# Lab2 for STAT 415

Xiaolong Liu

April 2024

## 1 Introduction

In this report, our goal is to gain insight into different systems of recommendation. In particular, we will focus on popularity matching, content-based filtering, and collaborative filtering methods. All three of these methods are important problems in the technology and data science communities. Broadly speaking, the outcome variable of interest in this dataset will be the user review score. We will at times use direct predictive modeling methods to predict the user review score from the data. However, much of our analysis will also focus on how to find similar restaurants and similar users across the dataset. Recommendations can then be made by finding restaurants that are similar to other restaurants the user enjoyed.

## 2 Exploratory Data Analysis

First of all, we found that there were some extra spaces behind the restaurants' names, which could lead to incorrect results, so we removed these extra spaces. Additionally, there were some extra commas (',') before the **Longitude** in `restaurant.csv`. We removed these commas.

Also, we found that some of the **Date of Review** entries were not correct, such as **4-0-2022**, which is a typo. We corrected this to **4-01-2022**, and then we transformed this variable into a datetime format.

Furthermore, we noted that some reviewers rated the same restaurant multiple times. We will keep only the latest review for these cases.

### 2.1 Missing Values

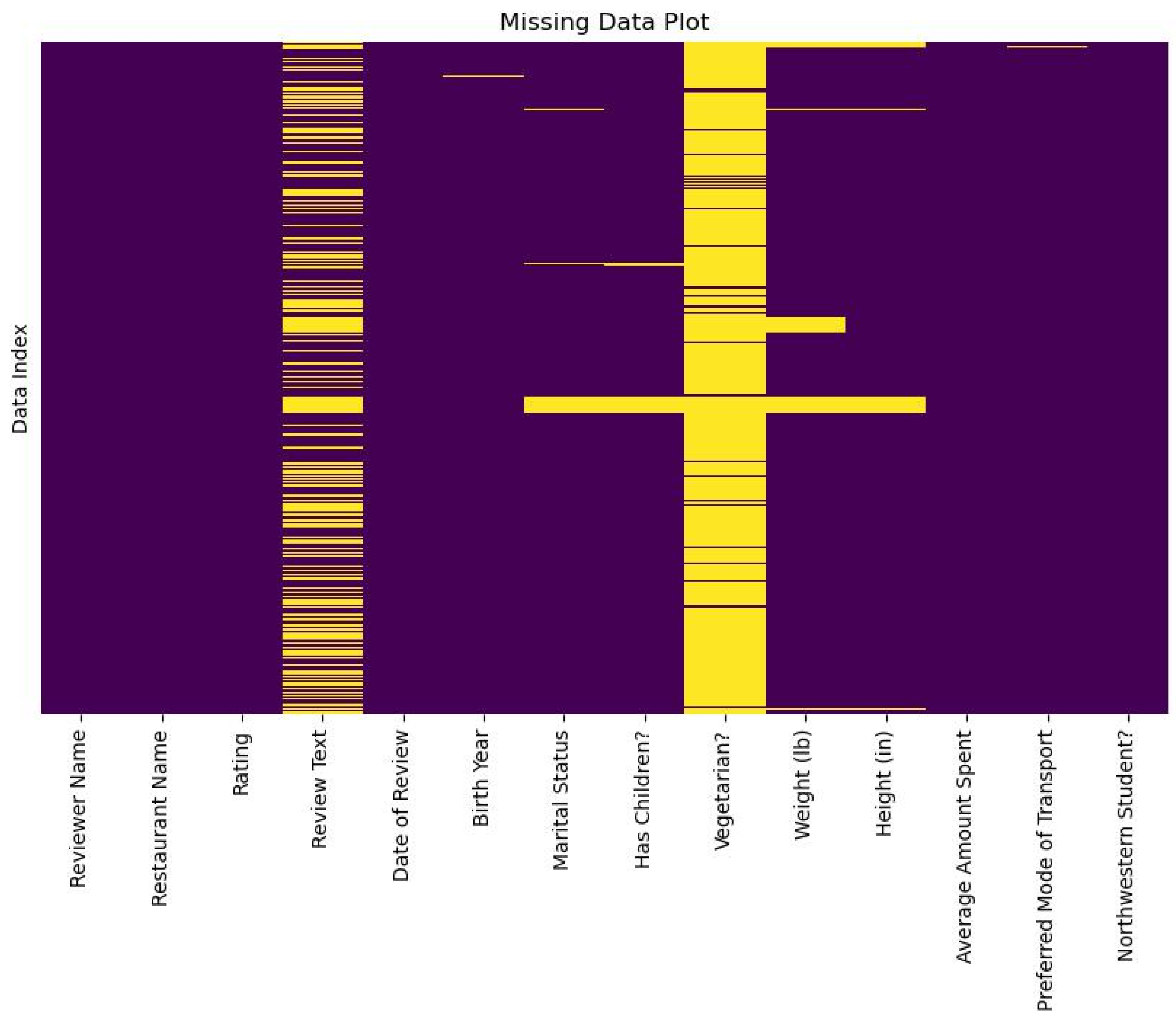
First, Let's examine the missingness in the dataset. There's no missingness in Restaurants.csv. Below is the missing plot for the Review.csv. Out of total 1436 observations, there are 38.3% missingness in **Review Text**, 0.14% missingness in **Birth Year**, 2.44% missingness in **Marital Status**, 2.65% missingness in **Has Children?**, 94.43% missingness in **Vegetarian?**, 6.75% missingness in **Weight (lb)**, 3.76% missingness in **Height (in)**, 0.07% missingness in **Average Amount Spent** and 0.28% missingness in **Preferred Mode of Transport**.

Of course, this do have impact on our further analysis, since the dataset is sparse. But this is also very common in the dataset of recommendation system and there're some methods to deal with it. And we will use some of them in the following part:

1) Matrix Factorization Techniques: Matrix factorization techniques are particularly effective for handling missing data in recommendation systems. These methods decompose the user-item interaction matrix into lower-dimensional user and item latent factor matrices. For example, we can use SVD to achieve this.

2) Using Content-Based Features: Incorporate content-based features (e.g., item descriptions, user demographics) into the recommendation algorithm to mitigate the impact of missing ratings. This approach can help provide recommendations based on item or user attributes, even when interaction data is sparse.

3) Clustering: For methods like k-nearest neighbors (k-NN), handle missing data by computing similarities based on the common ratings between users or items. Alternatively, impute missing values using the average of neighbors' ratings before computing similarities.



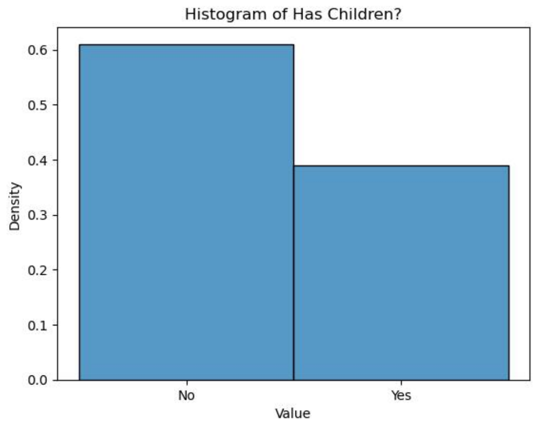
## 2.2 Data Distribution

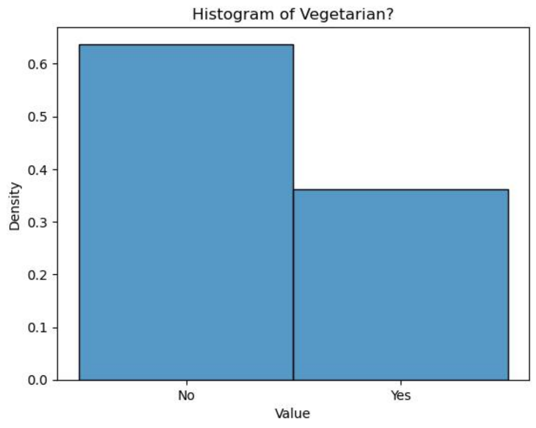
Next, we examine the data distribution (ignoring missingness temporarily) for the following variables in `Restaurants.csv`:

