Springboard Data Science Career Track | Nikki Seegars
Capstone Project II: Yelp Restaurant Recommendation Engine

Problem Statement

In any given city there are many restaurants that locals and tourists can patronize. In order to streamline users choices, a good recommender engine for Yelp (an online crowd sourced review website) will help to keep and drive more users to its website. An user personalized collaborative recommendation engine for restaurant choices based on the historical data that has been gathered about each user's past star rating is a great solution for this need. The recommendations are based on the rating similarities to other users through collaborative filtering.

Dataset Overview

Yelp is a popular online business directory with crowd-sourced user reviews and ratings. Many types of business have yelp pages including restaurants, clothing stores, and personal services such as tax preparation. Currently, restaurants tend to have more user inputs than other types of businesses.Yelp has a subset of their user data available to the general public for challenges and other academic purposes. This data can be downloaded from their website and comprises data from 10 metropolitan areas across two countries. There are six json files with information about businesses, reviews, and users. See Table 1 below which is a summary of the contents in each json file. The common features amongst the files are business_id and user_id.

Json File	Record Count	Number of Features	Features
business	192,609	14	**business_id**, name, address, city, state, postal code, latitude, longitude, stars, review_count, is_open, attributes, categories, hours
checkin	161,950	2	**business_id**, date
photo	200,000	4	photo_id, **business_id**, caption, label
review	6,685,900	9	review_id, **user_id**, **business_id**, stars, date, text, useful, funny, cool
tip	1,223,094	5	text, date, compliment_count, **business_id**, **user_id**
user	1,637,130	22	**user_id**, name, review_count, yelping_since, friends, useful, funny, cool, fans, elite, average_stars, compliment_hot, compliment_more, compliment_profie, compliment_cute, compliment_list, compliment_note, compliment_plain, compliment_cool, compliment_funny, compliment_writer, compliment_photos

Table 1.

Data Collection & Wrangling

Each of the json files were loaded into DataFrames to review the features and to calculate basic descriptive statistics. In addition, the files were saved to .csv files in order to easily load into

SQL tables for data aggregation. Although the dataset is marketed as being a subset of 10 metropolitan areas, the review.json file has 36 different states included. For simplicity purposes, the top 5 states were used for the initial data exploration. The states are shown along with their review counts in Figure 1.

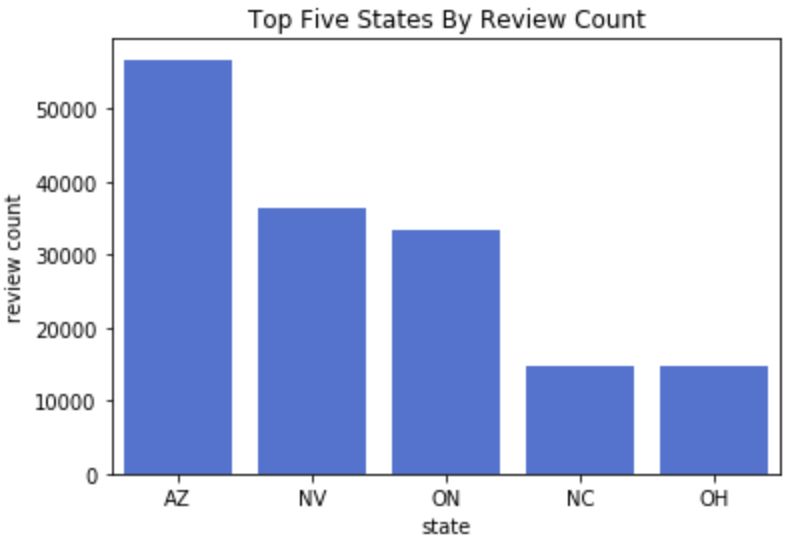


Figure 1.

Some of the businesses in the Yelp dataset have been identified as being closed, these businesses were not included in the recommender engine.

