Restaurant Recommendation System in Toronto

Prerit Mishra

Coursera Data Science capstone project

Abstract

Recommender systems are used to personalize your experience on the web, telling you what to buy, where to eat or even who you should be friends with. People's tastes vary but generally follow patterns. People tend to like things that are similar to other things they like, and they tend to have similar taste as other people they are close with. Recommender systems try to capture these patterns to help predict what else you might like. E-commerce, social media, video and online news platforms have been actively deploying their own recommender systems to help their customers to choose products more efficiently, which serves win-win strategy. In this project, we build a recommender system to provide restaurant recommendation in the Toronto neighborhood based on their likings and past visits.

Keywords: recommendation, restaurants, neighborhood

Restaurant Recommendation System in Toronto

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you.

Problem Description

Companies like Netflix, YouTube, Amazon etc., leverage recommendation systems to help users discover new and relevant items (movies, videos, music, products), creating a delightful user experience while driving incremental revenue. Most recommender systems are typically classified into two broad categories-

1. Contend Based
2. Collaborative filtering

However, complex algorithms can implement both approaches. Content-based methods leverage similarity of item attributes while collaborative based approach make used of similarity among users/interactions. Here we implement a recommender system for restaurants using user-based collaborative filtering approach. Food application companies such as Hungrygowhere, Zomato, and Deliveroo can leverage such system to provide explicit range of restaurants based on customer's previous choices and interests.

Data Description

Here I use Toronto neighborhood data with geolocation information, which was scraped from Wikipedia in the last week's assignment. Only neighborhood areas with Boroughs containing 'Toronto' in it are used.

For all these neighborhoods, Restaurants within the radius of 5kms were extracted using Foursquare API. Let's look at what each column represents:

Ratings data:

- *userId* - the ID of the user who rated the restaurant.
- *Restaurant_id* - the ID of the restaurant.
- *rating* - The rating the user gave the restaurant, between 1 and 5.
- *timestamp* - The time the restaurant was rated.

Restaurants data:

- *Id*- the ID of the restaurant.
- *Venue* – Restaurant name
- *Neighborhood* – the neighborhood where the restaurant is located
- *Venue Longitude* – Longitude data of the restaurant
- *Venue Latitude* – Longitude data of the restaurant
- *Category* – Restaurant category

Data Cleaning

A subset of 20 restaurants and 10, 000 user ratings is used for reducing complexity. In order to create a robust and accurate model, the whole dataset would make a better choice.

Data filtering:

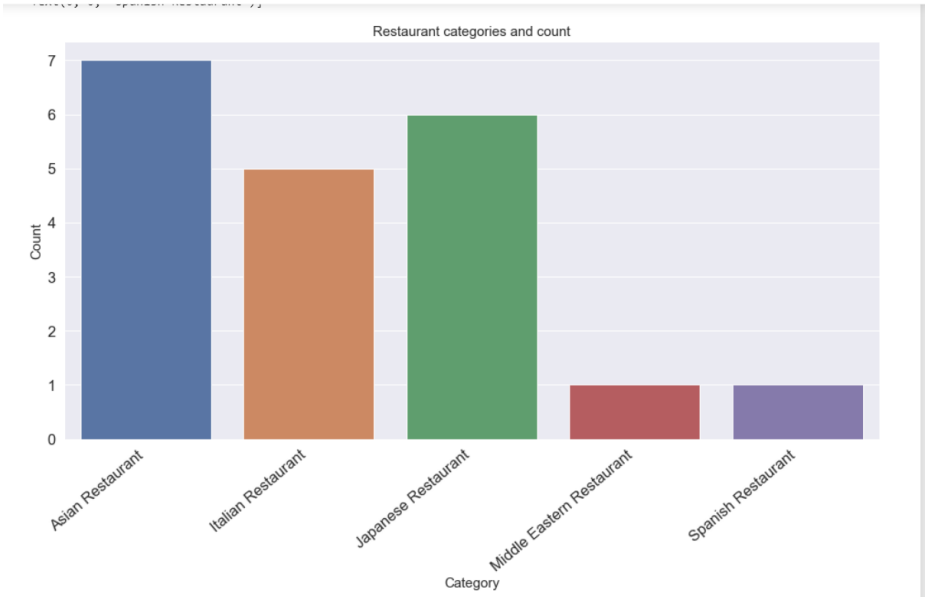
1. Null *userId* and *Restaurant_id* are removed.
2. Null values in Restaurant *Category column* are removed.
3. String cleanup in Restaurant names.