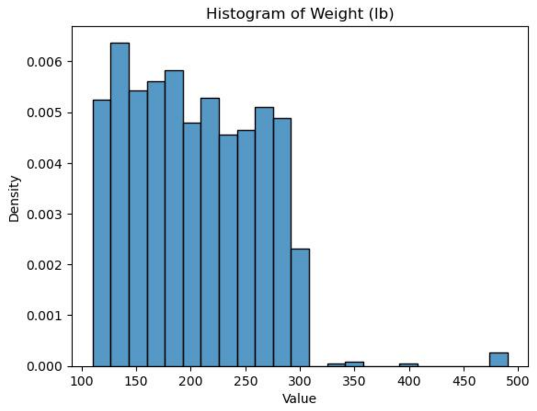
- **Has Children?**
- **Vegetarian?**
- **Weight (lb)**
- **Preferred Mode of Transport**
- **Average Amount Spent**
- **Cuisine**

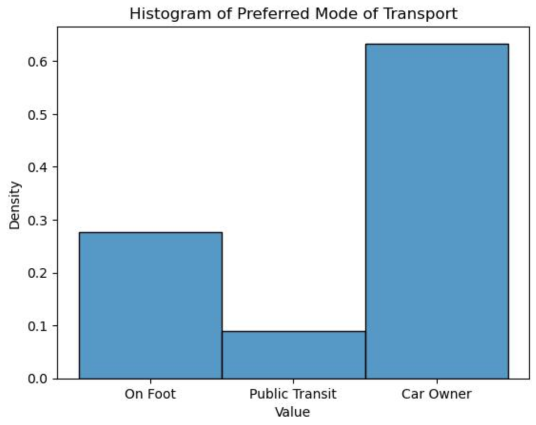
Here are the (density) histograms of these variables. According to these distributions, we could find that this dataset is not balanced. For **Has Children?**, **Vegetarian?**, **Preferred Mode of Transport**, and

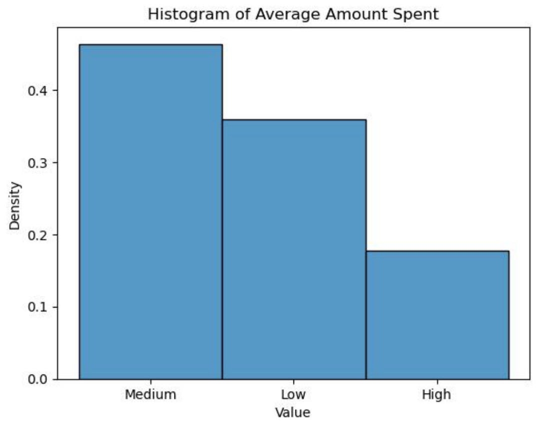
**Average Amount Spent**, there are obvious differences between different groups. For **Weight (lb)**, there are several outliers. In the case of **Cuisine**, **American** accounts for the majority, followed by **Japanese** and others.