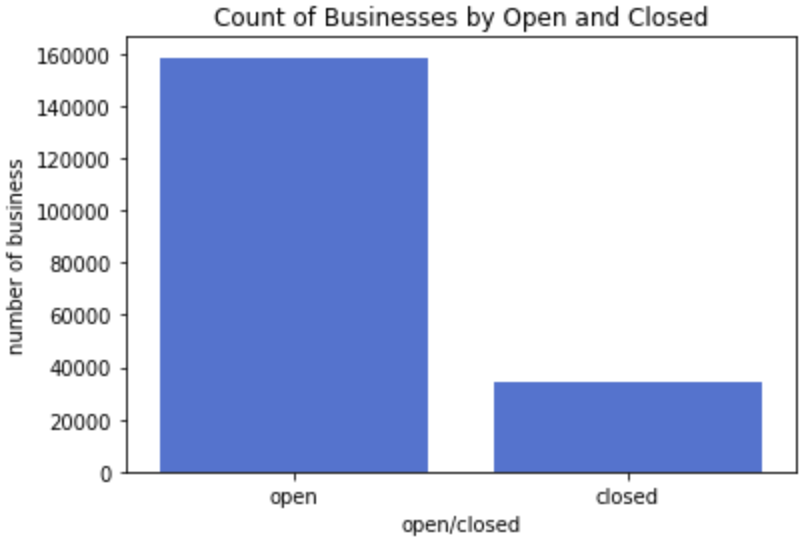


Figure 2.

In order to focus only on restaurants, the categories feature was used for identification and any records with categories beginning with select descriptors were assumed to be restaurants. There are 1,177 unique categories in the original dataset. Of those, 215 describe food and

beverage businesses. The review.json file has reviews, user id, and star ratings. The text in the reviews are very long so that column was left out of the DataFrame and .csv files since it is not needed for this analysis. The user.json file has attributes including number of reviews and any friends' user ids. The checkin.json, photo.json, and tip.json files do not contain any necessary information for this project, so those features were not incorporated in the model.

The .csv file data was imported into PostgreSQL tables and the following view was created of a query to combine all of the pertinent features.

```
SELECT
r.date,
r.review_id,
r.user_id,
r.stars AS user_stars,
b.categories,
r.business_id,
b.name AS business_name,
b.review_count,
b.stars AS business_stars,
b.address,
b.city,
b.state_,
b.postal_code,
b.latitude,
b.longitude,
b.hours,
b.is_open,
f.friend_count,
rank() OVER (PARTITION BY r.business_id ORDER BY r.business_id, r.date) AS order_rank
FROM yelp_review_data r
JOIN yelp_business_data b ON r.business_id = b.business_id
JOIN yelp_user_friend_count f ON r.user_id = f.user_id
WHERE b.is_open = 1
AND b.state_ IN ('AZ', 'NV', 'ON', 'NC', 'OH')
AND b.review_count > 4
AND b.categories SIMILAR TO '(Acai Bowls|Afghan|African|American|Arabian|Argentine|Armenian|Asian
Fusion|Australian|Austrian|Bagels|Bakeries|Bangladeshi|Bar Crawl|Barbeque|Bars|Basque|Beer|Belgian|Beverage
Store|Bistros|Brasseries|Brazilian|Breakfast & Brunch|Brewpubs|British|Bubble
Tea|Buffets|Burgers|Burmese|Butcher|Cafes|Cafeteria|Cajun/Creole|Cambodian|Canadian (New)|Candy
Stores|Cantonese|Caribbean|Caterers|Cheese|Chicken|Chinese|Chocolatiers & Shops|Cideries|Cocktail Bars|Coffee|Colombian|Comfort
Food|Convenience Stores|Conveyor Belt Sushi|Cooking|Creperies|Cuban|Cupcakes|Custom
Cakes|Czech|Delicatessen|Delis|Desserts|Dim Sum|Diners|Dinner Theater|Dive Bars|Do-It-Yourself
Food|Dominican|Donairs|Donuts|Eatertainment|Egyptian|Empanadas|Ethical Grocery|Ethiopian|Ethnic Food|Ethnic
Grocery|Falafel|Farmers Market|Fast Food|Filipino|Fish & Chips|Fondue|Food|French|Fruits &
Veggies|Gastropubs|Gelato|German|Gluten-Free|Greek|Grocery|Guamanian|Hakka|Halal|Hawaiian|Health Markets|Herbs &
Spices|Himalayan/Nepalese|Honduran Hong Kong Style Cafe|Hot Dogs|Hot Pot|Hungarian|Ice Cream & Frozen Yogurt|Imported
Food|Indian|Indonesian|International|Internet Cafes|Irish|Irish Pub|Italian|Izakaya|Japanese|Juice Bars & Smoothies|Kebab|Kids
Activities|Korean|Kosher|Laotian|Latin American|Lebanese|Live/Raw Food|Local Flavor|Macarons|Malaysian|Meat
Shops|Mediterranean|Mexican|Middle Eastern|Modern European|Mongolian|Moroccan|New Mexican Cuisine|Noodles|Organic
Stores|Pakistani|Pan Asian|Pasta Shops|Patisserie/Cake Shop|Persian/Iranian|Personal Chefs|Peruvian|Piano
Bars|Pizza|Poke|Polish|Popcorn Shops|Pop-Up Restaurants|Portuguese|Poutineries|Pretzels|Public Markets|Pubs|Puerto
Rican|Ramen|Restaurants|Russian|Salad|Salvadoran|Sandwiches|Scottish|Seafood|Seafood Markets|Shanghainese|Shaved Ice|Shaved
Snow|Singaporean|Slovakian|Smokehouse|Soba|Soul Food|Soup|South African|Southern|Spanish|Speakeasies|Specialty Food|Sports
Bars|Sri Lankan|Steakhouses|Street Vendors|Sushi Bars|Swiss Food|Szechuan|Tacos|Taiwanese|Tapas|Tea Rooms|Teppanyaki
Tex-Mex|Thai|Themed Cafes|Turkish|Tuscan|Ukrainian|Vegan|Vegetarian|Venezuelan|Vietnamese|Waffles|Whiskey Bars|Wine|Wraps )%';
```

