ratings of that business to help them make a decision. The review text itself also contains useful information about a certain merchant. From the above figure, we can see that restaurants are the 2nd largest category on Yelp. As a food lover, I chose to analyze the restaurant data and perform my analysis.

The user base on Yelp.com grows every year and so does the number of reviews. As of the first quarter of 2018, the total number of reviews has reached 155 million as shown by the following figure.

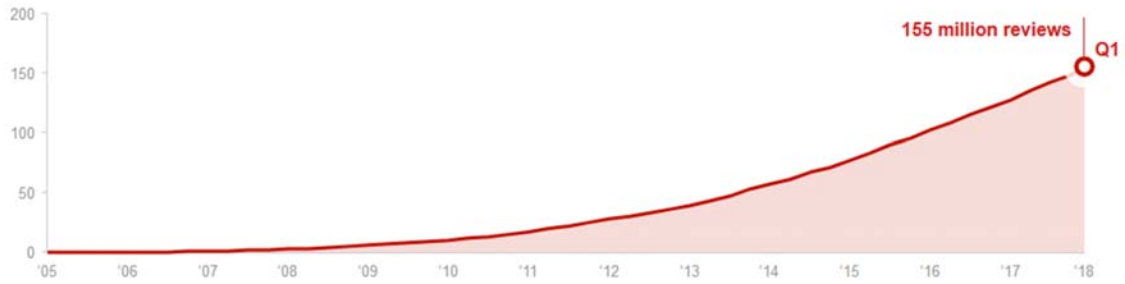

Figure 2: Cumulative number of reviews on Yelp (Source)

This shows how popular this kind of online rating system has become as more and more people would utilize the crowd wisdom when choosing a business. Among all services, food service is especially important as food eaten can not be returned. Yelp already has a good system for providing restaurant reviews and ratings. I have used Yelp when picking a restaurant before. My analysis is aimed to accomplish something more from the data such as a weighted rating system to take into account of the number of reviews for a restaurant and the time the rating was made. Besides, I will also cluster good reviews (4 and 5-star ratings) to see what are the factors that are indicative of a great restaurant. Finally, I will build recommender systems using various techniques.

2. **Data acquisition and wrangling**

The complete Yelp review data set is available on Yelp website as json files. This data set includes five different files:

- 'business.json' contains information about the business such as name, address, location, number of reviews, category and operation hours.
- 'user.json' contains user information such as name of user, review count and join date etc.
- 'review.json' is the main file that contains all the textual reviews as well as the user that made the review and the business the review is for.
- 'checkin.json' and 'tip.json' contain some other complementary information which I will not be using in this analysis

The link to the data set is below:

Yelp Data Set

The json files were read in using Pandas package for Python (Pandas website) by chunks as data frames and then concatenated. The first one was 'review.json'. It contained very few missing values. Here are the first 5 rows of the data.

	business_id	cool	date	funny	review_id	stars	text	useful	user_id
0	0W4lkclzZThpx3V65bVgig	0	2016-05-28	0.0	v0i_UHJMo_hPBq9bxWvW4w	5.0	Love the staff, love the meat, love the place....	0.0	bv2nCi5Qv5vroFiqKGopiw
1	AEx2SYEUJmTxVVB18LlCwA	0	2016-05-28	0.0	vkVSCC7xljjrAI4UGfnKEQ	5.0	Super simple place but amazing nonetheless. It...	0.0	bv2nCi5Qv5vroFiqKGopiw
2	VR6GpWIda3SfvPC-lg9H3w	0	2016-05-28	0.0	n6QzlUObkYshz4dz2QRJTw	5.0	Small unassuming place that changes their menu...	0.0	bv2nCi5Qv5vroFiqKGopiw
3	CKC0-MOWMqoeWf6s-szl8g	0	2016-05-28	0.0	MV3CcKScW05u5LVfF6ok0g	5.0	Lester's is located in a beautiful neighborhoo...	0.0	bv2nCi5Qv5vroFiqKGopiw
4	ACFtxLv8pGrrxMm6EgjreA	0	2016-05-28	0.0	IXvOzsEMYtiJI0CARmj77Q	4.0	Love coming here. Yes the place always needs t...	0.0	bv2nCi5Qv5vroFiqKGopiw

Figure 3: Reviews data frame

The 'stars' column contains the star ratings for each business while the 'text' column contains the actual review text by each user which is what I will use for my star prediction and topic clustering.

The second file that was loaded was 'business.json'. This file contained detailed information about the business. There is a nested dictionary in this file names 'attributes'. This dictionary had a lot of the attributes of the businesses such as whether they accept credit cards, whether there are parking garages, or whether wifi is available etc. The file was read in line by line and then normalized to flatten the nested dictionary items. After flattening, there were 34 attribute columns which later on could be used as important features in machine learning. Below was an unflattened attribute:

```
"{'BusinessParking': {'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False}, 'HairSpecializesIn':
{'coloring': True, 'africanamerican': False, 'curly': True, 'perms': True, 'kids': True, 'extensions': True, 'asian': True, 'st
raightperms': True}, 'BusinessAcceptsCreditCards': True, 'RestaurantsPriceRange2': 3, 'GoodForKids': True, 'ByAppointmentOnly':
False, 'WheelchairAccessible': True}"
```

Figure 4: Business nested attributes dictionary

After flattening the file, the attributes looked like the following figure:

	address	attributes.AcceptsInsurance	attributes.AgesAllowed	attributes.Alcohol	attributes.Ambience.casual	attributes.Ambience.classy	attributes.Ambience
0	4855 E Warner Rd, Ste B9	True	NaN	NaN	NaN	NaN	
1	3101 Washington Rd	NaN	NaN	NaN	NaN	NaN	

Figure 5: Flattened attributes (not fully shown)

To only get data for restaurants, I searched the 'category' column with regular expression containing either 'restaurant' or 'food', then filtered the business data frame to get all restaurants. This new data

frame was joined with the review data frame to combine all the information. The combined data frame contained 3.5 million rows of data with a total of 109 columns. Then columns with more than 80% null values were dropped.

3. **Exploratory Data Analysis (EDA)**

To begin building my machine learning model to predict star ratings, feature exploration was performed on the data set to see which factors affect the star rating a user might give to a restaurant. The distribution of all the star ratings was shown below:

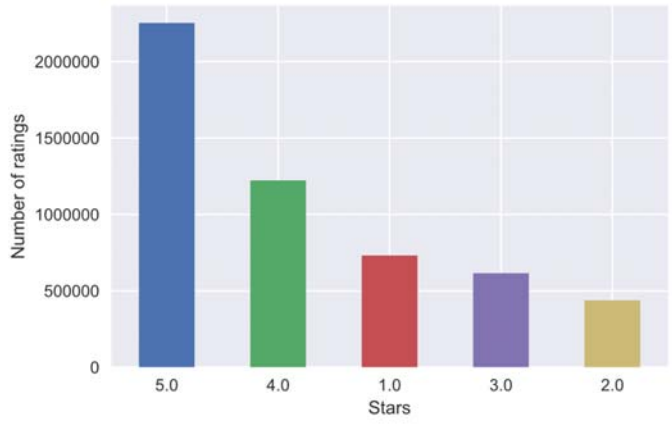

Figure 6: Distribution of all star ratings for restaurants

As shown by the bar graphs, 5-star ratings had the highest fraction and 4-star ratings were second then followed by 1-star ratings. So, there was clearly class imbalance in this data set. Next visualization was to see what the average star ratings looked like in the whole data set (figure 7).

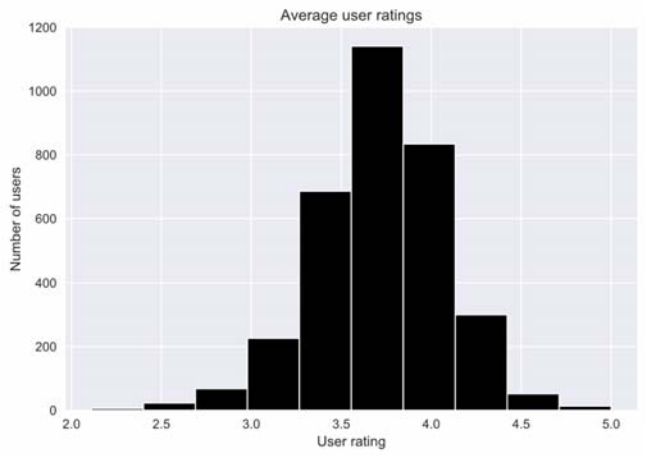

Figure 7: Average user ratings of Yelp restaurants data

The average rating for restaurants was mostly around 3.7.

The next thing was to look at what were the most frequent words associated with good ratings such as 4 and 5 stars. A sample of the entire data set was taken randomly and then the review texts were tokenized using NLTK library. Stop words and punctuations were removed. A word cloud was plotted to show the most frequent words for good restaurants (Figure 8).

Figure 8: Word cloud for good restaurant reviews

Words that were most frequently associated with good restaurants included: ‘amazing’, ‘best’, ‘love, and ‘good’, etc. which made sense.

Next, all restaurants attributes were explored to see which ones had an impact on star ratings (figure 9).