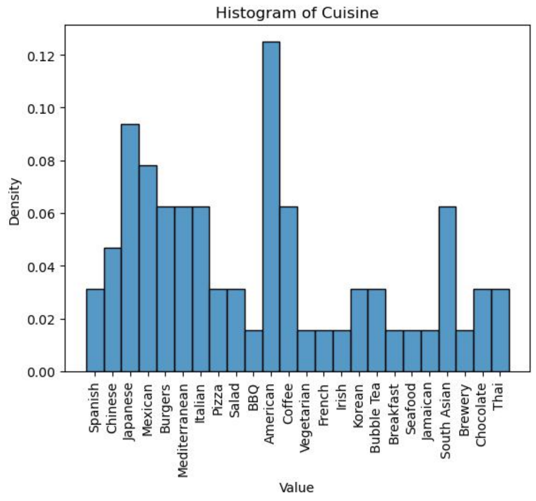








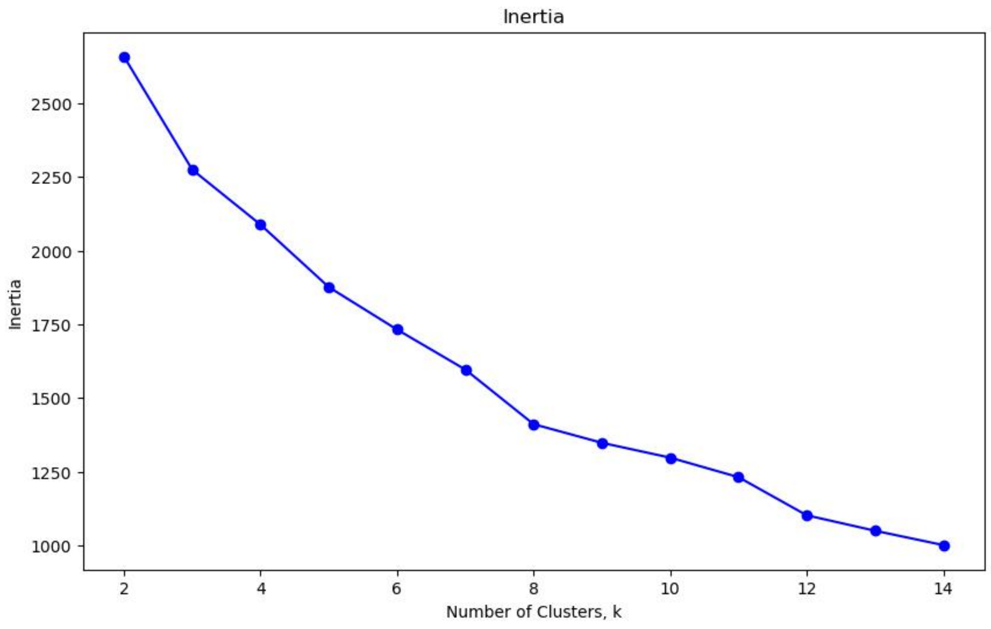


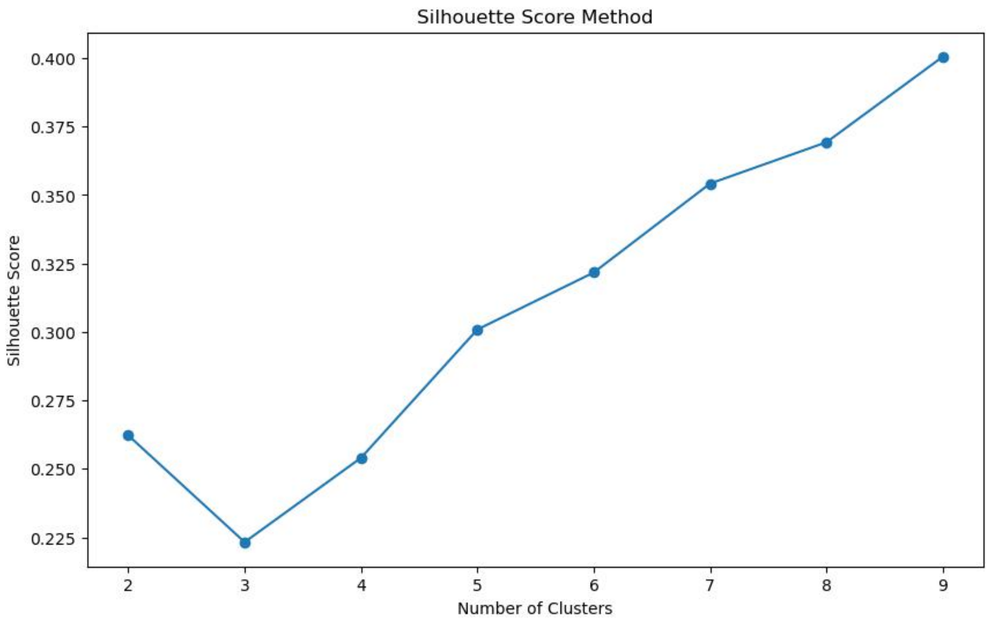


## 2.3 Clustering

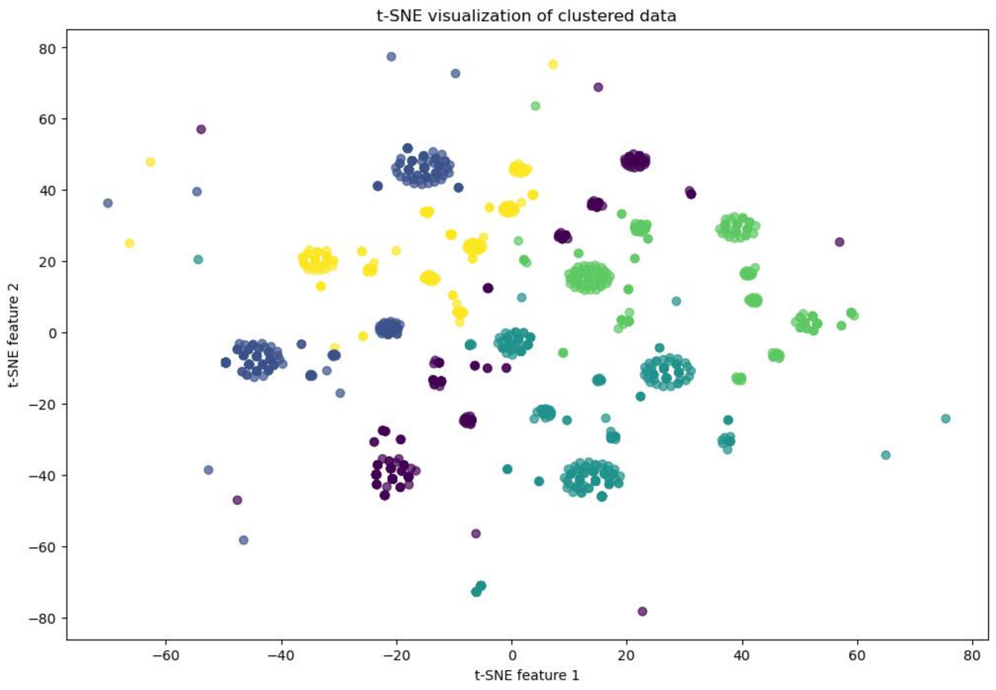
Now, we perform clustering on one-hot encoded user demographic data **Marital Status**, **Has Children?**, **Vegetarian?**, **Average Amount Spent**, **Preferred Mode of Transport**, and **Northwestern Student?** to identify distinct user groups. We apply different clustering methods to determine the most effective approach.

**K-means**: First, we calculate the inertia to evaluate the within-cluster sum of squares. The elbow method, which is usually employed to find the optimal number of clusters $k$, does not show a clear elbow in this case (a similar ambiguity is observed for silhouette scores). Consequently, we choose $k = 5$ as it represents a reasonable balance between the number of clusters and the clarity of the resulting separation. This choice is also supported by a noticeable inflection point in the silhouette scores at $k = 5$.

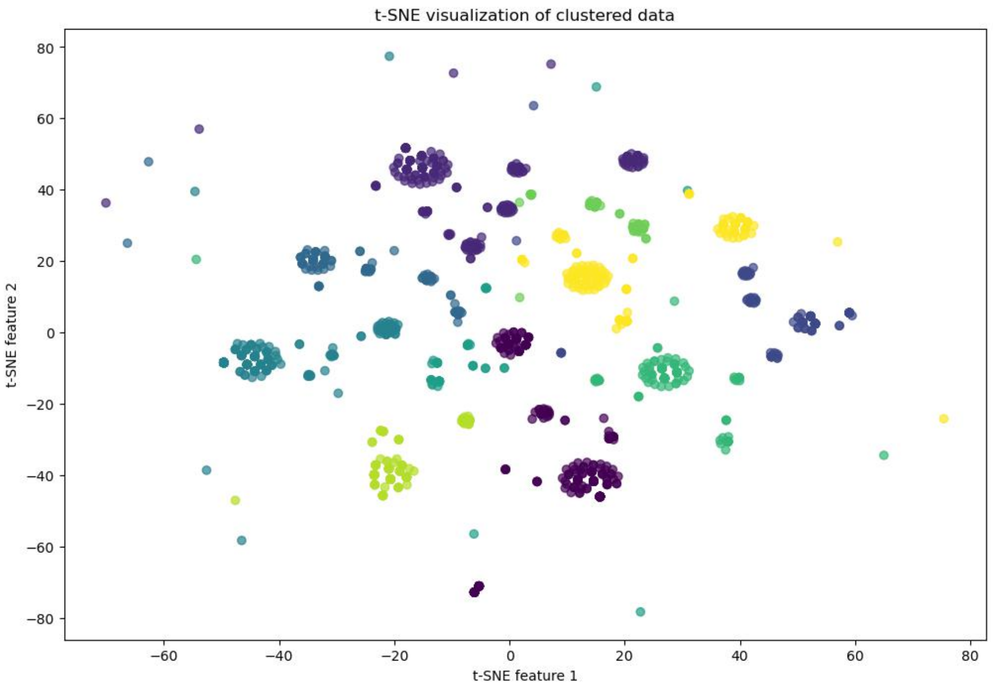




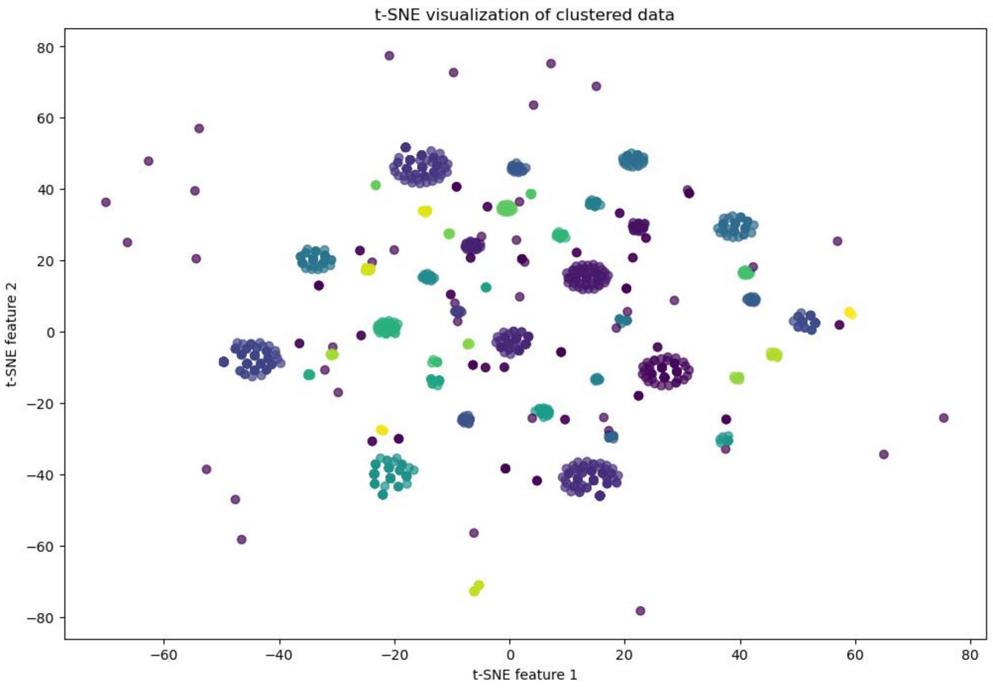
Due to the high dimensionality of the clustering, direct visualization is not feasible. However, dimensionality reduction techniques such as t-SNE can be employed to facilitate a two-dimensional visualization. Utilizing t-SNE, we observe that K-means clustering with $k = 5$ reveals certain characteristics of the clusters, although the delineation is not particularly distinct.



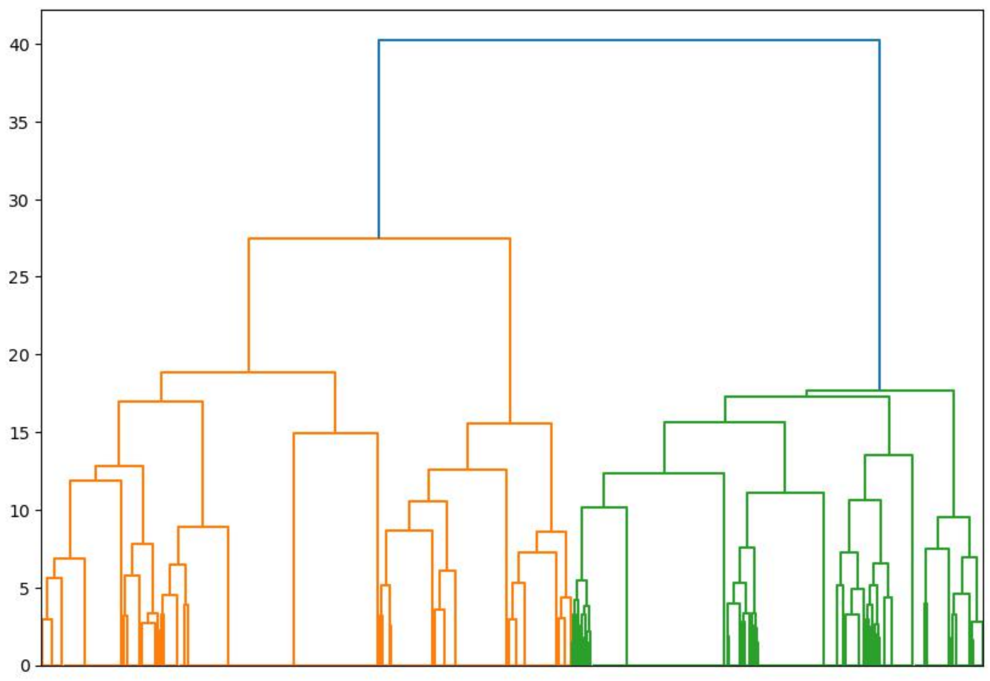
Given that the t-SNE visualization suggests the presence of more than 5 clusters, we experiment with increasing the number of clusters to $k = 10$. The resulting t-SNE plot is as follows. Unfortunately, this configuration does not yield the clarity of cluster separation that we anticipated.