The PostgreSQL query was imported into python by creating a function that used the pd.read_sql to retrieve the filtered data.

```
# Set up a connection to the postgres server.
conn_string = "host="+ creds.PGHOST +" port="+ "5432" +" dbname="+ creds.PGDATABASE +" user=" + creds.PGUSER \
+" password="+ creds.PGPASSWORD
conn=psycopg2.connect(conn_string)

# Create a cursor object
cursor = conn.cursor()

def load_data(schema, table):

    sql_command = "SELECT * FROM {}.{};".format(str(schema), str(table))
    print (sql_command)

    # Load the data
    data = pd.read_sql(sql_command, conn)

    print('data shape', data.shape)
    return (data)
psql_data =   load_data('public', 'yelp_business_review_subset0')
```

Each of the five metropolitan geographic sizes was approximated with the haversine formula which calculates the great-circle distance (shortest) between two points on Earth using their latitude and longitude. Python has a haversine library that calculates the spherical difference given the latitude and longitude of two points.

```
from haversine import haversine
haversine((start latitude, start longitude), (end latitude, end longitude), unit='mi') #unit in miles
```

The five metropolitan areas of interest are Metro Phoenix, Las Vegas Valley, Greater Toronto, Charlotte Metropolitan, and Greater Cleveland. The great-circle distances calculated for each using the minimum and maximum latitude and longitude values are 69.61, 45.15, 69.42, 59.76, and 226.23 miles respectively. Greater Cleveland had an outlier with the haversine well greater than 70 miles. Upon closer inspection of the data it was determined that one restaurant incorrectly listed the state as “OH” instead of “ON”. After the changes were made to the DataFrame, Greater Cleveland’s longest distance went down to 78.08 miles and Greater Toronto remained the same.

Metro Area	Distance (miles) Before Adjustment	Distance (miles) After Adjustment
Metro Phoenix	69.61	69.61
Las Vegas Valley	45.15	45.15
Greater Toronto	69.42	69.42
Charlotte Metropolitan	59.76	59.76
Greater Cleveland	226.23	78.08

Table 2.

Each of the 215 restaurant categories had a column created using one hot encoding with a 1 indicating that the category was included and 0 indicating it is not for each record.

```
for category in category_list:
    psql_data[category] = np.where(psql_data['categories'].str.contains(category), 1, 0
```

Exploratory Data Analysis

Grouping by state, a summary of aggregate descriptive statistics gives a picture of how each metropolitan area compares to one another.

	business_stars					review_count				
	count	min	max	mean	median	count	min	max	mean	median
state_										
AZ	1012812	1.0	5.0	3.801264	4.0	1012812	5	2556	388.067710	258
NC	235738	1.0	5.0	3.732587	4.0	235738	5	1572	241.455451	144
NV	1118221	1.0	5.0	3.823314	4.0	1118221	5	8348	994.148511	506
OH	197360	1.0	5.0	3.742301	4.0	197360	5	1074	161.299787	93
ON	489709	1.0	5.0	3.600254	3.5	489709	5	2121	170.063403	97

Table 3.

Yelp allows users to give a review and rate each business between 1 and 5 stars, with 1 being the least favorable and 5 being the most favorable. Each metropolitan area's mean average business star rating is fairly similar, between 3.6 and 3.8 with Toronto having a bit of lower average star rating. For this project, restaurants with 5 or more reviews were included in the dataset. The median number of reviews by restaurant by metropolitan area range from 93 to 506.

Most of the star ratings fall between 3 and 4 stars as shown in the histograms in Figure 3 with Greater Toronto being the only metropolitan area showing more 3 star ratings than 4 star ratings.