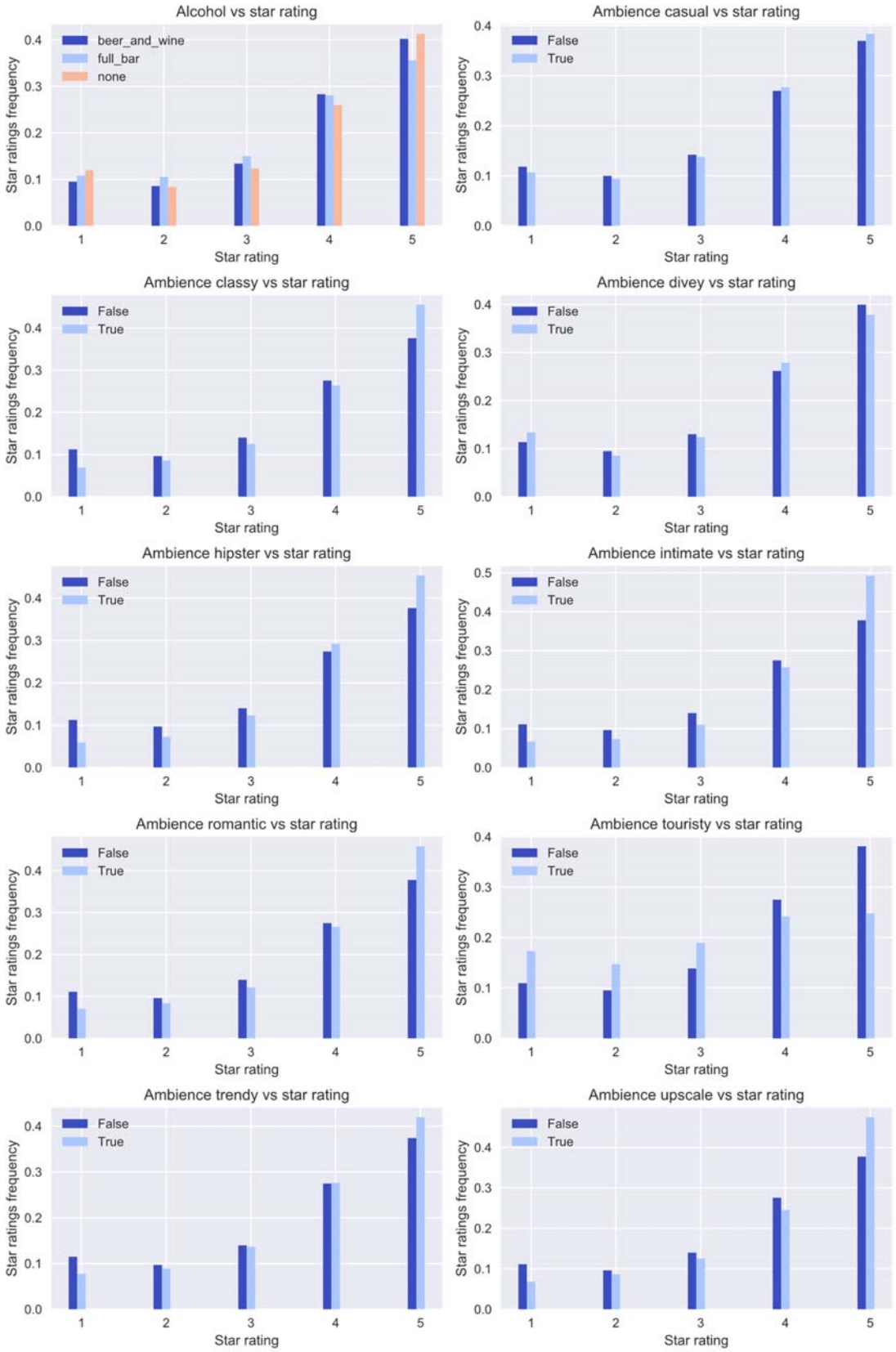

Figure 9: Restaurant attributes vs star ratings (part 1)

More attributes (figure 10).

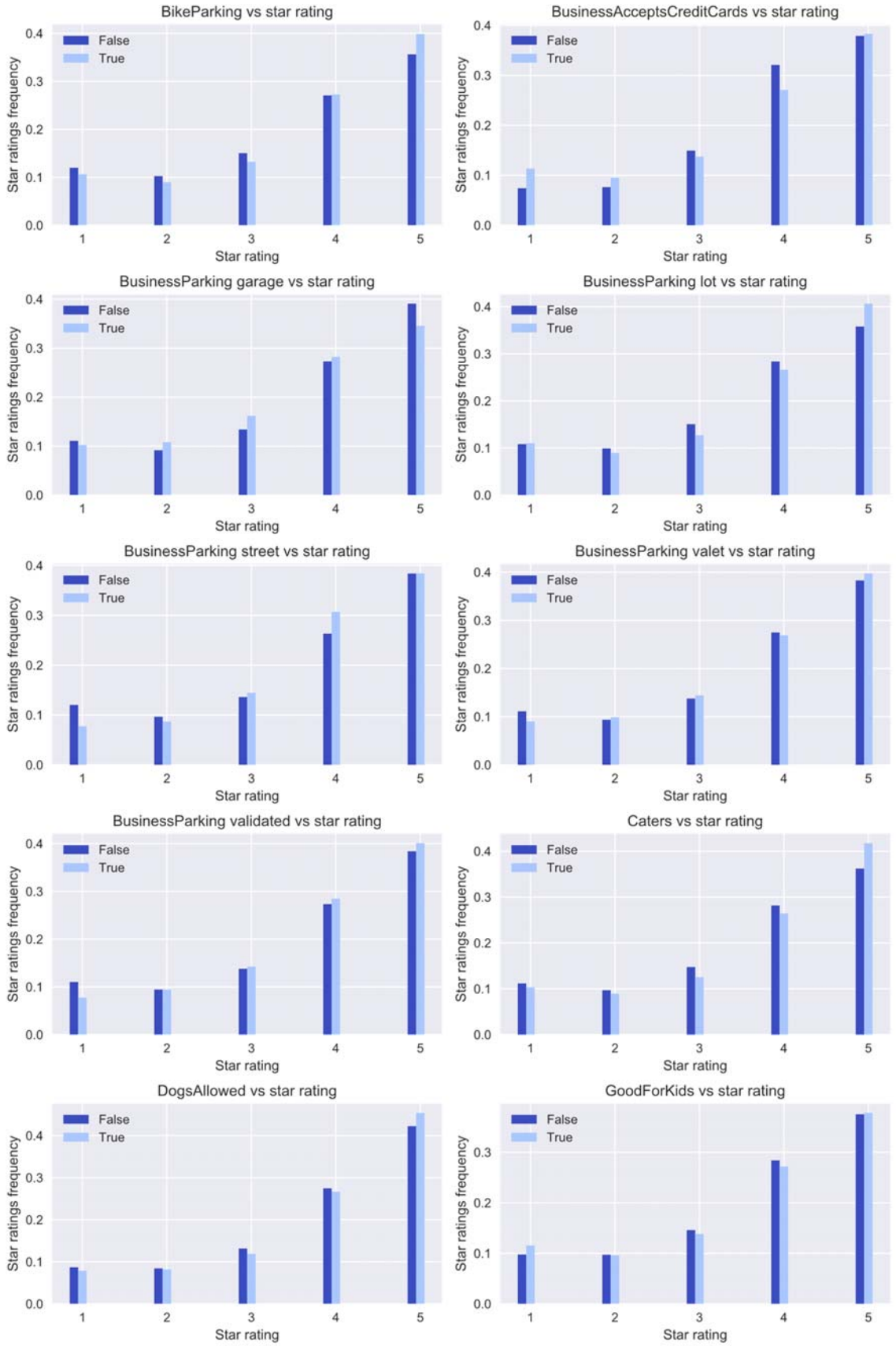

Figure 10: Restaurant attributes vs star ratings (part 2)

More attributes (figure 11).