Subsequently, we experiment with the DBSCAN clustering algorithm. The performance of DBSCAN, as interpreted through the t-SNE visualization, is not satisfactory.

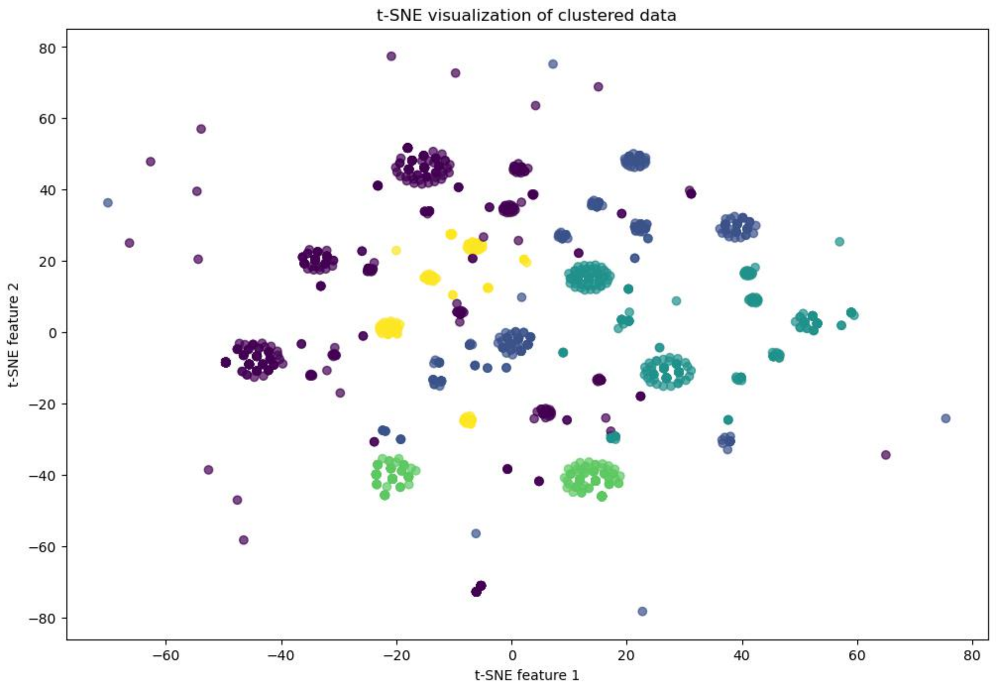


Next, we apply Agglomerative Clustering to the dataset. The hierarchical nature of this method is illustrated in the dendrogram provided below.

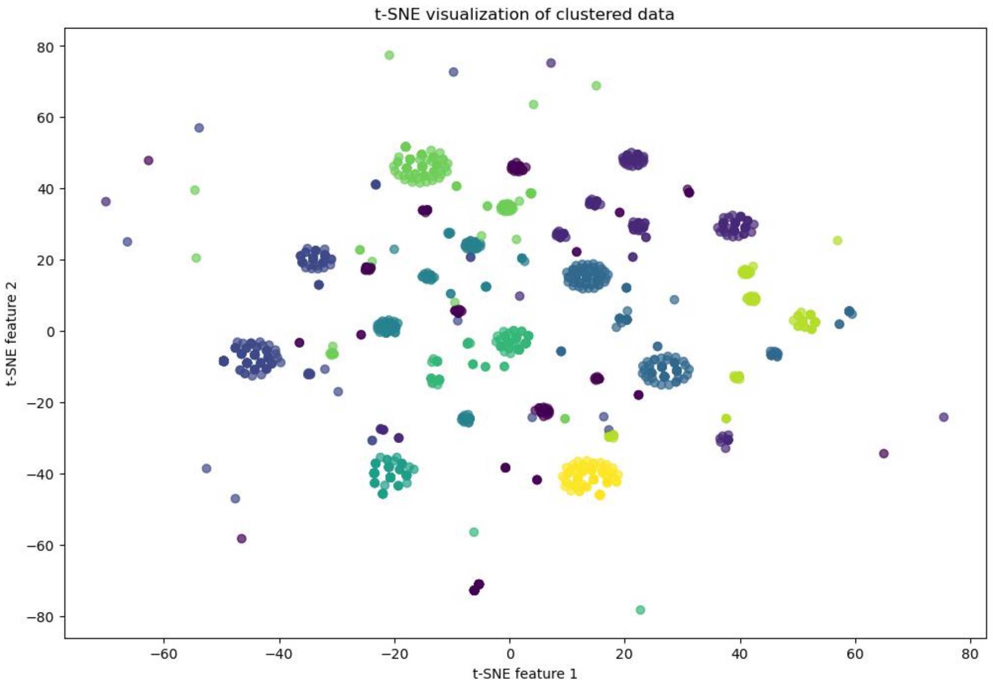


We choose to test $k = 5$ and $k = 10$ for Agglomerative Clustering to facilitate a direct comparison with K-means. The respective t-SNE plots for these cluster counts are provided below. Based on the t-SNE visualizations, Agglomerative Clustering appears to perform better than K-means, as it more effectively delineates the properties of the clusters. Consequently, we decide to proceed with $k = 10$ for further analysis.

$k = 5$:

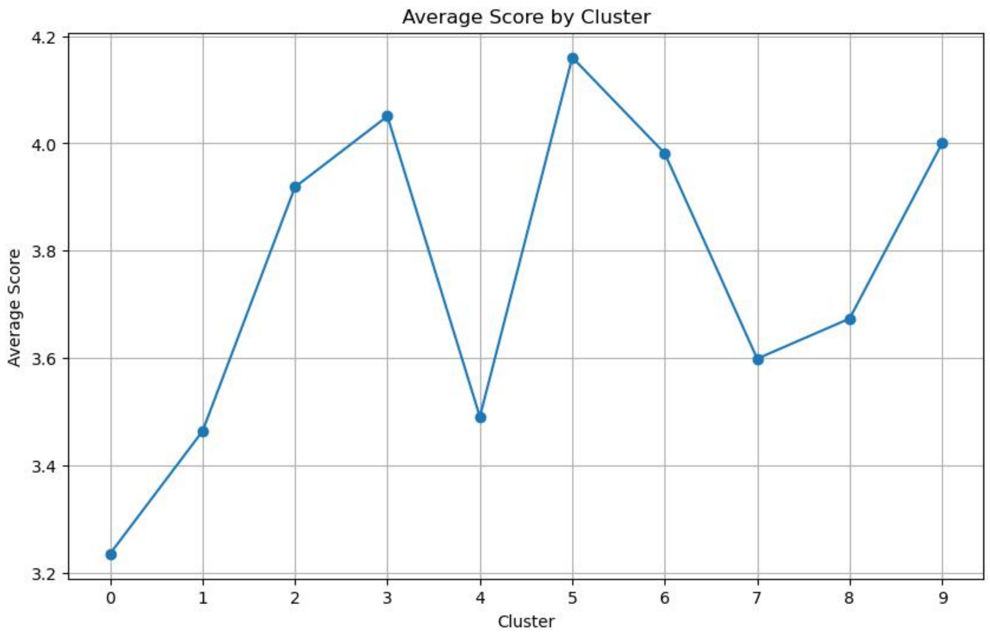


$k = 10$:



The average review scores for each identified cluster are as follows. We observe that clusters 0, 1, and 4 have lower average review scores compared to others. Conversely, clusters 3 and 5 exhibit higher average review scores than the rest. This differentiation in review scores may indicate differing levels of satisfaction or performance among the clusters.