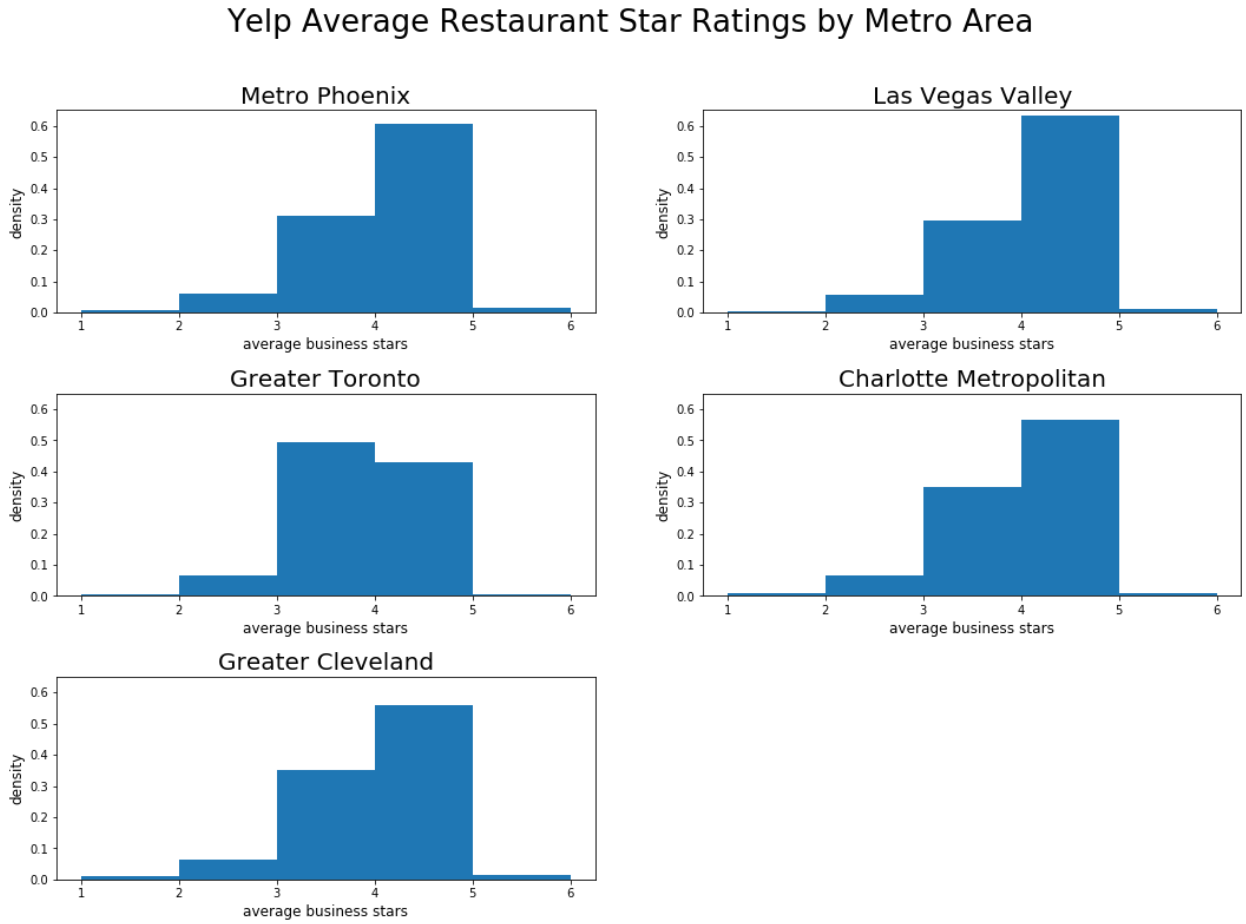


Figure 3.

The top 10 restaurant categories are similar for each of the metropolitan areas although Greater Toronto shows higher counts categorized as Asian inspired (Japanese, Chinese, and Sushi Bars).