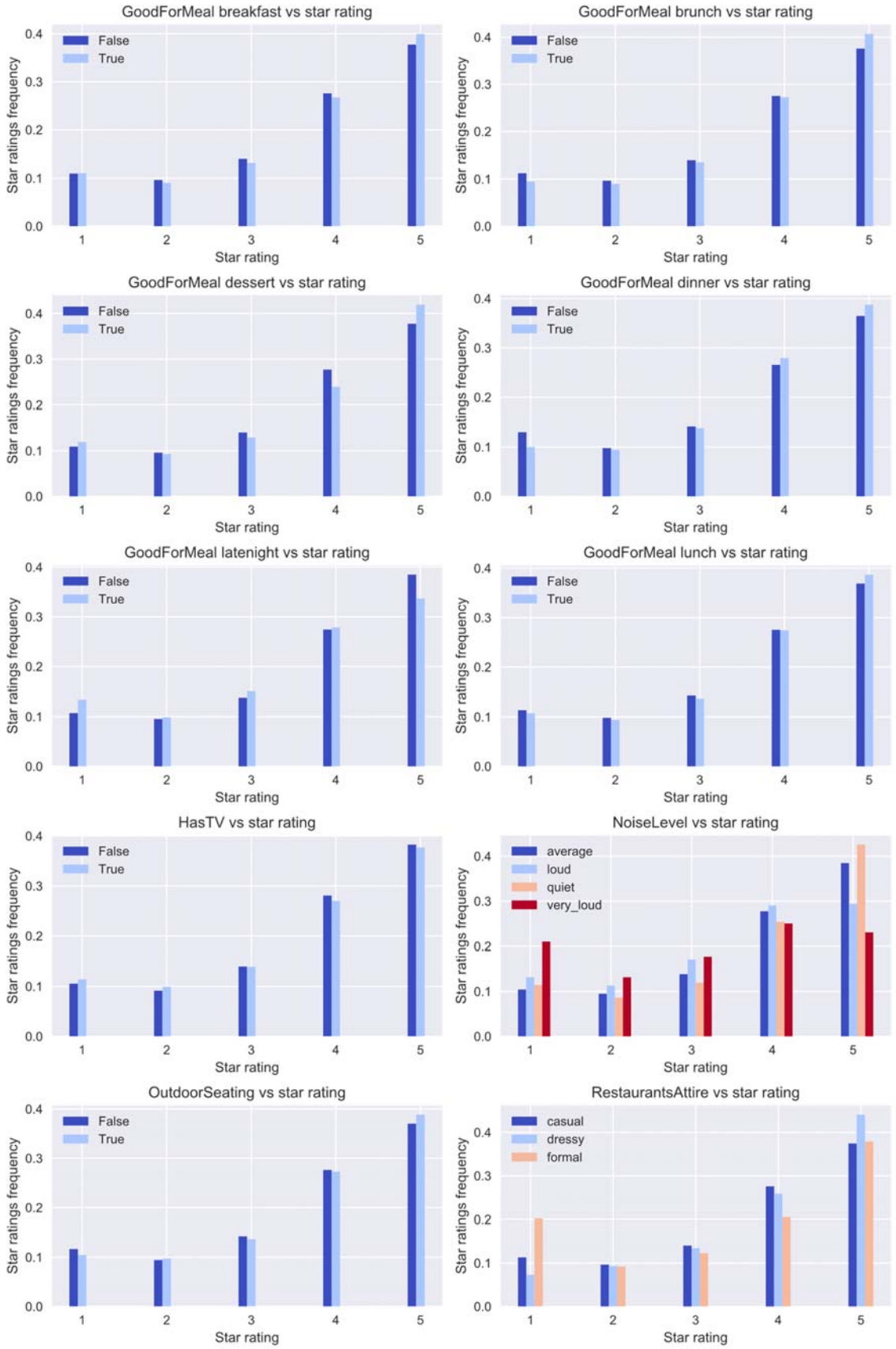

Figure 11: Restaurant attributes vs star ratings (part 3)

Last attributes (figure 12).

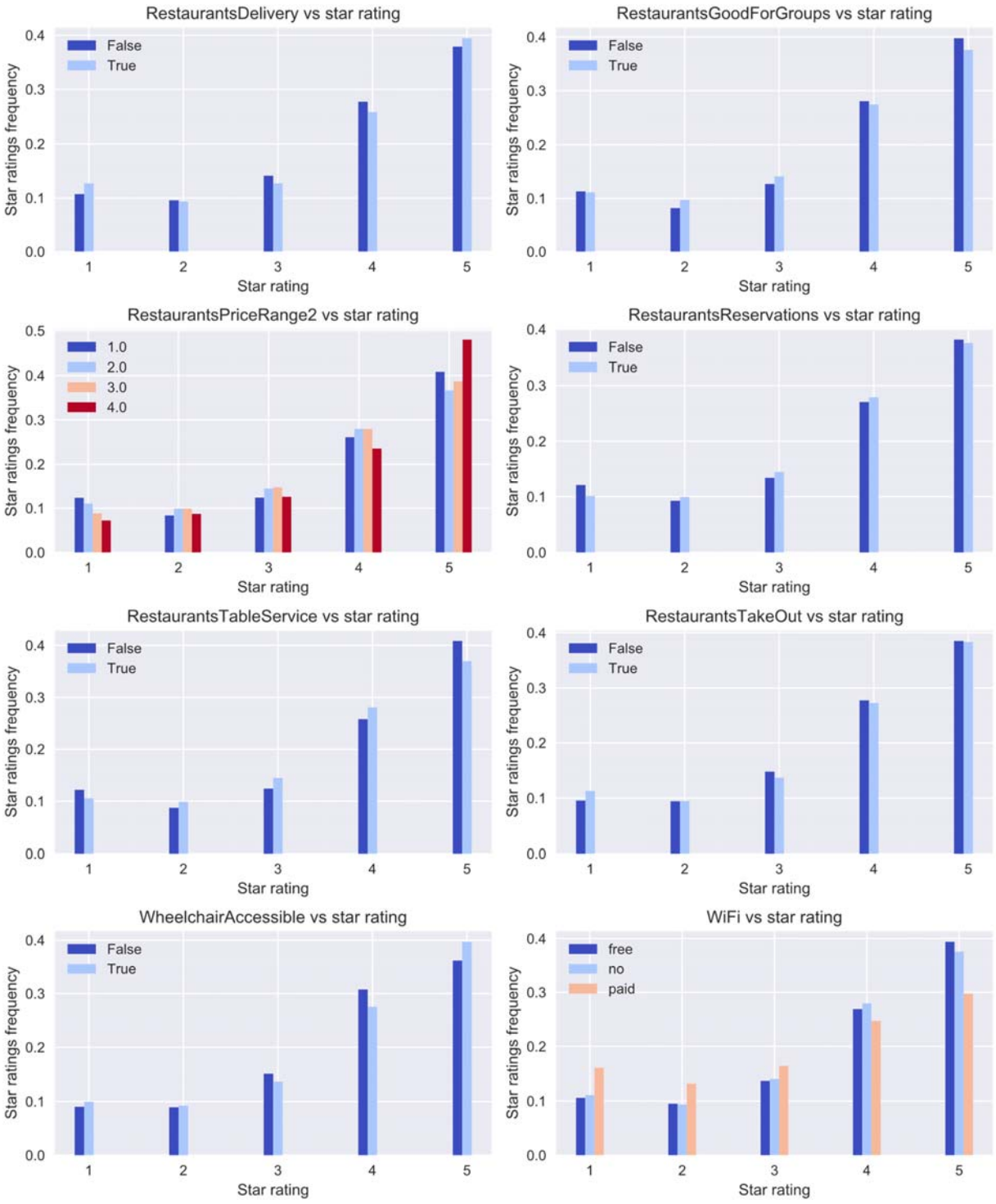

Figure 12: Restaurant attributes vs star ratings (part 4)

For attributes that showed an asymmetric distribution (one category has higher frequency in higher ratings and the other category is the opposite) of its categories, they could be used as features for predicting star ratings.

The reviews all had a time stamp which made time series visualization possible. The time series plot showed the relationship between year and average star ratings (figure 13).

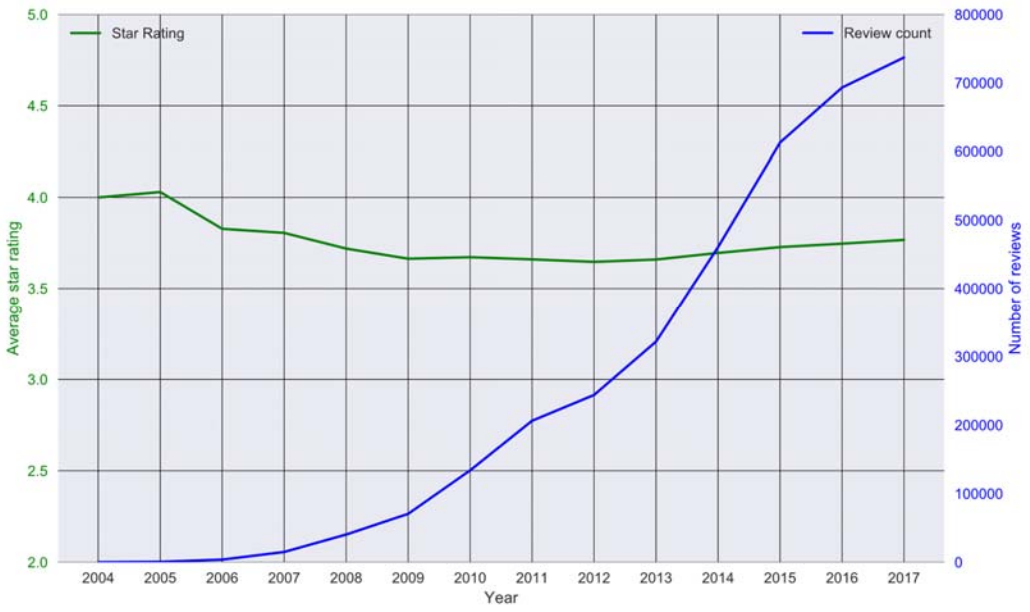

Figure 13: Star ratings plotted by year

The star ratings did go down initially and then slowly climbed up. Older comments tended to have better ratings. However, monthly plots showed no changes at all (figure 14).