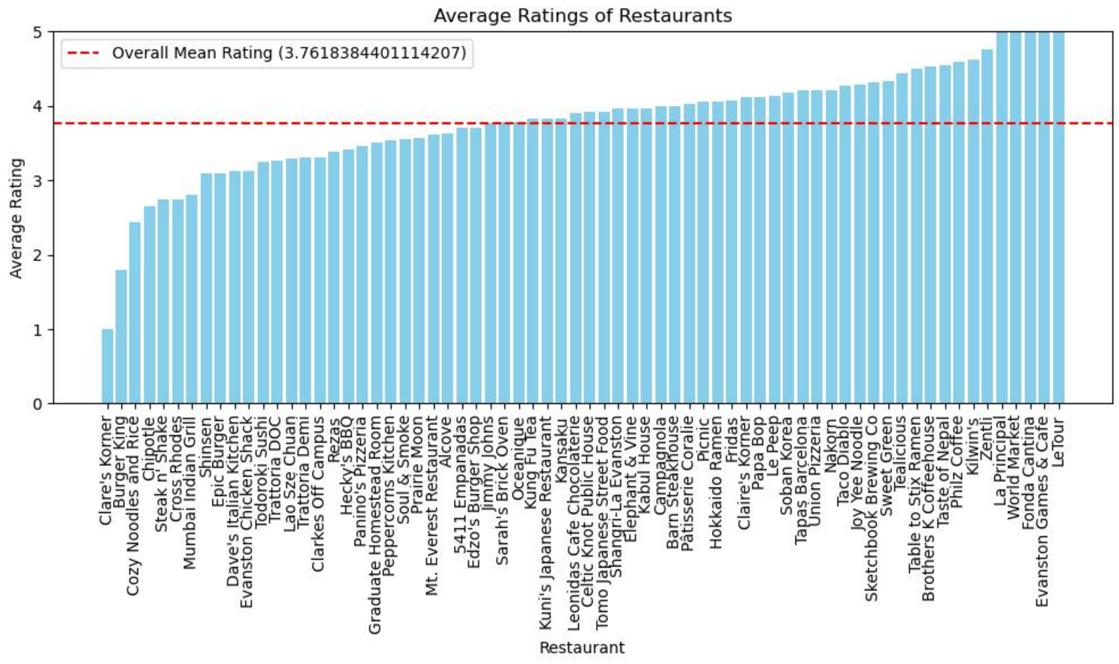
```
Cluster
0    3.235294
1    3.463687
2    3.918455
3    4.050505
4    3.490566
5    4.160000
6    3.981308
7    3.598684
8    3.673469
9    4.000000
```



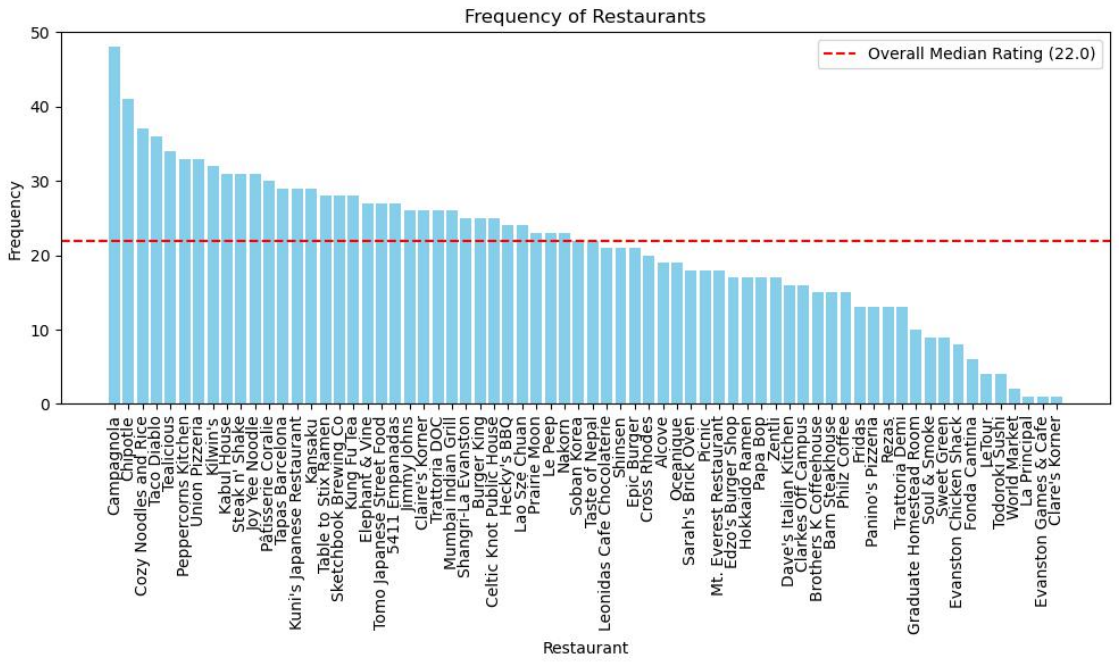
## 3 Popularity Matching

### 3.1 Average Score and median number of reviews

The general dataset's average score of rating is around 3.74. There are 5 restaurants that achieve the highest score (5): **World Market**, **Fonda Cantina**, **Evanston Games & Cafe**, **LeTour**, and **La Principal**.



The median number of ratings is 22. **Campagnola** has the highest number of ratings (48).



## 3.2 A simple recommendation engine based on a popularity score.

We use a shrinkage estimator for each cuisine type. To be concrete, let $\mu_s$ be the mean rating for all restaurants in this cuisine type and $N_\mu$ be the mean number of ratings in this cuisine type. Denote $\mu_p$ as the mean rating for restaurant $i$ and $N_p$ as the total number of ratings for restaurant $i$.

Let $\alpha = \min\left(\frac{N_p}{N_\mu}, 1\right)$. Then, the shrinkage estimator would be $(1-\alpha)\mu_s + \alpha\mu_p$. We will recommend the restaurant that has the highest shrinkage estimator.

Here are the recommendations according to our simple recommendation engine:

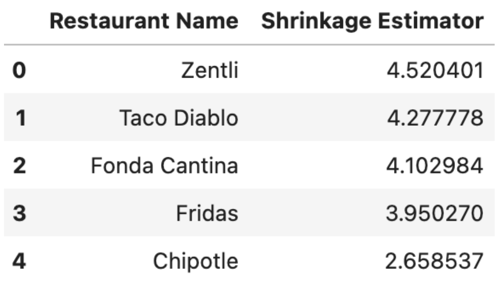
**Spanish**

| | Restaurant Name | Shrinkage Estimator |
|---|---|---|
| **0** | Tapas Barcelona | 4.206897 |
| **1** | 5411 Empanadas | 3.713010 |

**Chinese**

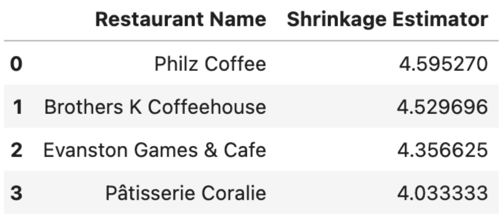
| | Restaurant Name | Shrinkage Estimator |
|---|---|---|
| **0** | Joy Yee Noodle | 4.290323 |
| **1** | Peppercorns Kitchen | 3.545455 |
| **2** | Lao Sze Chuan | 3.372934 |

**Mexican**



| | Restaurant Name | Shrinkage Estimator |
|---|---|---|
| **0** | Zentli | 4.520401 |
| **1** | Taco Diablo | 4.277778 |
| **2** | Fonda Cantina | 4.102984 |
| **3** | Fridas | 3.950270 |
| **4** | Chipotle | 2.658537 |