Data Exploration

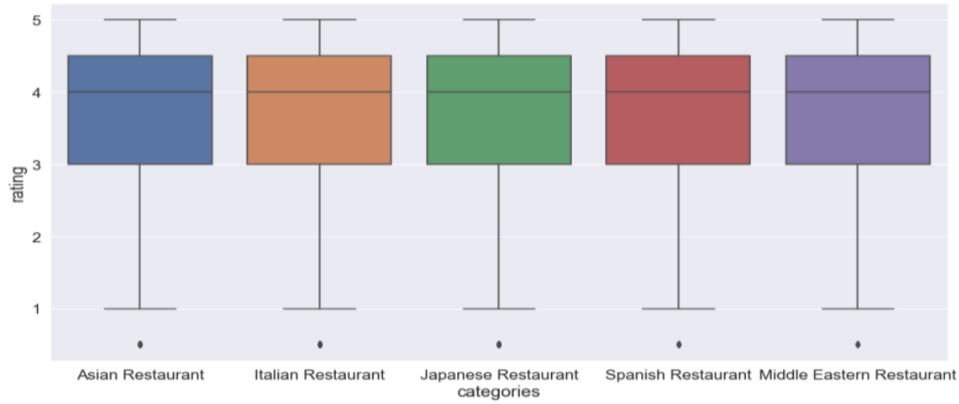
Ratings and Restaurant data is joined. Sample below:

	userId	rating	timestamp	restaurant_id	Venue
0	64	3.5	1434644734	118e9b40-ea07-408e-9b70-f66583bc149f	Studio Restaurant
1	471	5.0	1500417760	c81ed71a-b5de-467d-aabf-f8a12fcee86c	Site Of Great Canary Restaurant
2	301	1.5	1484675858	ea5054cb-b4a0-4cc0-a058-ffde30aa21eb	Rol San Restaurant
3	125	4.0	1023423520	9009f48a-3838-405c-a693-8fd18d42c211	Ryan Restaurant
4	230	3.0	864120692	3b9e13e4-a023-46d4-87e9-179a190f9318	The Hot House Restaurant Bar

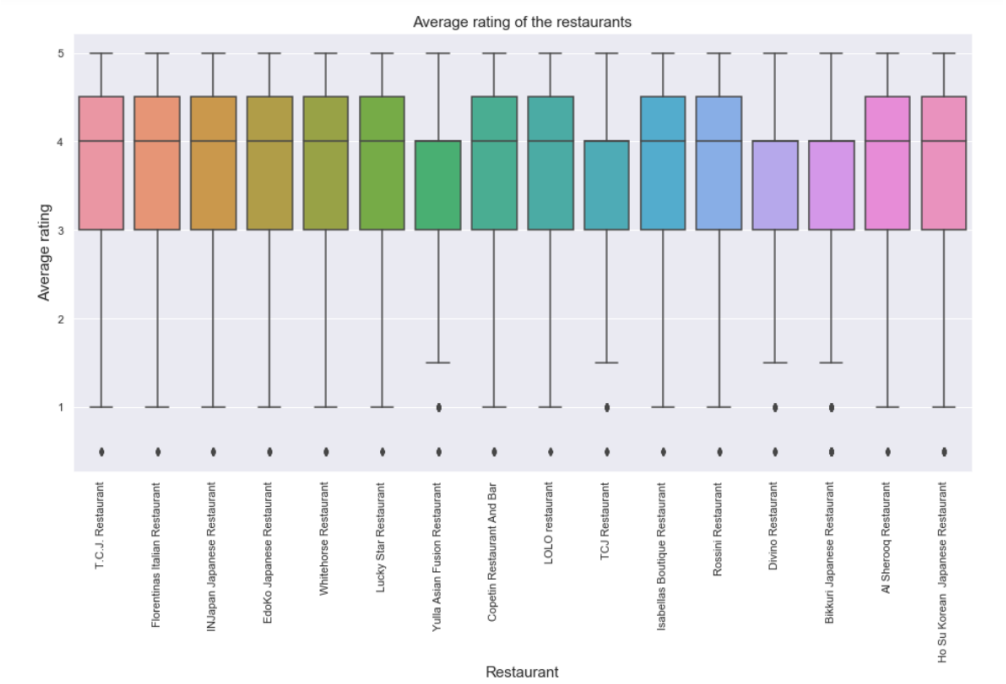
Categories and number of restaurants in each category :