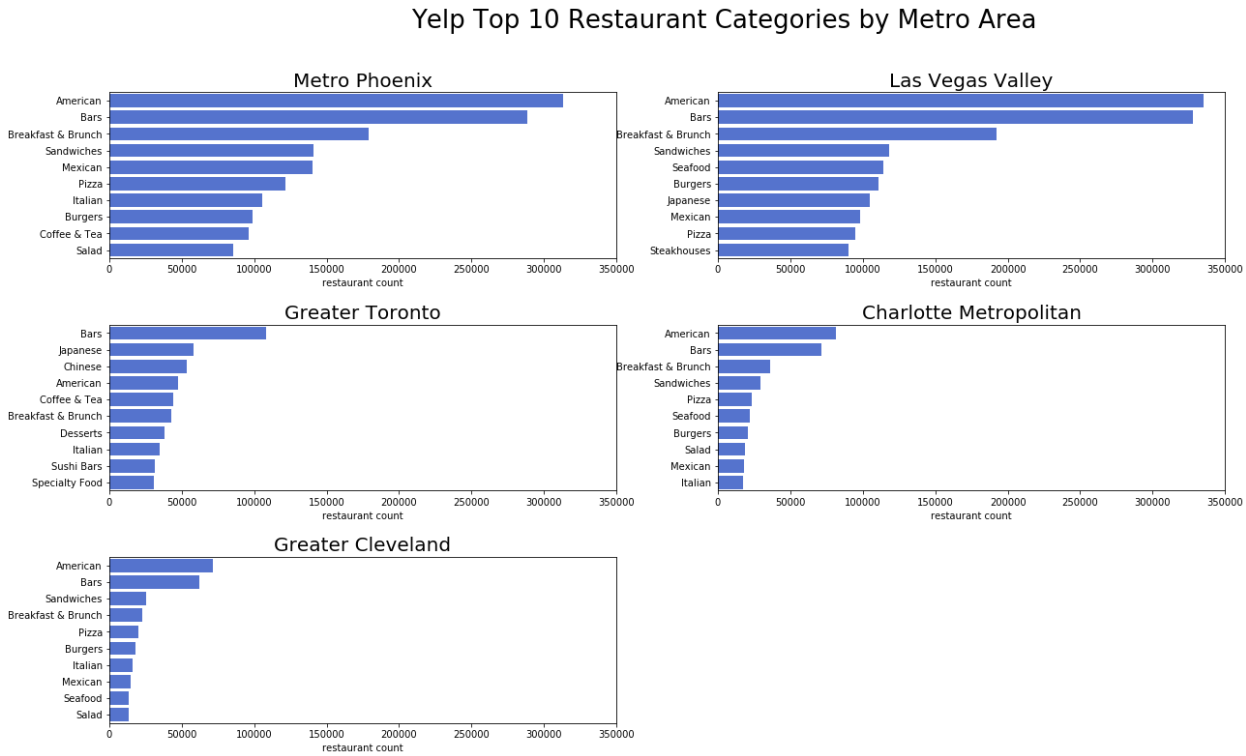


Figure 4.

Initially, on average, the star ratings start off relatively high and modestly decrease as the rating counts go up. However, after approximately having 5-15 reviews the star rating starts to increase. This would indicate the more reviews, the more stable the star ratings become.