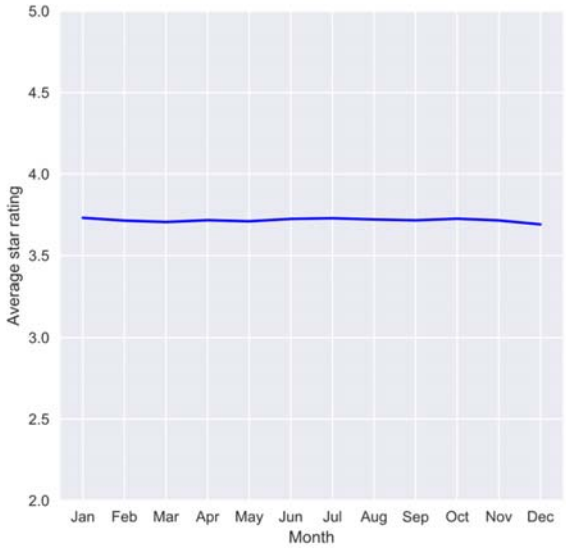

Figure 14: Monthly average star ratings

Do certain cuisines always get good ratings or bad ratings? To explore this potential feature, average rating frequencies were plotted side by side with each top 8 cuisines rating frequencies in figure 15.

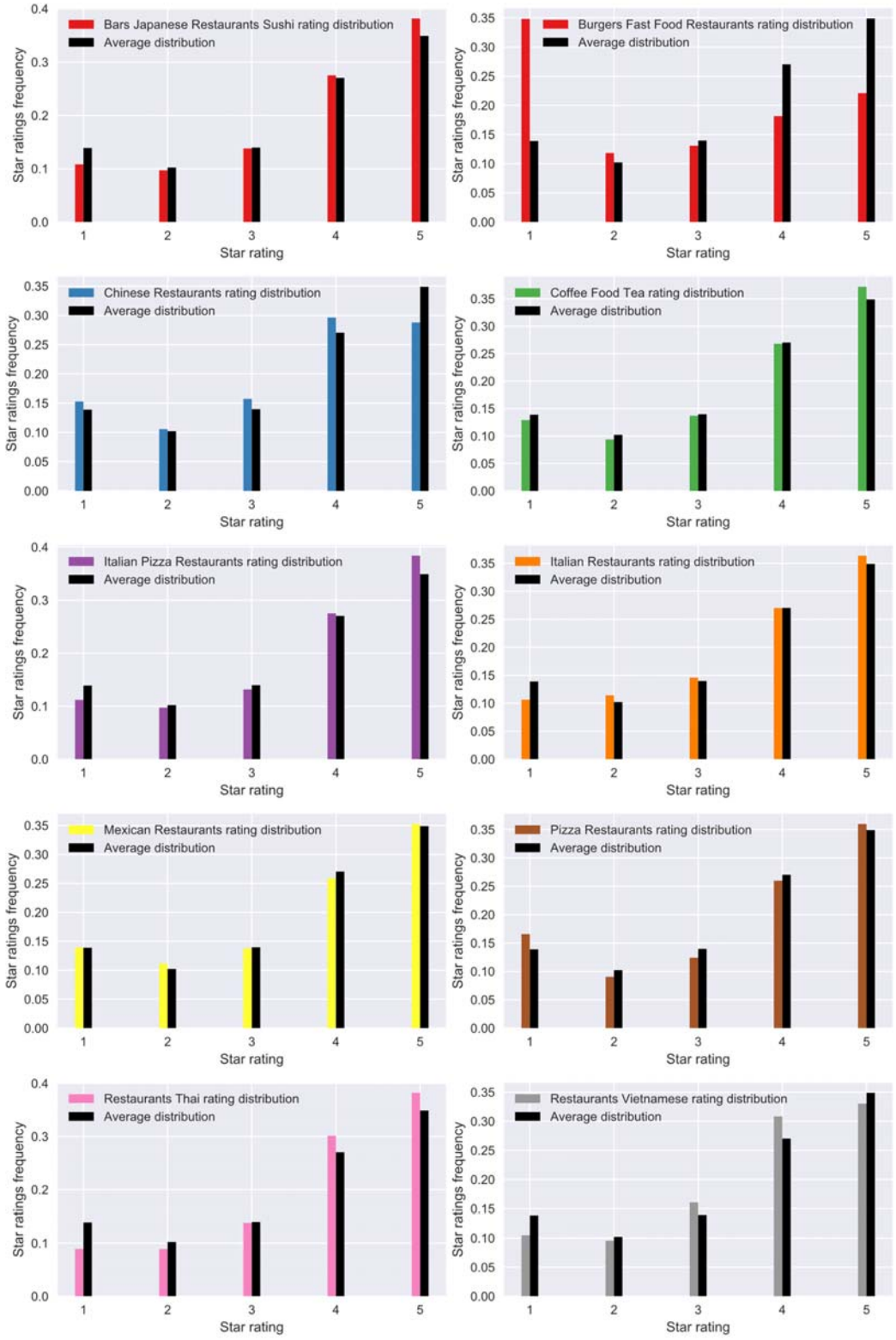

Figure 15: Compare each cuisine rating frequencies with average

This revealed some interesting patterns. For example, Burger fast food restaurants showed substantially higher low ratings compared to average while Sushi or Italian Pizza had more good ratings than average. This could serve as a feature when predicting star ratings.

Would the hours of the restaurants also affect their ratings? Did longer operating hours contribute to better ratings? To look this relationship, each restaurant's operating hours were extracted and their closing times were recalculated to reflect how late they closed. Then different plots were made to explore patterns. First, weekday hours were plotted as histogram to see how long the majority of restaurants were open during weekdays (figure 16).

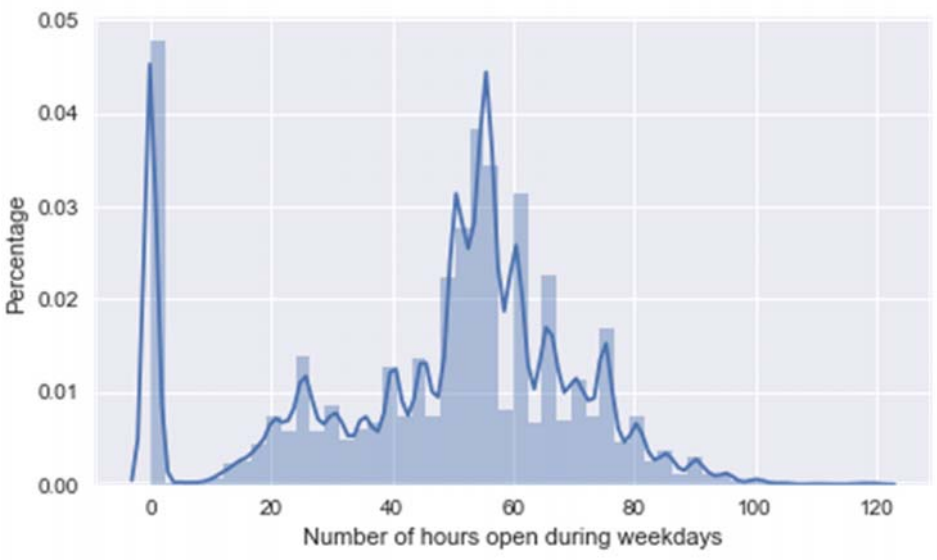

Figure 16: Weekly open hours of all restaurants

Most restaurants were open for 50 hours during weekdays (the 0 peak indicated restaurants that were closed). Was there a relationship between the number of hours during weekdays and star ratings? (figure 17).

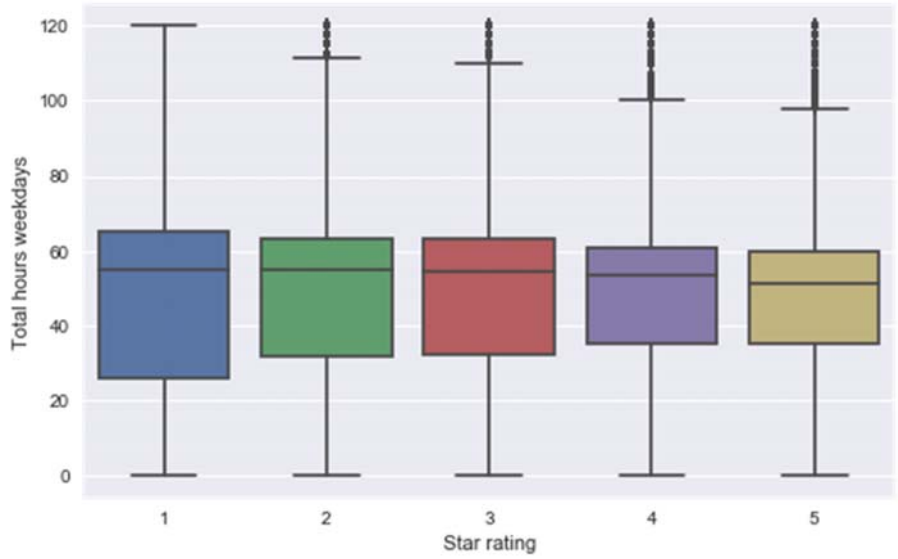

Figure 17: Star rating distribution by restaurants open hours during weekdays

There was no apparent relationship.

The same plot was used to plot against weekend hours (figure 18).