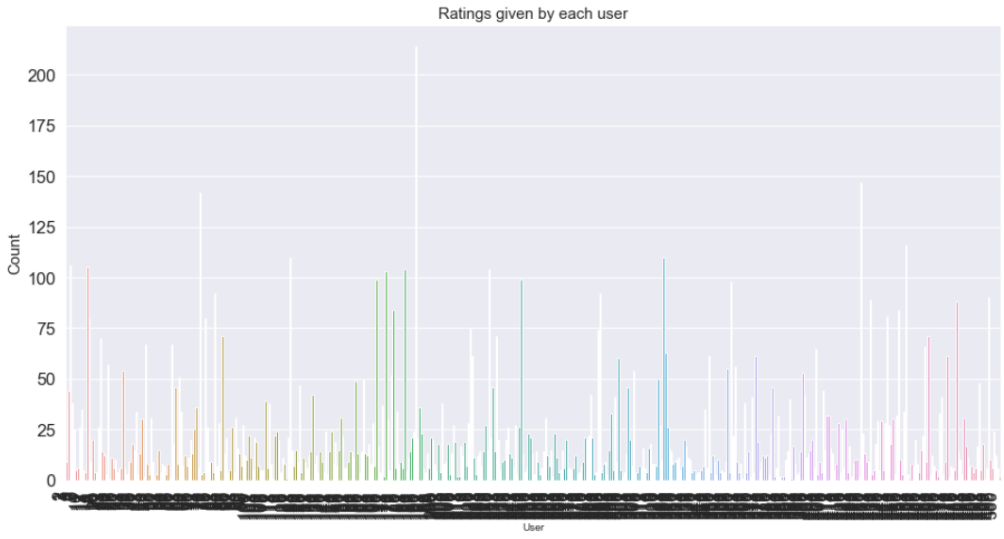
Average rating of all the restaurants by category:



Average rating of all the restaurants by each restaurant.



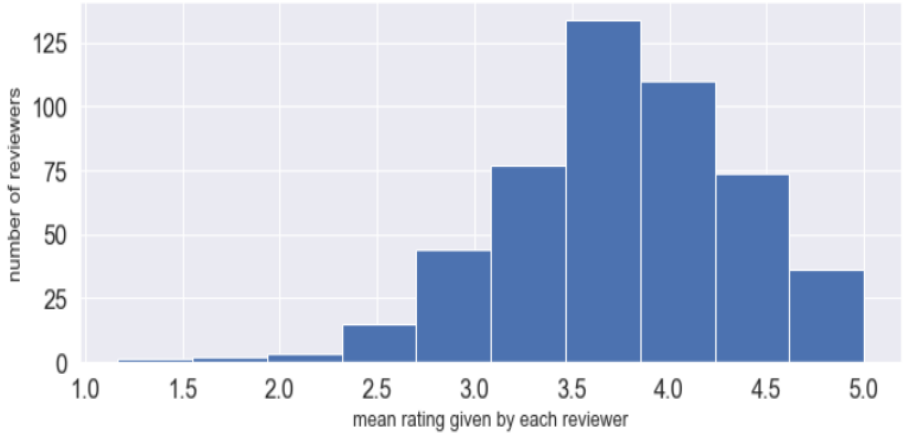
Rating given by all users.



Some statistics:

Number of unique users	497
Number of unique venues	20
Avg Number of ratings per reviewer	16

Mean rating distribution given by each user:



Results

Firstly, we will have to predict the rating that a user will give to restaurant. In user-based CF, we will find say k=3 users who are most similar to user 3. Commonly used similarity measures are cosine, Pearson, Euclidean etc. We will use cosine similarity here which is defined as below:

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

In sklearn, NearestNeighbors method can be used to search for k nearest neighbors based on cosine similarity metrics. For example, top 4 similar users for user 1 are:

```
1: User 472, with similarity of 0.9476609356270141
2: User 162, with similarity of 0.7700769315417864
3: User 83, with similarity of 0.6798734977619258
4: User 135, with similarity of 0.6767587118081178
```

Discussion

This method also called Memory-based algorithms are easy to implement and produce reasonable prediction quality. The drawback of memory-based CF is that it doesn't scale to real-world scenarios and doesn't address the well-known cold-start problem, i.e. when a new user or new item enters the system. Model-based CF methods are scalable and can deal with higher sparsity level than memory-based models, but also suffer when new users or items that don't have any ratings enter the system.Here, we only used 20 restaurants and 500 user-rating to reduce the computation of the model. One can use spark framework to run this model for a large dataset to build a robust and accurate model.