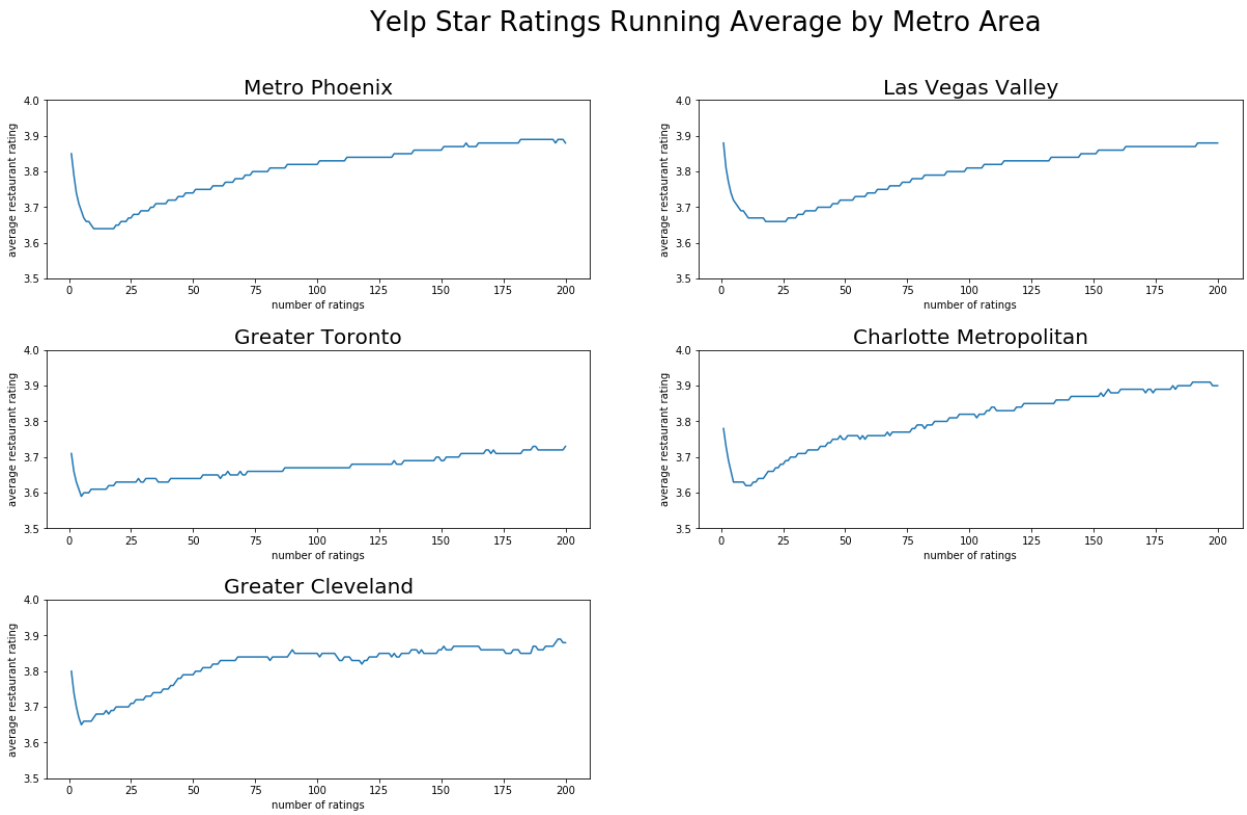


Figure 5. **NOTE**: the y scale does NOT begin at 0, but instead begins at 3.5.

Machine Learning

Supervised machine learning techniques were used to create the recommender engine consisting of labeled data (star ratings) that were split into a train set and test set. A sample set of the restaurant review data was used to develop several machine learning models for comparison of which performed better using precision and recall metrics. For efficiency purposes, data from only four categories of restaurants were used with those categories being 'Italian', 'Japanese', 'Mexican' and 'Burgers'. These restaurant categories were a few of the more popular categories. All other category columns in the DataFrame were dropped. Additionally, for practicality, only those restaurants in the metropolitan Phoenix reviewed by a randomly chosen 25,000 reviewers were trained and tested.

```
# Select one metropolitan area --> Metro Phoenix & chose randomly sample of 25,000 users
area = 'AZ'
users = np.random.choice(psql_data['user_id'][psql_data['state_']==area].drop_duplicates(), size=25000, replace=False)
subset_data = psql_data[(psql_data['Category_Count'] > 0) & (psql_data['user_id'].isin(users)) & (psql_data['state_']==area)]
len(subset_data)
```

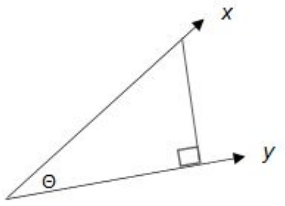
Four different collaborative filtering machine learning algorithms were evaluated with precision and recall metrics. Collaborative filtering is a method of making predictions by comparing and

assessing the preferences of users to one another. The basic premise of collaborative filtering stems from the notion that if two users have similar taste on one item, the likelihood that they will have similar taste on another item is higher than a randomly selected user. As shown below in Table 4, using the data from other users, the recommender engine predicts the star ratings for the users with missing values (**?**) . These predictions are based upon the similarity or commonality with other users and not from generic averages.

User	**Restaurant 1**	**Restaurant 2**	**Restaurant 3**	**Restaurant 4**
A	4	2	3	3
B	5	**?**	4	3
C	**?**	2	2	**?**

Table 4.

For the custom weighted average algorithm the cosine similarity score was used as weights in order to compare users based on the number of stars given by the users to each restaurant. Cosine similarity is a measure of similar two items. These items are mathematically represented as two nonzero vectors of an inner product that measures the cosine of the angle between them.



cosine similarity = $\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$ where x and y represent vectors of the user restaurant ratings. The resultants from the cosine similarity calculation ranges from 0 to 1, with numbers closer to 1 being more similar. Table 5 lists some sample data from the Yelp dataset followed by an example of the calculation for cosine similarity.

	Star Ratings	
Restaurants	**User A**	**User B**
Joe’s Farm Grill	5	5
Fiesta Burrito	5	
The Original Blue Adobe Grill	5	5
Brio Tuscan Grill	5	
McDonald’s	5	
Oregano’s Pizza Bistro	1	
Serrano’s Mexican Restaurant	5	
Rubio’s	5	
Cafe Rio	2	
SeÃ±or Taco	3	
Filiberto's Mexican Food	2	
Baja Joe's Mexican Cantina	3	
Pizzeria Bianco	4	
Pomo Pizzeria - Phoenix	5	
Spotted Donkey Cantina, el Pedregal		4

Table 5.