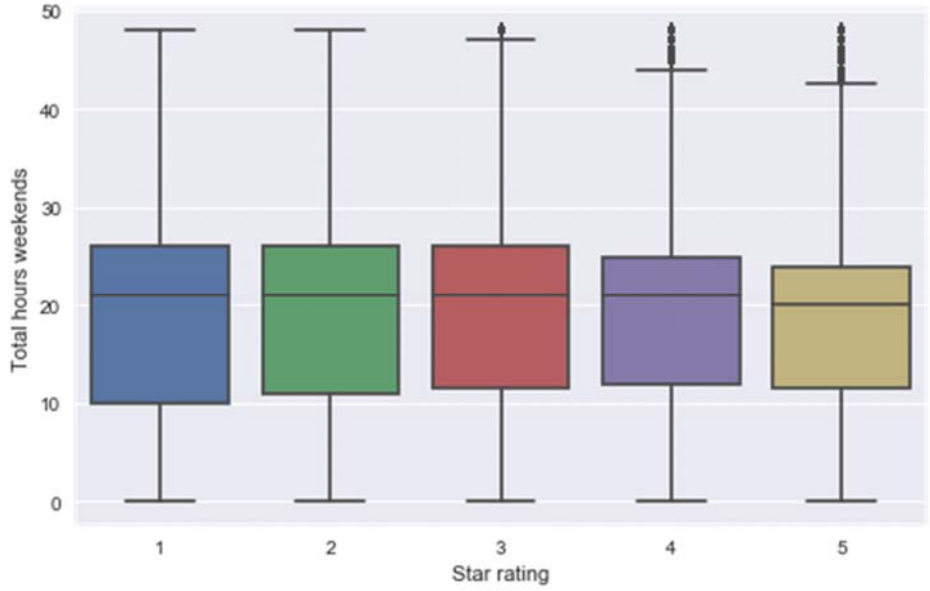

Figure 18: Star rating distribution by restaurants open hours on weekends

Again, there didn't seem to be a clear relationship.

Then the closing times were also plotted to see if they would have any effect on star ratings. Monday's data were used first (figure 19).

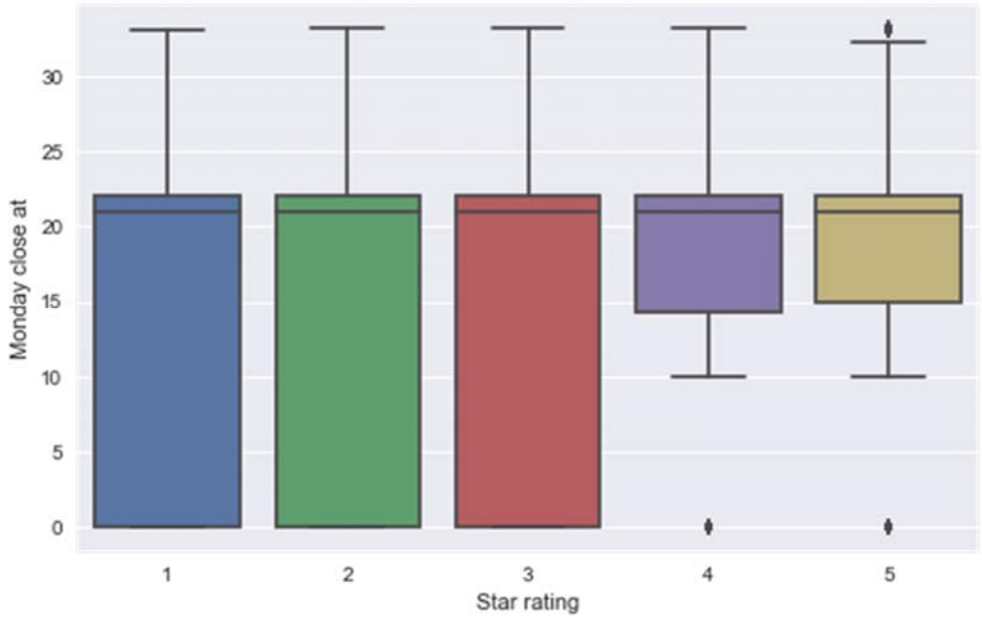

Figure 19: Star rating distribution by restaurants closing times

Now clearly that for 4 and 5-star ratings, they were mostly given to restaurants that closed late whereas for 1, 2, and 3-star ratings, they had a lot more restaurants that closed early.

Similar patterns were observed for other days as well (figure 20).

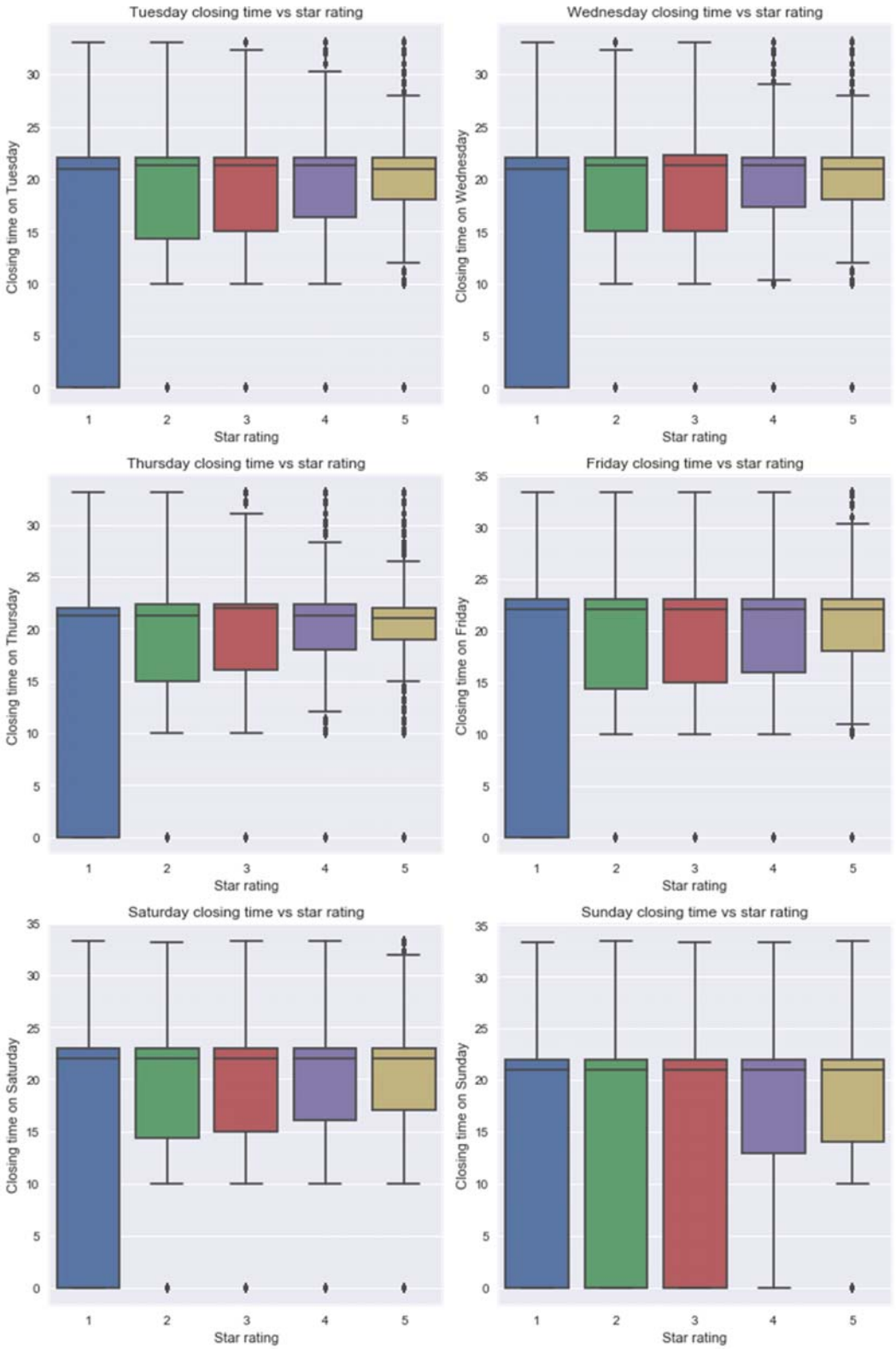

Figure 20: Star rating distribution by restaurants closing times

So, the closing times were useful when determining star ratings.

Now the features to use for building a machine learning model to predict star ratings had been selected. First step was to transform the review text data into a tf-idf matrix. The tf-idf matrix was used to normalize the appearances of each word by adjusting its weight based on its frequency in each document and then inversing the frequency in the entire corpus. Another step I took in the processing of the text was to do stemming. Stemming is to reduce various forms of the same word to its root form (e.g. 'like' and 'liked' mean the same thing but if there is no stemming they would produce two tokens instead of 1). This step would greatly reduce duplicate words and redundancy in the features extracted. This tf-idf matrix was the text feature matrix. This matrix was later combined with the categorical features (restaurants attributes) and numeric features (year of comments and closing times of restaurants) as explored above.

The first prediction was made on the 5 classes (star ratings from 1 to 5). The following classifier were tested:

Logistic Regression

Linear SVD

Random Forest

Adaboost Classifier

For the 5 class models, the train and test scores were plotted and compared directly with each other (figure 21).