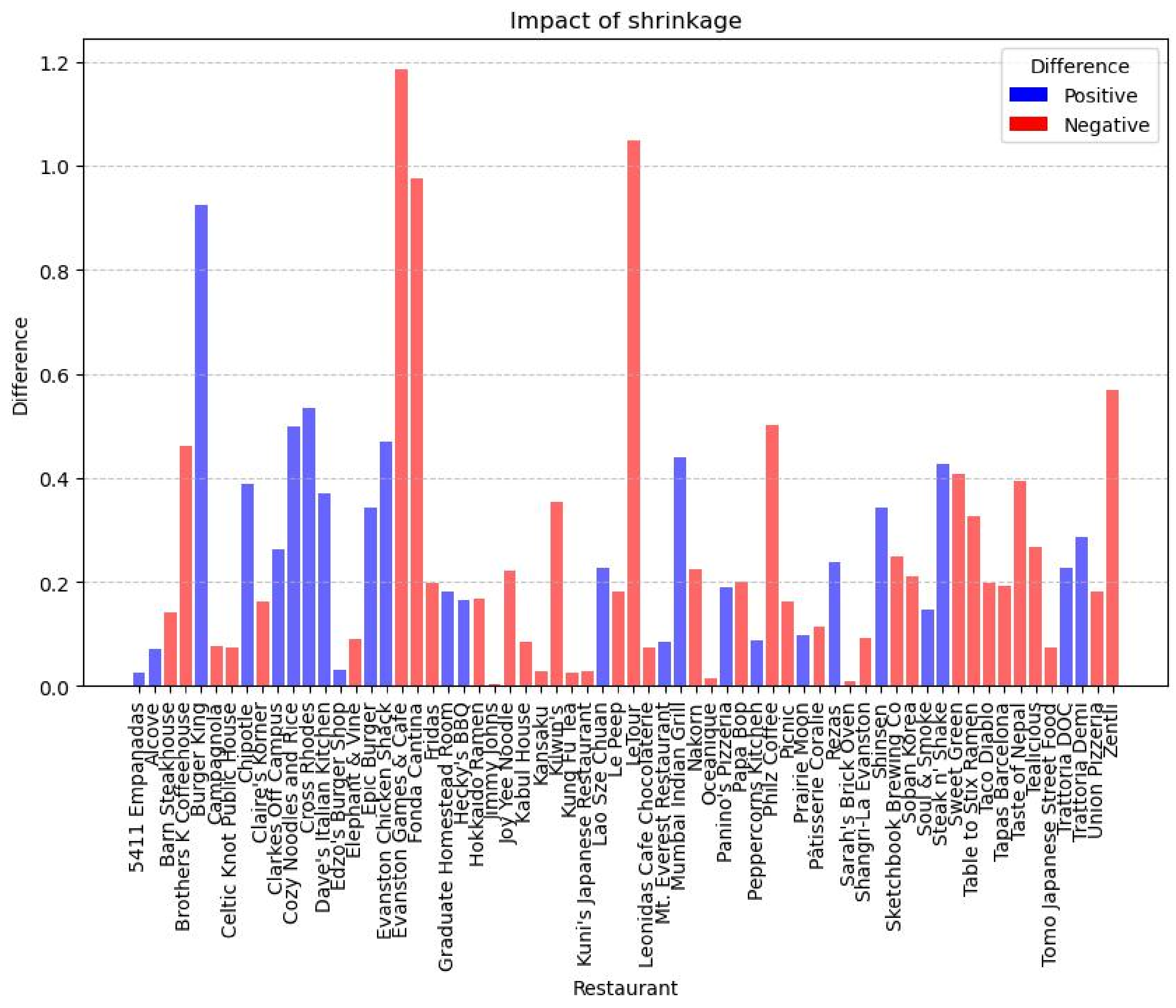
**Coffee**



| | Restaurant Name | Shrinkage Estimator |
|---|---|---|
| **0** | Philz Coffee | 4.595270 |
| **1** | Brothers K Coffeehouse | 4.529696 |
| **2** | Evanston Games & Cafe | 4.356625 |
| **3** | Pâtisserie Coralie | 4.033333 |

## 3.3 Impact of Shrinkage

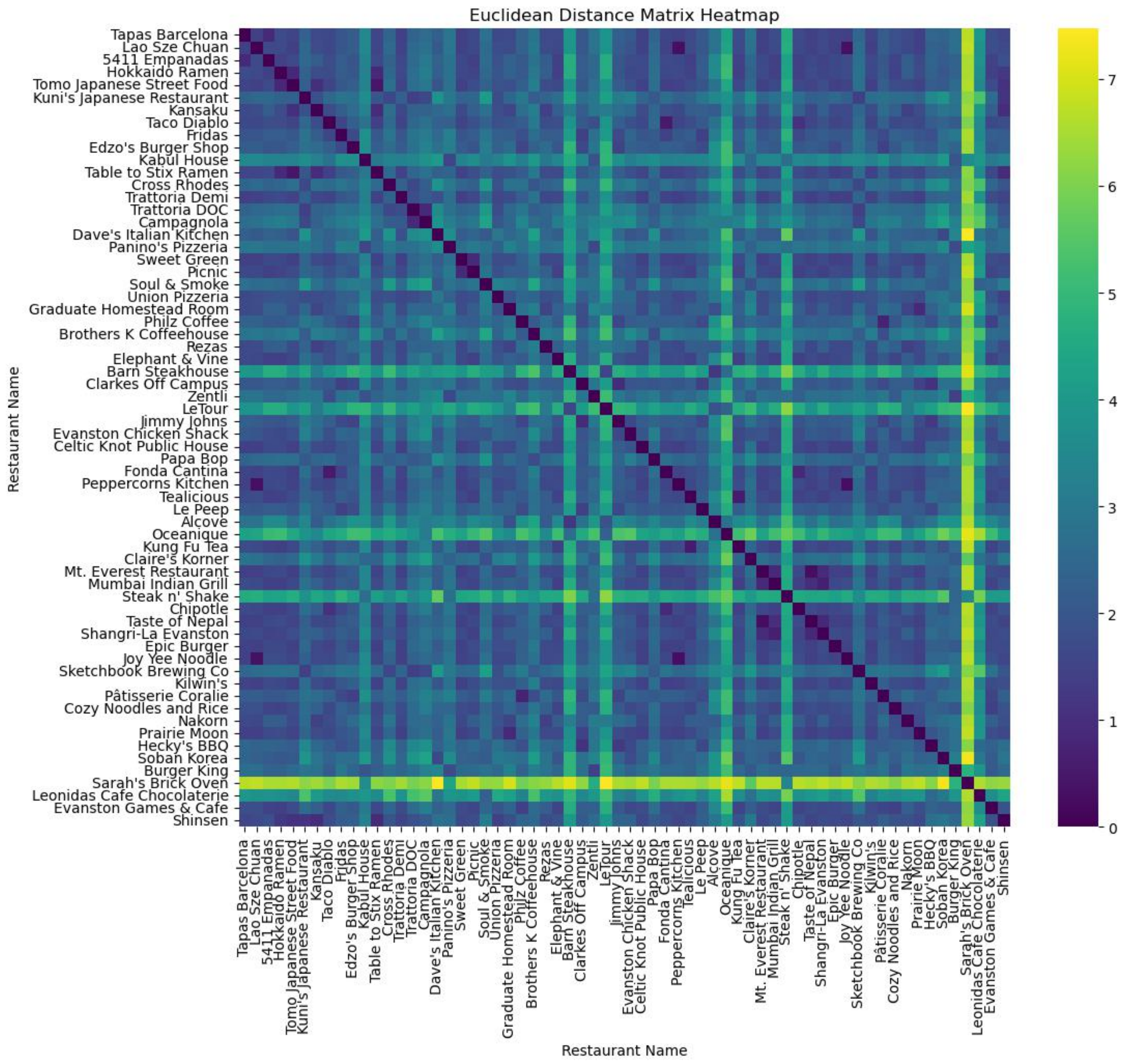
In this part, we use a different shrinkage formula, which is $\frac{N_\mu \mu_s + N_p \mu_p}{N_\mu + N_p}$, to identify the restaurant that benefits the most from this adjustment and the one that is most negatively affected. Here's the plot that shows the positive and negative changes of these restaurants. **Burger King** experiences the highest positive changes, while **Evanston Games & Cafe** undergoes the most significant negative changes.