$$\text{cosine similarity} = \frac{(5\ x\ 5\ )+(5\ x\ 5\ )}{\sqrt{5^2+5^2+5^2+5^2+5^2+1^2+5^2+5^2+2^2+3^2+2^2+3^2+4^2+5^2}\ \sqrt{5^2+5^2+4^2}} = 0.39482$$

All the NaN values were replaced with zeros in the DataFrame containing the restaurant star ratings to prevent an error in the code. In addition, the restaurants that were not rated by users were dropped with the weighted average prediction calculation of the dot product of a matrix of user-restaurant reviews and the user-user similarity matrix. The numpy dot product function was used to calculate the weighted average.

$$\text{weighted average (prediction)} = \frac{\sum_{u=1}^{n} (ratings\ for\ movie, r_u)\ x\ (cosine\ similarity\ for\ user, u)}{\sum_{u=1}^{n} cosine\ similarities, u}$$

The remaining three of the tested algorithms used the surprise package in scikit for the restaurant recommender engine. The surprise package offers several built-in algorithms with measurement functions. The BaselineOnly algorithm makes a prediction by adding the mean of star ratings and any bias from the user and/or restaurant. If the user is unknown, then only the mean of all ratings is used for the prediction.

$\hat{r} = \mu + b_u + b_i$
$\hat{r}$, prediction rating
μ, mean of all ratings
b_u, bias of user
b_i, bias of item (restaurant)

Surprise has a built in matrix factorization using the widely popular SVD, singular value decomposition, algorithm. SVD allows for highly dimensional and complex data to be reduced to a lower dimensional space to help find better features for classifying data. In surprise, this algorithm adds on a dot product calculation to the BaselineOnly calculation accounting for the factors in an user matrix and a transposed restaurant matrix.

$\hat{r} = b_{ui} = \mu + b_u + b_i + q_i^T p_u$
q_i^T, transposed item factors
p_u, user factors

The final algorithm that was explored was one based on k nearest neighbors (KNN). KNN is a non-parametric algorithm that makes a classification decision based on its proximity to known data.

The surprise KNNBasic algorithm = $\hat{r_{ui}} = \frac{\sum_{v \varepsilon N_i^k(u)} sim(u,v) \cdot r_{vi}}{\sum_{v \varepsilon N_i^k(u)} sim(u,v)}$

$\hat{r_{ui}}$, predicted rating for user u and item i
$v \varepsilon N_i^k(u)$, the set in user v for the k nearest neighbors of user u for item i
$sim(u,v)$, similarity of u & v
r_{vi}, known rating for user v and item i

The weighted (across all star ratings, 1 to 5) precision and recall results for each of the algorithms tested are listed below in Table 6.

Algorithm	**Precision (weighted average)**	**Recall (weighted average)**
Custom Weighted Average	0.34	0.30
Surprise BaselineOnly	0.44	0.24

Surprise Matrix Factorization SVD	0.55	0.25
Surprise KNNBasic	0.38	0.27

Table 6.

Precision is also known as the positive predictive value and calculated as $\frac{True\ Positives}{True\ Positives + False\ Positives}$.

Recall is also known as sensitivity and calculated as $\frac{True\ Positives}{True\ Positives + False\ Negatives}$.

Surprise Matrix Factorization SVD was chosen as the algorithm to use for this recommender engine since its precision score for was higher than all the rest of the algorithms and having the 2^{nd} highest recall score.

Parameter Tuning

Several of the parameters in the Surprise Matrix Factorization SVD were tuned in order to improve the model's predictability. Surprise offers a streamlined GridSearchCV function that iterates through values for identified parameters and calculates the RMSE (root mean square error) and/or MAE (mean absolute error) scores. Both these errors are good measures for the difference between two continuous variables. However, with a classification problem precision and recall tend to give a better indication of how accurately each class was predicted. Unfortunately, surprise does not have a built in function for precision and recall. The precision and recall function from scikit learn was used. A for loop was created to iterate through different values for several of the parameters listed in Table 7.

Parameter	**Description**	**Tested Values**	**Chosen Value**
n_factors	number of factors used in matrix	3, 12, 50, 100	100
n_epochs	number of iteration of the stochastic gradient descent procedure	5, 10, 20, 40	10
init_mean	mean of the normal distribution for factor vectors initialization	0, 0.05, 0.1, 0.15	0
init_std_dev	standard deviation of the normal distribution for factor vectors initialization	0, 0.05, 0.1, 0.15	0.1
lr_all	learning rate for all parameters	0.001, 0.003, 0.005, 0.007	0.007
reg_all	regularization term for all parameters	0.01, 0.02, 0.03, 0.04	0.04

Table 7.