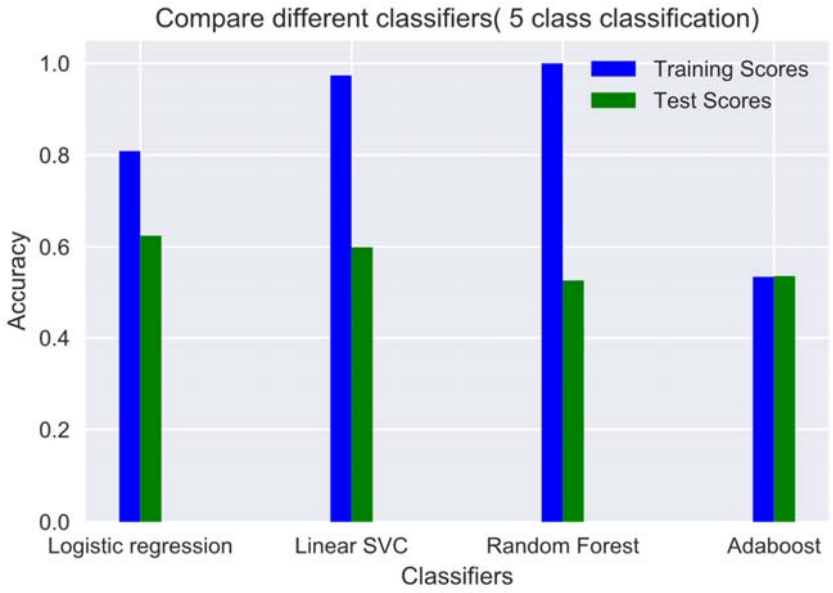

Figure 21: Compare 4 different classifiers on the 5-class rating prediction

Logistic Regression gave the highest test score of the all the classifiers tested. Even though 0.63 was not a very great score but considering for 5 classes, random guess would only give 0.2 accuracy. I then printed the confusion matrix and classification report to see which classes underperformed (figure 22).

```
array([[ 5242,  1492,   284,    68,    86],
       [ 1624,  2956,  1483,   219,   101],
       [  563,  1653,  4718,  2021,   419],
       [  227,   462,  2946, 10057,  5034],
       [  242,   235,   739,  5708, 19598]], dtype=int64)
```

	precision	recall	f1-score	support
1 star	0.66	0.73	0.70	7172
2 stars	0.43	0.46	0.45	6383
3 stars	0.46	0.50	0.48	9374
4 stars	0.56	0.54	0.55	18726
5 stars	0.78	0.74	0.76	26522
avg / total	0.63	0.62	0.63	68177

Figure 22: Confusion matrix and classification report for 5 class predictions

As shown by the confusion matrix and classification report, classes 2 and 3 were the most difficult to predict. This could be due the vague messages people wrote about those not so good and not so bad restaurants. Multiple negations could appear in the review text. Four-star rating prediction was most often predicted as 5 stars and vice versa. This prompted me to consider combining classes and try the predictions gain. A few of the important features were shown below using Random Forest classifier (figure 23).

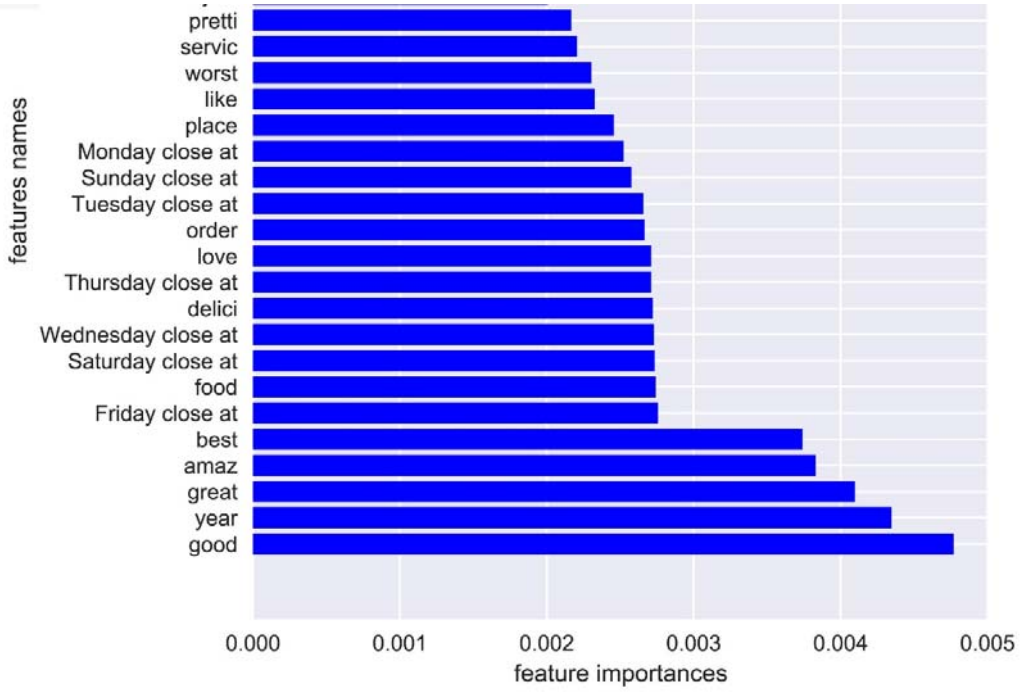

Figure 23: Feature importances using Random Forest.

The feature year and most closing times were shown as top features in the feature importances.

I then tried grouping classes into bad (1 and 2 stars) and good (4 and 5 stars) while discarding the 3 stars completely. I applied the same classifiers again and plotted the training and test scores side by side to compare (figure 24).

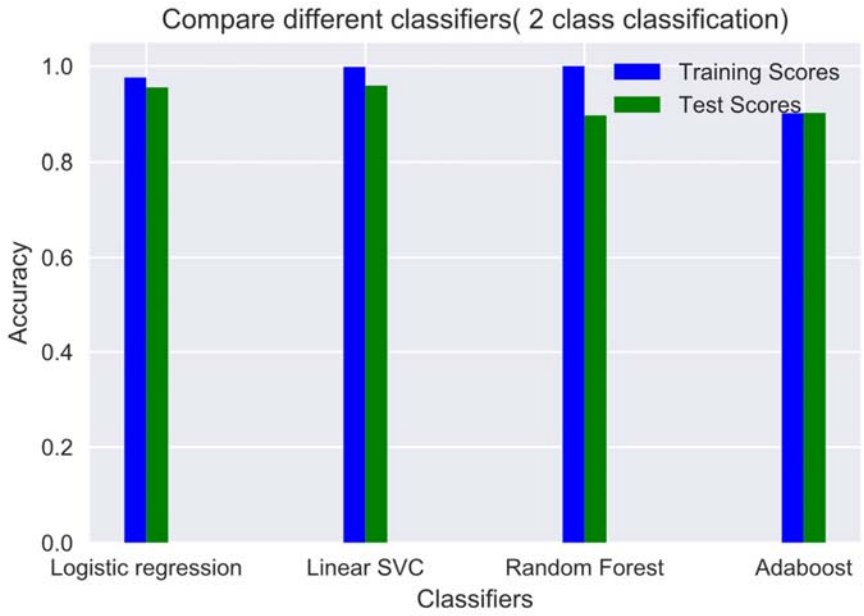

Figure 24: Compare 4 different classifiers on the 2-class rating prediction

This time, I got exceptional scores for all the classifiers. Linear SVC performed marginally better than logistic regression. The confusion matrix and classification report also showed good precision and recall for both classes (figure 25).

```
array([[12062,  1493],
       [  916, 44332]], dtype=int64)
```

	precision	recall	f1-score	support
Bad restaurant	0.93	0.89	0.91	13555
Good restaurant	0.97	0.98	0.97	45248
avg / total	0.96	0.96	0.96	58803

Figure 25: Confusion matrix and classification report for 5 class predictions

For the binary classification, I got 0.96 average f1 score. The f1 scores for both classes were very high implying a good model.

I then picked the logistic regression model for the 5 class prediction and linear SVC for the 2 class (binary) prediction and performed cross validation using GridSearchCV. The cross validated scores were shown below (figure 26).

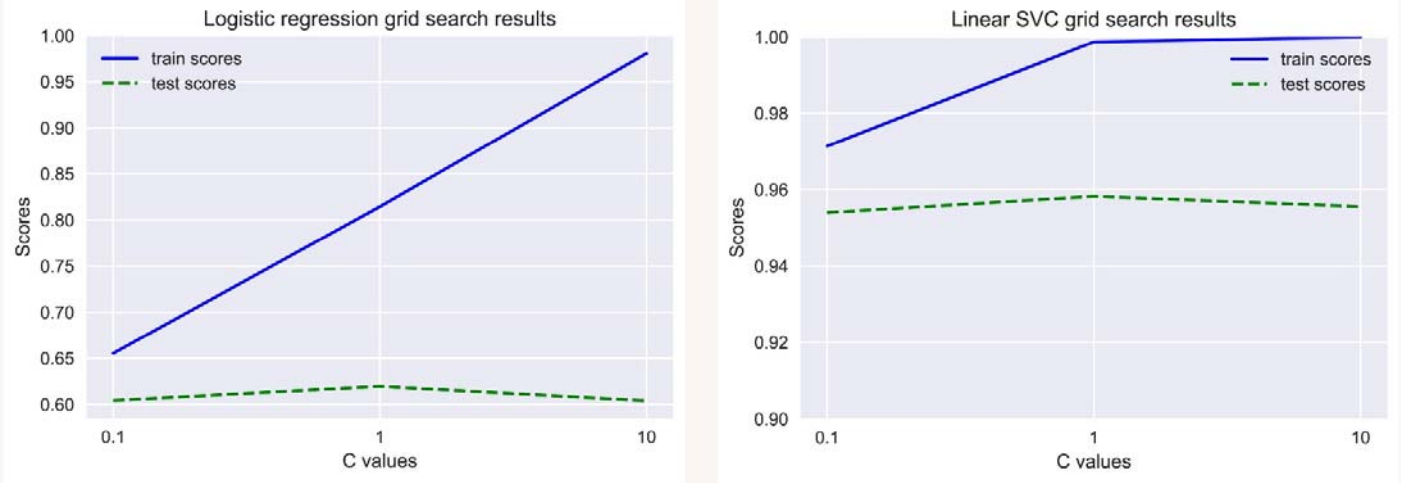

Figure 26: Grid search/cross validation results of logistic regression (5 class) and linear SVC (2 class)

4. **Clustering of review texts to find potential topics**

This idea was inspired by the thought that maybe there is something latent/intrinsic to good restaurants which could be discovered by clustering all the review texts for restaurants that had received either a 4-star or 5-star rating.