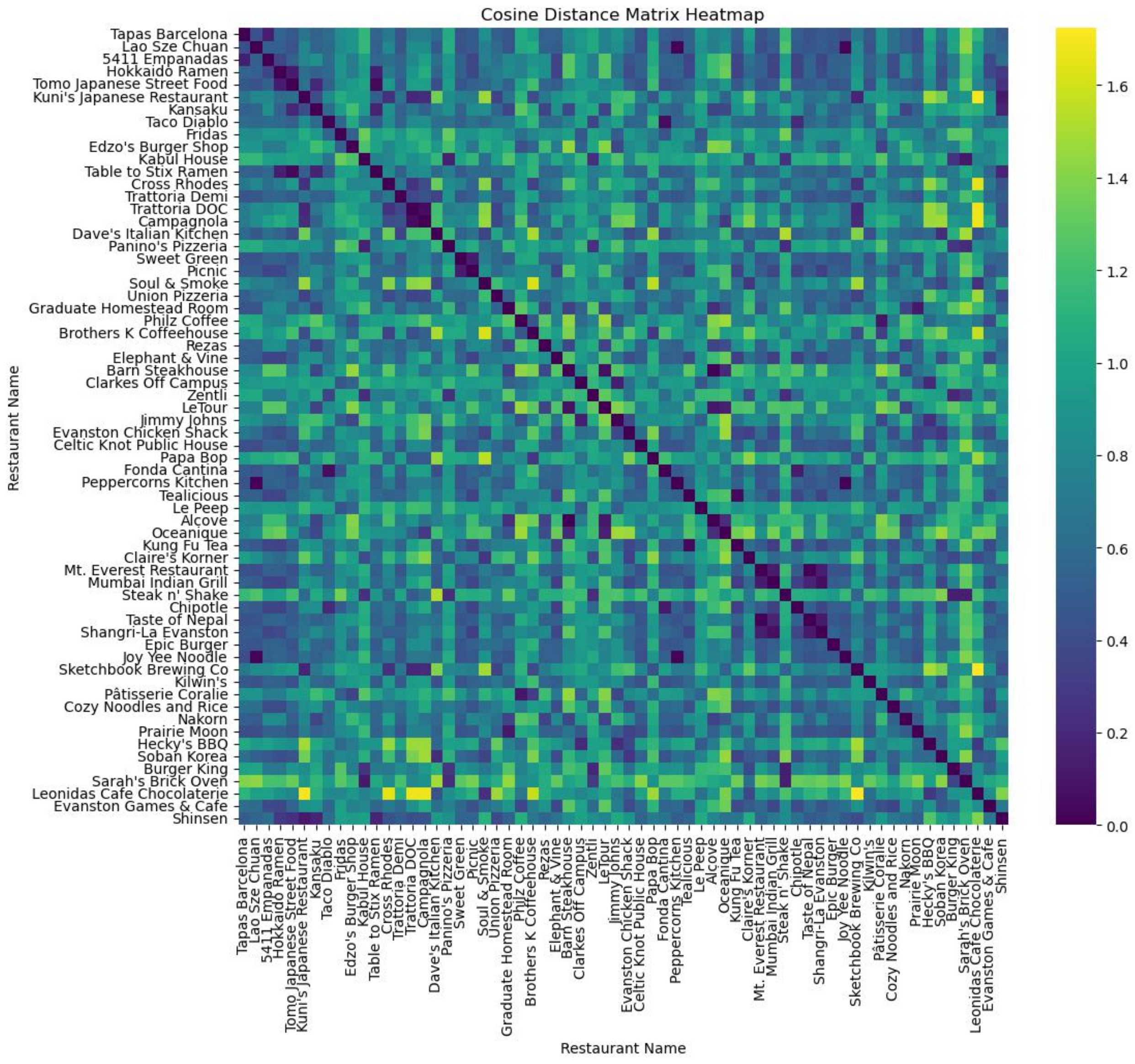


## 4 Content-based Filtering

### 4.1 The Euclidean distance between every restaurant

First, we transform the categorical variables **Cuisine** and **Open After 8pm?** into one-hot embeddings. Additionally, we scale the numerical variables **Latitude**, **Longitude**, and **Average Cost**. Since there are 64 restaurants, the distance between every restaurant results in a large matrix; therefore, we decide to use a heatmap to display this information.


Euclidean Distance Matrix Heatmap
Restaurant Name
Restaurant Name
Tapas Barcelona
Lao Sze Chuan
5411 Empanadas
Hokkaido Ramen
Tomo Japanese Street Food
Kuni's Japanese Restaurant
Kansaku
Taco Diablo
Fridas
Edzo's Burger Shop
Kabul House
Table to Stix Ramen
Cross Rhodes
Trattoria Demi
Trattoria DOC
Campagnola
Dave's Italian Kitchen
Panino's Pizzeria
Sweet Green
Picnic
Soul & Smoke
Union Pizzeria
Graduate Homestead Room
Philz Coffee
Brothers K Coffeehouse
Rezas
Elephant & Vine
Barn Steakhouse
Clarkes Off Campus
Zentli
LeTour
Jimmy Johns
Evanston Chicken Shack
Celtic Knot Public House
Papa Bop
Fonda Cantina
Peppercorns Kitchen
Tealicious
Le Peep
Alcove
Oceanique
Kung Fu Tea
Claire's Korner
Mt. Everest Restaurant
Mumbai Indian Grill
Steak n' Shake
Chipotle
Taste of Nepal
Shangri-La Evanston
Epic Burger
Joy Yee Noodle
Sketchbook Brewing Co
Kilwin's
Pâtisserie Coralie
Cozy Noodles and Rice
Nakorn
Prairie Moon
Hecky's BBQ
Soban Korea
Burger King
Sarah's Brick Oven
Leonidas Cafe Chocolaterie
Evanston Games & Cafe
Shinsen
7
6
5
4
3
2
1
0



## 4.2 Compare to the cosine distance

We compute the cosine distance instead of the Euclidean distance. We now compare two different metrics on the pairs (**Peppercorns Kitchen**, **Epic Burger**) and (**Peppercorns Kitchen**, **Lao Sze Chuan**).

The Euclidean distance between **Peppercorns Kitchen** and **Epic Burger** is: 1.436. The cosine distance between **Peppercorns Kitchen** and **Epic Burger** is: 0.472.

The Euclidean distance between **Peppercorns Kitchen** and **Lao Sze Chuan** is: 0.315. The cosine distance between **Peppercorns Kitchen** and **Lao Sze Chuan** is: 0.019.

We believe both metrics adequately reveal the relationships and similarities between the restaurants: They all share the same **Average Cost** and are **Open After 8pm**. Their locations differ from each other. In the Euclidean sense, it suggests **Peppercorns Kitchen** and **Epic Burger** are closer, based solely on **Latitude** and **Longitude**. In the cosine sense, it considers **Lao Sze Chuan** and **Epic Burger** to be closer, also based solely on **Latitude** and **Longitude**.

Given that **Peppercorns Kitchen** and **Lao Sze Chuan** both offer **Chinese food**, while **Epic Burger** specializes in **Burgers** within the **Cuisine** category, it makes sense that the distance between **Peppercorns Kitchen** and **Lao Sze Chuan** is smaller than that between **Peppercorns Kitchen** and **Epic Burger**.

## 4.3 Content-based filtering recommendation engine

Now we write a script for a content-based filtering recommendation engine. This script accepts a user's name, identifies the user's favorite restaurant, and then finds the 10 most similar restaurants based on Euclidean distance. Since there are 64 restaurants in the restaurant.csv, but 68 different restaurants in reviews.csv, we do not have information of **Clare's Korner**, **La Principal**, **Todoroki Sushi**, and **World Market**. So if we encounter this, we will use the second most favorite one instead.

Now we choose **Willie Jacobsen** as the input. We find the restaurant that he give the highest score first. Then we find top 10 closest restaurants with this restaurant based on Euclidean distance (which we calculate before). And then we return these restaurants by order (the one with smaller distance is more preferable). Here're the restaurants we recommend for **Willie Jacobsen**.

```
Restaurant Name
Clarkes Off Campus        0.930041
Hecky's BBQ               1.235294
Edzo's Burger Shop        1.490472
Pâtisserie Coralie        1.502059
Philz Coffee              1.578572
Evanston Chicken Shack    1.641670
Le Peep                   1.656719
Fridas                    1.664978
Prairie Moon              1.695856
Mumbai Indian Grill       2.000934
```

## 4.4 A method to compare performance

The recommendation engine we previously constructed assigns a rank to all restaurants based on the Euclidean or cosine distance from the user's favorite restaurant. Often, customers rate more than one restaurant. Therefore, we can employ the following methodology: we gather all the ratings a user has provided for restaurants other than his or her favorite. These ratings are then used to assign a rank to these restaurants, denoted as $r_0$. Simultaneously, the recommendation engine generates a comparative ranking for these restaurants, which we transform into rankings, denoted as $r_{euclidean/cosine}$. We then employ a metric to compare $r_0$ and $r_{euclidean/cosine}$ to evaluate the performance of the recommendation engine.

For example, consider the user **Willie Jacobsen**, who, aside from his favorite restaurants, has rated **Chipotle** and **Steak n' Shake** with the **Ratings** of 3 and 1, respectively. Consequently, we will assign **Chipotle** and **Steak n' Shake** ranks of 1 and 2, respectively, where $r_0 = [1, 2]$. Additionally, based on the Euclidean distance from the user's favorite restaurant, our recommendation engine assigns these two restaurants ranks of 13 and 60, respectively. For the purposes of comparison within this specific context, we adjust these ranks to 1 and 2, respectively, resulting in $r_{euclidean} = [1, 2]$. In this scenario, both $r_0$ and $r_{euclidean}$ coincide, which aligns with the expectations we have set for our engine across all users.

Thus, we will select users who have more than 2 ratings, as fewer ratings do not yield sufficient information. We utilize Spearman's Rank Correlation Coefficient as the criterion because we are primarily concerned with the relative order—higher correlation indicates a more preferable outcome. We will calculate the average Spearman's Rank Correlation Coefficient for both the Euclidean distance and the 'true' rank, as well as for the cosine distance and the 'true' rank.

