* Web Scraping in Python using BeautifulSoup. Restaurants (rest\_name, locality, cost\_for\_two, rating, votes, cuisines) [zomato\_restaurants.csv]. User (cust\_name, cust\_rating, rest\_name) [zomato\_reviews.csv].

**DATA CLEANING**

* Remove restaurants without ratings.
* Arrange restaurants in decrease order of their ratings.
* Strip white spaces in restaurant names and user name.

**TAKE A QUIZ**

* Find the average rating.
* Find the min\_votes required for the restaurant to be in the chart.
* Using the min\_votes, get the restaurants that qualify for qualified\_restaurants.
* Calculate the ‘score’ for each restaurant using the formula: (v/(v+m) \* R) + (m/(m+v) \* C)

where v – number of votes

m – min\_votes calculated in the previous step

R – rating of each restaurant

C – average rating

* Arrange the restaurants in decreasing order of score.
* Input (get from the thumbnail) the selected cuisines.
* Match the restaurant containing the cuisines.
* Select top 7 restaurants from the recommendations [zomato\_qualified\_restaurants.csv].

**CONTENT-BASED RECOMMENDATION**

* **TF-IDF** (term frequency-inverse document frequency) is a numerical statistic that is intended to reflect how important a word is to a document. This value increases proportionally to the number of times a word appears in a document and is offset by the number of documents that contain the word, which helps to adjust for the fact that some words appear more frequently in general.
* **Cosine Similarity** is used to denote the similarity between two restaurants. Calculating the dot product of TF-IDF Vectorizer will give the cosine similarity score.
* **Function to perform recommendations:**

1. Get the index of the restaurant that matches the restaurant name inputted.
2. Get the list of cosine similarity scores for that particular restaurant with all restaurants. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
3. Sort the restaurants based on the similarity score.
4. Get the restaurant indices based on the similarity score.
5. Get the restaurant name based on the restaurant indices.

* Finally get the list of restaurants for which recommendations is not possible and store it in [content\_based\_non\_rec.csv].
* While recommending to a user:

1. Go further only if the username entered is present in [user\_ratings.csv].
2. Get the number of restaurants that the user has rated.
3. Iterate through the restaurants that the user has rated.
4. Stop if the restaurant is present in [content\_based\_non\_rec.csv].
5. Else stop if the restaurant has less than 3.0 ratings (because it’s not useful to recommend similar restaurants to a restaurant that has gotten less rating from the user).
6. Else generate a list of recommendations for each restaurant that the user has rated.

**COLLABORATIVE-FILTERING RECOMMENDATION**

* **Item-based filtering –** They identify similar items based on how people have rated it in the past.
* **Pearson coefficient –** It is a measure of the linear correlation (how close two variables are) between two variables.
* **Steps –**

1. Pivot table with cust\_name , rest\_name with cust\_ratings as the column to aggregate.
2. Using the pivot table, find Pearson’s correlation.
3. Input the username.
4. Get a list of all restaurants rated by the user.
5. For each restaurant in the list:
6. Use the correlation matrix and get the row of that particular restaurant. Drop all NaN values.
7. Multiply each of the correlation value with the rating that particular restaurant.
8. You now have a series containing the index of the restaurants along with its correlation to the inputted restaurant. (end for)
9. Arrange this series in decreasing order of the correlation score.
10. Obtain the top recommendations and get the restaurant name using the index.

**DIFFERENCE BETWEEN CONTENT-BASED AND ITEM-BASED FILTERING**