The approach was to take the entire column of review text and tokenize using tfidf with a stemming step just like the one used for the star prediction task. I also tried another preprocessing method called part of speech tagging (POS tagging). POS identifies tokens in the text as either noun, verb, adjective, or adverb (there are other tags, but I didn't use those for this project). This further reduced the number of tokens (tags such as pronouns and prepositions were discarded, and they usually don't carry much information). A few of the stemmed words are shown below (figure 27).

```
'aaa',
'aaron',
'ab',
'aback',
'abalon',
'abandon',
'abbi',
'abc',
'abend',
'aber']
```

Figure 27: Stemmed tokens.

To do the clustering, I used two methods: KMeans and Non-negative Matrix Factorization. KMeans sets the number of centroids first and then minimizes the distances of each sample to the centroid. Non-negative Matrix Factorization (NMF) factorizes the original token matrix into two matrices and finds the latent factors in the clusters. For NMF to work, all the entries in the matrix must be non-negative. The token matrix satisfied this requirement.

After applying KMeans with 5 clusters to initialize and also NMF with 5 components, I got the trained models. The components or centers then could be accessed by these models' methods (for KMeans, the method is cluster_centers_ and for NMF, the method is components_). I compared the two preprocessing methods (stemming vs stemming and pos tagging) by listing the top 10 words associated with the 5 centers. For stemming only, the top 10 words were shown below (figure 28).

```
Top 10 words associated with cluster  1  are:

['good', 'order', 'veri', 'food', 'chicken', 'place', 'like', 'realli', 'time', 'tri']
--------------------------------------------------------------------------------
Top 10 words associated with cluster  2  are:

['great', 'food', 'servic', 'place', 'good', 'veri', 'friend', 'alway', 'love', 'staff']
--------------------------------------------------------------------------------
Top 10 words associated with cluster  3  are:

['pizza', 'crust', 'good', 'great', 'place', 'order', 'best', 'love', 'veri', 'slice']
--------------------------------------------------------------------------------
Top 10 words associated with cluster  4  are:

['place', 'love', 'best', 'good', 'food', 'great', 'coffe', 'time', 'like', 'tri']
--------------------------------------------------------------------------------
Top 10 words associated with cluster  5  are:

['burger', 'fri', 'good', 'great', 'place', 'order', 'food', 'veri', 'chees', 'best']
```

Figure 28: Top 10 words associated with each cluster using stemming of tokens

This gave some good clusters and I summarized as below:

- **Good chicken food**
- **Great service, good staff**
- **Great pizza place**
- **Best coffee shop**
- **Good cheese burger**

These were the 5 topics extracted from all the restaurants review texts.

I then retrieved the top 10 words using stemming and pos tagging (still KMeans). They were shown in the following figure (figure 29).

```
Top 10 words associated with cluster  1  are:

['goddess', 'option', 'verdict', 'foil', 'chicharon', 'pizzeria', 'libr', 'realist', 'til', 'trend']
--------------------------------------------------------------------------------------------------------
Top 10 words associated with cluster  2  are:

['grassi', 'foil', 'seri', 'pizzeria', 'goddess', 'verdict', 'freshen', 'alyssa', 'lorsqu', 'stabl']
--------------------------------------------------------------------------------------------------------
Top 10 words associated with cluster  3  are:

['pitt', 'crouton', 'goddess', 'grassi', 'pizzeria', 'option', 'bestellen', 'lorsqu', 'verdict', 'sleev']
--------------------------------------------------------------------------------------------------------
Top 10 words associated with cluster  4  are:

['pizzeria', 'lorsqu', 'bestellen', 'goddess', 'foil', 'grassi', 'cocoa', 'til', 'libr', 'trend']
--------------------------------------------------------------------------------------------------------
Top 10 words associated with cluster  5  are:

['burger', 'frequent', 'goddess', 'grassi', 'pizzeria', 'option', 'foil', 'verdict', 'cheeki', 'bestellen']
```

Figure 29: Top 10 words associated with each cluster using stemming of tokens and KMeans

This did not yield much useful information somehow. So, doing pos tagging actually made the clustering worse.

NMF clustering gave similar results to the KMeans shown above when applied to just the stemmed tfidf matrix. The top 10 words associated with the 5 components were shown below (figure 30).

```
Top 10 words associated with cluster  1  are:

['order', 'like', 'tri', 'time', 'just', 'place', 'chicken', 'did', 'delici', 'got']
--------------------------------------------------------------------------------------
Top 10 words associated with cluster  2  are:

['great', 'food', 'servic', 'place', 'love', 'alway', 'amaz', 'friend', 'staff', 'atmospher']
--------------------------------------------------------------------------------------
Top 10 words associated with cluster  3  are:

['pizza', 'crust', 'slice', 'wing', 'chees', 'best', 'order', 'pepperoni', 'sauc', 'salad']
--------------------------------------------------------------------------------------
Top 10 words associated with cluster  4  are:

['good', 'veri', 'food', 'nice', 'price', 'realli', 'friend', 'servic', 'staff', 'reason']
--------------------------------------------------------------------------------------
Top 10 words associated with cluster  5  are:

['burger', 'fri', 'chees', 'beer', 'onion', 'shake', 'bun', 'bacon', 'best', 'good']
```

Figure 30: Top 10 words associated with each cluster using stemming of tokens and NMF

By doing the clustering, I discovered some interesting topics for all the restaurants using the user review text. Four out of the five topics had something to do with a kind of food and one actually was associated with the service and the people working at the restaurant.

5. **Recommender systems**

The last part of this project was building recommender systems to recommend restaurants to new users or registered Yelp users. I took three different approaches: adjusted weighted star ratings based ranking, content based, and collaborative filtering. Each of these recommender systems would recommend top restaurants based on the criterion provided by the user by zip code.

New ranking based on adjusted weighted ratings:

I took two factors into account when calculating adjusted ratings: year the rating was made, and the number of reviews received by the restaurant. For the time adjustment, I gave a penalty of 0 to the most recent year ratings and then as the year went back in time, I subtracted a penalty from the star rating. The penalty was calculated as follows: (most recent year-rating year)/10. Here I assumed a linear attenuation because the most recent reviews would be the most informative and relevant (other models could be considered too). The older the review was, the less influence it should have on the actual rating of the restaurant. For example, for a restaurant with a 5-star rating given in 2010, the penalty would be (2017-2010)/10=0.7. Then the new adjusted rating would be 5-0.7=4.3 (most recent year was 2017 when the data were downloaded).

To factor the number of reviews received by each restaurant, I borrowed the weighted rating formula from IMDB as follows (figure 31):

$$\text{weighted rating} = \frac{V}{V+M} \cdot R + \frac{M}{V+M} \cdot C$$

Figure 31: Formula to calculate new ratings based on the number of reviews

Here:

V=number of reviews for the restaurant

M=minimum number of reviews required to be put on the recommendation list

R=average star rating of the restaurant

C=mean star rating of all restaurants

For the project, I set M, the minimum number of reviews required to be 50. After adjusting ratings using the two factors discussed above, I was able to get a new ranking of all the restaurants. To get recommendations, the user would have to specify the postal code (zip code) and find restaurants within that postal code. Figure 32 was generated using postal code 15222 (present in the data):

Recommended restaurants are:

	Restaurants	Total Reviews	Address	Postal Code	Star Rating
1	Gaucho Parrilla Argentina	1267	1601 Penn Ave	15222	4.4
2	DiAnoia's Eatery	219	2549 Penn Ave	15222	4.2
3	Smallman Galley	339	54 21st St	15222	4.2
4	täkō	806	214 6th St	15222	4.2
5	Bakersfield	457	940 Penn Ave	15222	4.1

Figure 32: Recommended top 5 restaurants based on the new ranking system

If the user supplied a postal code not present in the data, it was handled properly (figure 33):

Sorry, no restaurants were found in the postal code you specified

Figure 33: Message shown when the postal code was not present in the data

This was a rather simple recommender system and did not consider the similarities between restaurants. However, this system would handle new users quite nicely even though the new user hadn't made any reviews yet. Next, I would introduce the content-based recommendation.

Content based system:

For the content-based system, I took metadata associated with each restaurant such as its attributes and cuisine type. The idea was to represent each restaurant by its metadata as a vector. I then calculated cosine similarity between each pair of restaurants represented by vectors. Cosine similarity would tell us whether the two restaurants were similar or not. The higher the cosine similarity, the more similar they were. To further learn more about cosine similarity, refer to this article. Cosine similarity was calculated by this formula (from Wikipedia):

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Here A and B are vectors (each restaurants metadata were represented as a vector) and Ai, Bi are components of the two vectors.