Note that some individuals assign the same scores to all restaurants, resulting in a NaN for Spearman's Rank Correlation Coefficient. Since these observations provide no information about the superiority of one method over another (Euclidean or cosine), they will not be utilized.

The average correlation coefficient for the Euclidean-based method is 0.095. The average correlation coefficient for the Cosine-based method is 0.106. Although this difference is small, it suggests that the cosine-based method is more preferable.

## 5 Natural Language Analysis – version A

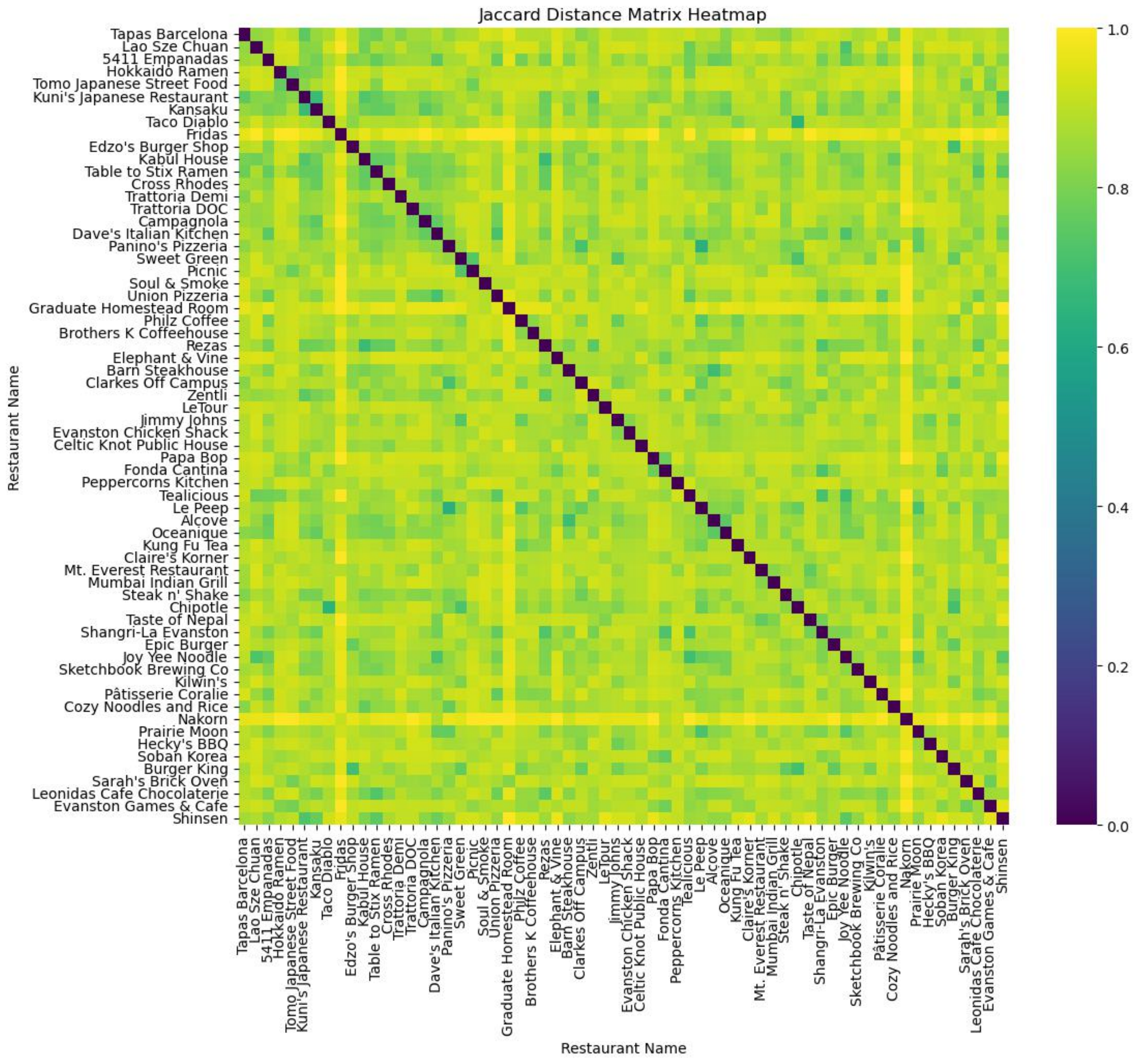
### 5.1 Construct Augmented Description

We consider the **Brief Description** column of the Restaurants dataset. We augment this description by appending the restaurant's cuisine type to the end of the description. For example, for Tapas Barcelona, the description is: 'Festive, warm space known for Spanish small plates is adorned with colorful modern art & posters.' The cuisine is **Spanish**. Therefore, the augmented description becomes: 'Festive, warm space known for Spanish small plates is adorned with colorful modern art & posters. Spanish.' We name this variable **Augmented Description**.

### 5.2 Compute the Jaccard matrix

We use 1 minus the Jaccard Similarity as the Jaccard Distance and compute the Jaccard matrix. For **Burger King** and **Edzo's Burger Shop**, the Jaccard distance is 0.7. For **Burger King** and **Oceanique**, the Jaccard distance is 0.88. For **Lao Sze Chuan** and **Kabul House**, the Jaccard distance is 0.78125.

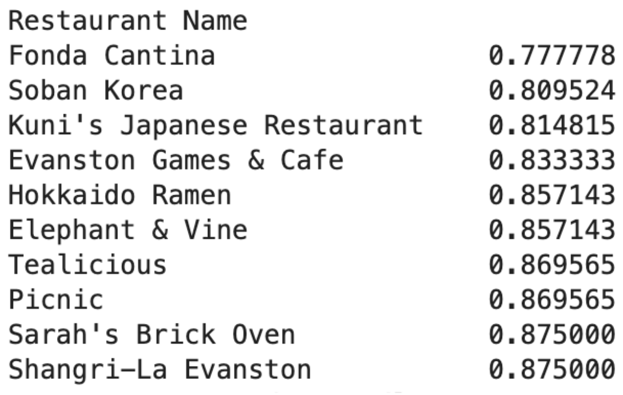
According to this, we can find that **Burger King** is closer to **Edzo's Burger Shop** than to **Oceanique**, which makes sense, since they are both burger shops.
Here is the Jaccard matrix showed by heat map:



## 5.3 Jaccard distance-based recommendation

Now we use the Jaccard distance matrix to perform recommendations. The logic is exactly the same as the one in the part **Content-based Filtering Recommendation Engine**, but now we employ the Jaccard distance instead of Euclidean or cosine distances.

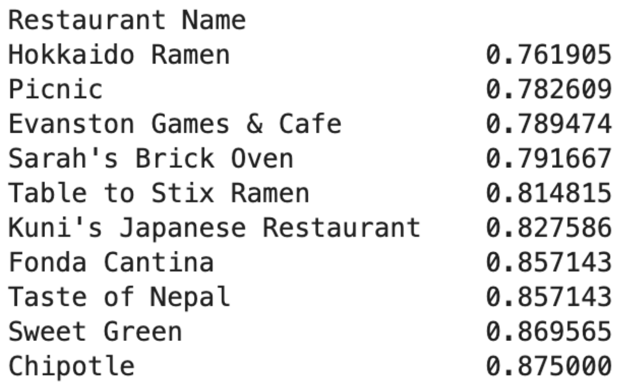
Here are the top 10 recommendations for **Calvin Smith**:



```
Restaurant Name
Fonda Cantina                  0.777778
Soban Korea                    0.809524
Kuni's Japanese Restaurant     0.814815
Evanston Games & Cafe          0.833333
Hokkaido Ramen                 0.857143
Elephant & Vine                0.857143
Tealicious                     0.869565
Picnic                         0.869565
Sarah's Brick Oven             0.875000
Shangri-La Evanston            0.875000
```

Figure 1: Top 10 Recommendations for Calvin Smith

Here are the top 10 recommendations for **Solomon M**:



```
Restaurant Name
Hokkaido Ramen                 0.761905
Picnic                         0.782609
Evanston Games & Cafe          0.789474
Sarah's Brick Oven             0.791667
Table to Stix Ramen            0.814815
Kuni's Japanese Restaurant     0.827586
Fonda Cantina                  0.857143
Taste of Nepal                 0.857143
Sweet Green                    0.869565
Chipotle                       0.875000
```

Figure 2: Top 10 Recommendations for Solomon M