The values that maximize the precision value were ultimately used for the recommender engine resulting in a precision of 0.56 and recall of 0.24.

By assuming restaurant star ratings of 4 or 5 as those being highly rated, the algorithm was adjusted to have binary labels with 1 identifying a highly rated restaurant and 0 as identifying a restaurant that is not highly rated. A precision recall curve was created with these binary values to show the various precision and recall values at different thresholds ranging from 1-5 with the threshold being the value at which a predicted rating would be classified as highly rated and thus recommended to the user.

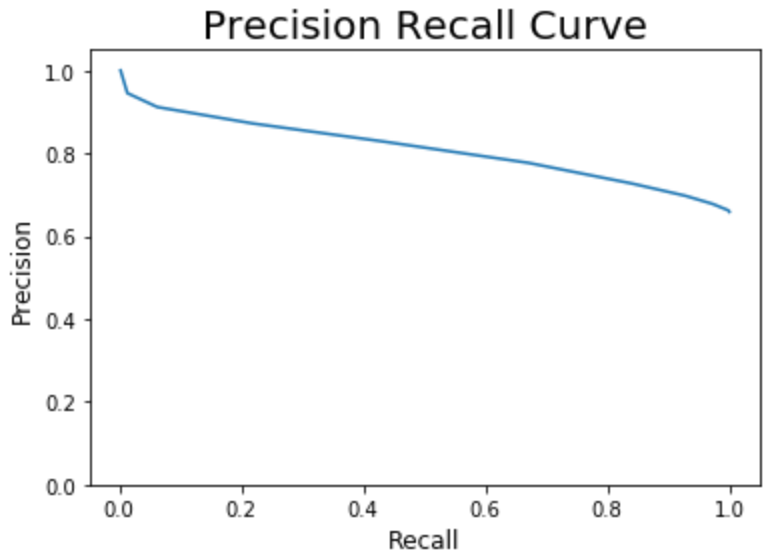


Figure 6.

The area under the curve is calculated using scikit learn's auc function and the result is 0.80798 indicating an above average result. The binary classification with a threshold of 3.5 (all predictions 3.5 or higher are considered as recommendations) has a weighted precision of 0.67 and weighted recall of 0.69.

Finally, restaurant recommendations from the training set for each user is given based on having a star rating prediction of 1 for recommended. Also, included is the actual and predicted star ratings as shown in the example below.

user_id	business_id	business_name	predicted_value	true_value	predicted_rating	true_rating
iDlkZO2iILS8Jwfdy7DP9A	oMBNvB6tHlwW3UwGBYqljw	Blue Fin	1	1	4.521318	5.0
iDlkZO2iILS8Jwfdy7DP9A	cTZmf7B-4yciMc1WKiCVOA	Welcome Diner	1	1	4.303440	5.0
iDlkZO2iILS8Jwfdy7DP9A	DaVTuhzi6EgWStb2eAjNjA	Presidio Cocina Mexicana	1	1	4.170000	5.0
iDlkZO2iILS8Jwfdy7DP9A	Tw3miGKZHtmxmaQZIYFRrA	Federal Pizza	1	1	4.157582	5.0
iDlkZO2iILS8Jwfdy7DP9A	LtNgP4FqXp5nMFOHErK8cw	Yen Sushi & Sake Bar	1	1	4.039577	4.0
iDlkZO2iILS8Jwfdy7DP9A	qUPUCcBbn-ugXFSItXLmGw	Akai Hana Sushi & Grill	1	1	4.001893	4.0
iDlkZO2iILS8Jwfdy7DP9A	wa8QgXQu1ZxwPgdRI9IYIg	Tampopo Ramen	1	0	4.001597	3.0
iDlkZO2iILS8Jwfdy7DP9A	CUivTcULsu5MJIYYNVm1zw	Hana Japanese Eatery	1	1	3.992266	4.0
iDlkZO2iILS8Jwfdy7DP9A	eS29S_06lvsDW04wVrIVxg	Barrio CafÃ©	1	0	3.958837	3.0
iDlkZO2iILS8Jwfdy7DP9A	89uU51kOiQXbJHVA3C6XMQ	The Original Carolina's Mexican Food	1	1	3.949200	4.0

Table 8.

Conclusion

The Yelp restaurant recommender engine for the categories of Italian, Japanese, Mexican and Burgers can be scaled to more or less categories. However, with a very sparse user-restaurant matrix (~99.9%), the precision and recall of the predictions are better than simply guessing and for the recommended set (prediction=1) the precision is 0.72 and the recall is 0.86. These values indicate a recommender engine that is fairly good at predicting whether or not a user will like a restaurant.