To represent each restaurant as a vector, I first transformed the cuisine type into tfidf vectors. I then used scikit-learn LabelEncoder to transform other metadata categorical features into numbers. Finally, I combined the text matrix and the metadata matrix into a sparse matrix using scipy. The following showed the first three restaurant vector representations (figure 34):

Alcohol	Ambience casual	Ambience classy	Ambience hipster	Ambience intimate	Ambience romantic	Ambience touristy	Ambience trendy	Ambience upscale	BikeParking	BusinessAcceptsCreditCards	BusinessParking garage
1	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	1	1	1
2	2	2	2	2	2	2	2	2	2	1	0

Figure 34: Vector representation of restaurants metadata

Using content based system, the user would be able to find restaurants similar to the one they like or specify. Here (figure 35, 36, and 37) I would show a few examples of using the content based recommender system.

```
content_recommender('Kashmir', 71229, 5)
```

```
Recommended restaurants are:
```

	Name	Address	Cuisine
1	Valentin's Bistro	Marktplatz 5	Bistros Restaurants
2	nah und gut Alsadi	In den Ziegelwiesen 7	Food Grocery Shopping
3	China-Restaurant Jasmin	Eltinger Str. 5	Chinese Restaurants
4	Don Giovanni	Niederhofenstr. 64	Italian Pizza Restaurants
5	Restaurant Glemseck	Glemseck 1	German Restaurants

Figure 35: Recommended restaurants similar to ‘Kashmir’ in postal code 71299

For Papa John’s Pizza (figure 36):

```
content_recommender("Papa John's Pizza", 61820, 10)
```

```
Recommended restaurants are:
```

	Name	Address	Cuisine
1	Pizza Hut	411 E Green St	Chicken Italian Pizza Restaurants Wings
2	Wood N' Hog Barbecue	904 N 4th St, Ste B	Barbeque Chicken Restaurants Wings
3	Sliders	616 E Green St	Burgers Restaurants
4	Pancheros Mexican Grill	2009 S Neil St	Mexican Restaurants
5	Chopstix	202 E Green St, Ste 1	Chinese Restaurants
6	Drew's Pizza	508 E Green St	Pizza Restaurants
7	Layalina Mediterranean Grill	40 E Springfield Ave	Mediterranean Restaurants
8	Domino's Pizza	102 E Green St	Chicken Italian Pizza Restaurants Sandwiches W...
9	Prime Time Pizza	505 E University Ave	Pizza Restaurants
10	M Sushi & Grill	715 S Neil St, Ste A	Asian Bars Fusion Japanese Korean Restaurants ...

Figure 36: Recommended restaurants similar to ‘Papa John’s Pizza’ in postal code 61820

Restaurants recommended when Starbucks was entered (figure 37):

```
content_recommender('Starbucks', 44118, 10)
```

```
Recommended restaurants are:
```

	Name	Address	Cuisine
1	Phoenix Coffee	2287 Lee Rd	Coffee Food Tea
2	Zero Below	1844 Coventry Rd	Cream Desserts Food Frozen Ice Yogurt
3	Piccadilly Artisan Yogurt	1767 Coventry Rd	Cream Food Frozen Ice Yogurt
4	Phoenix Coffee Company	1854 Coventry Rd	Coffee Food Rooms Tea
5	Walgreens	3020 Mayfield Rd	Beauty Convenience Cosmetics Drugstores Food P...
6	KiwiSpoon Frozen Yogurt	1854 Coventry Rd, Ste C	Cream Food Frozen Ice Yogurt
7	Mitchell's Fine Chocolates	2285 Lee Rd	Candy Chocolatiers Food Shops Specialty Stores
8	The Sweet Fix Bakery	2307 Lee Rd	Bakeries Food
9	Bialy's Bagels	2267 Warrensville Ctr Rd	Bagels Food
10	Ben & Jerry's	20650 N Park Blvd	Cream Desserts Food Frozen Ice Yogurt

Figure 37: Recommended restaurants similar to ‘Starbucks’ in postal code 44118

The content based recommender system has more flexibility and better performance when compared to the ranking based system which only takes into account the rankings of the restaurants but not how similar they are.

Collaborative filtering (CF) recommender system:

The last recommender system I made was collaborative filtering system. This system usually uses a user/item/rating data frame (figure 38) and tries to find similarities either between item/item or user/item.

date	user_id	business_id	stars_x
2004-07-22	le_brG6cwrzvWdKEGqA7YA	uz7UbvVUwsg68Rok6kbqRg	5
2004-09-15	w_6miJytUt6z8oRkGjVG-A	9X-43jnj6-6ZBuBdFm7BLA	2
2004-10-12	sE3ge33huDcNJGW3V4obww	PD2MAlYYi9HCqPH7lBKwTg	5
2004-10-19	nkN_do3fJ9xekchVC-v68A	oYMsq2Xvzw6UbrllMWjb-A	4
2004-10-19	c6HT44PKCaXqzN_BdgKPCw	u8C8pRvaHXg3PgDrsUHJHQ	5

Figure 38: User/Item/Rating matrix format

This approach finds users that have rated the same items similarly and then calculates the predicted ratings they would give to items they have not rated yet. Then those predicted ratings are ranked from high to low. Items with top ratings would then be recommended to users.

The Surprise package for Python is very good at building effective Collaborative Filtering systems. It takes data format as shown above and then calculates item/item or user/item similarities as specified. I used Singular Value Decomposition algorithm (SVD) to train the train data set. The test data set contains all the restaurants that users have not rated and predicted ratings would be made for those restaurants for each user. The following table shows the top 5 restaurants recommended for the first 5 users with predicted ratings (figure 39).

Users	1st Restaurant	2nd Restaurant	3rd Restaurant	4th Restaurant	5th Restaurant
-5KiEoe-mb4sU2VwnVwrYw	(Yardbird Southern Table & Bar, 4.448143511632...	(Rollin Smoke Barbeque, 4.361078514298695)	(Jean Philippe Patisserie, 4.338873641980125)	(Yard House, 4.325951264322038)	(Hiroba Sushi, 4.30917748011664)
LPWSfa9mGf59P7KZg_eOdA	(Mon Ami Gabi, 4.560510815225531)	(Yard House, 4.3325320086542085)	(Yardbird Southern Table & Bar, 4.273418535906...	(The Combine Eatery, 4.237205892813327)	(Brew Tea Bar, 4.236163737902843)
EyLVCFOKItmIMg7XcRxU9Q	(Yard House, 4.420580330439966)	(Mon Ami Gabi, 4.390624184521719)	(Lola Coffee, 4.281053068868802)	(Brew Tea Bar, 4.278227307282902)	(Yardbird Southern Table & Bar, 4.275357509866...
-tBJZffckiQC6zhPIwVpVQ	(Yardbird Southern Table & Bar, 4.30498559007849)	(Yard House, 4.251247996344009)	(Mon Ami Gabi, 4.236412309252682)	(District One, 4.2217447790537825)	(Gelatology, 4.21387119440429)
TPa6ZGNRjH2Zfrwq4vsUPw	(Gordon Ramsay Steak, 4.239030581165461)	(Brew Tea Bar, 4.192484834780216)	(Mon Ami Gabi, 4.177138518433449)	(Yardbird Southern Table & Bar, 4.168899445418...	(Millie's Homemade Ice Cream, 4.147826665191982)

Figure 39: Recommendation of restaurants using Collaborative Filtering

6. Summary and future work

This work shown above used Yelp data set and performed machine learning/clustering of topics/recommender engines. Upon finishing this project, I have learned a great deal in data acquisition, cleaning, manipulation, machine learning model/evaluation and building effective CF recommender engines. And of course, there are limitations and more future work that could be done for this project as listed below:

Conclusions:

- The star rating prediction model did not generate a very good score when predicting the 5 classes. This could be due to: not enough data trained; class imbalance; feature selection; parameter tuning/cross validation.
- Due to time and resources limitation, the whole data set was not used in the prediction model but rather a subset of the data was used. This could contribute to the sub-optimal score obtained.
- There were many missing values for a lot of the categorical features which were impossible to impute from the data set and I had to drop those features. The loss of the information could contribute to the less optimal performance of the model.
- Typos and misspelled words in the review text could contribute to data inaccuracy when they were transformed by tfidf vectorizer to generate text features.
- For clustering, there were no effective way to evaluate the model rather than inspect by eye.

There is still more to work on this data set and I will bring up some ideas I had but did not have the time to accomplish for this project.

Future work:

- To use cloud computing to utilize all the data for the star rating predictions and see if that would improve the scores. Also, try more complex neural network/deep learning.
- Compare feature importances using Random Forest and Logistic Regression and check if they agree with each other.
- For the 5-class prediction, try down sampling of the over-represented classes to make all classes equal and see if that would increase the scores.
- For text features, try doing tri-grams to see if that would capture more trends such as negations in the text.
- For year of ratings, maybe don’t use scaling as when new data come in, the max of years will also change.

Acknowledgement

I’d like to thank Springboard and my career track mentor Danny Wells for your help along the way. The weekly Skype meeting and the community on Springboard website are great resources for me to learn and get insights. Thanks to Vishal Ratra for giving me valuable comments during our mock interviews. And always thank my family for their caring and understanding.