HW2

May 3, 2023

1 415-HW1

1.1 Shuo Han

```
2023-05-03 23:16:09.343835: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
```

1.2 EDA

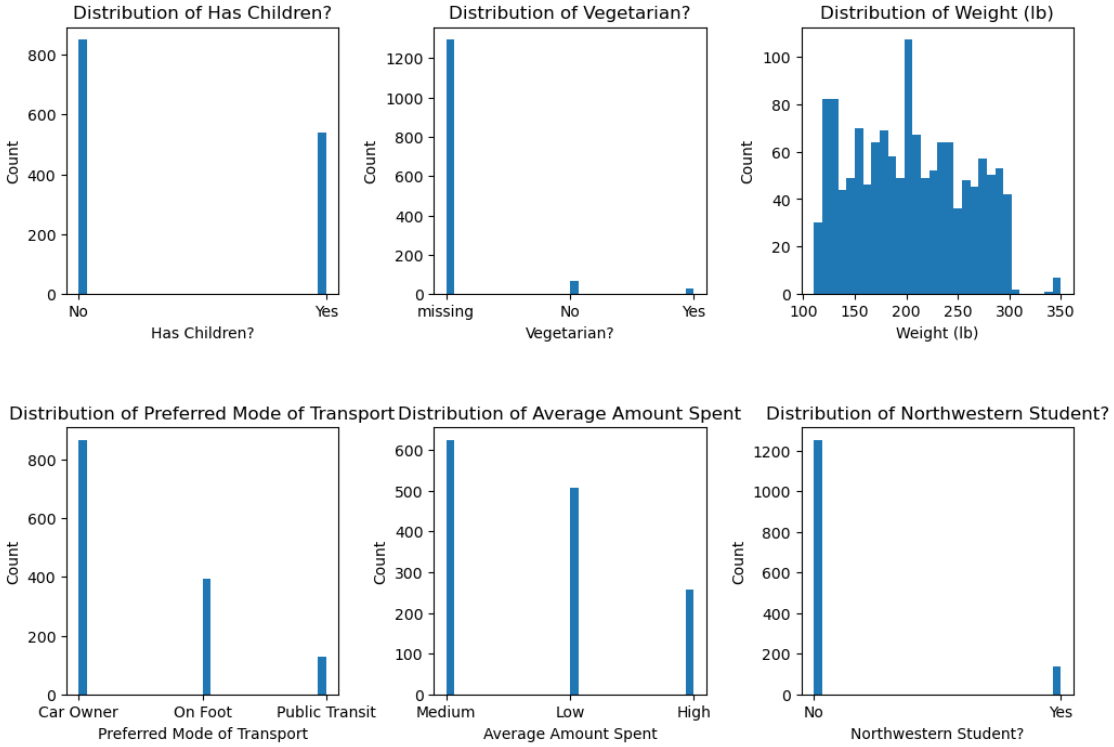
1.2.1 1 Import and examine the data

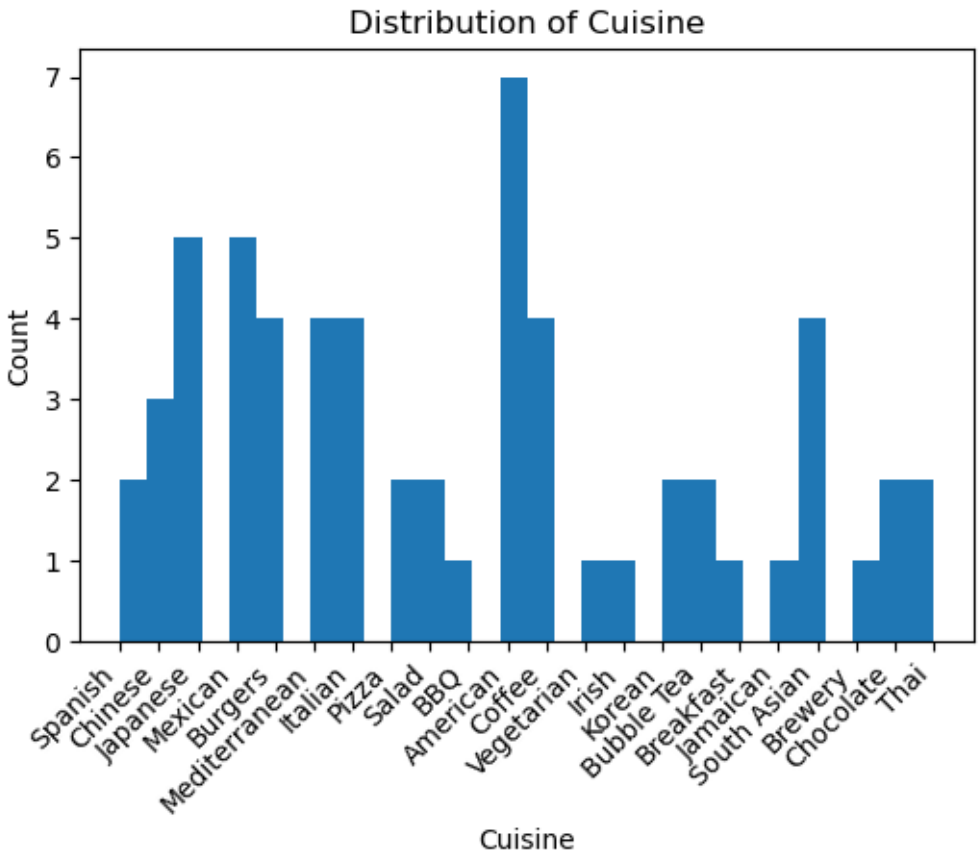
```
Duplicated columns: Index([], dtype='object')
Duplicated columns: Index([], dtype='object')
Missing columns: Index([], dtype='object')
Missing columns: Index([], dtype='object')
Number of outliers: 8
Number of data: 1444
Percentage of outliers:  0.00554016620498615
Number of outliers: 3
Number of data: 63
Percentage of outliers:  0.047619047619047616
Reviewer Name                     0
Restaurant Name                   0
Rating                            0
Review Text                     546
Date of Review                    0
Birth Year                        2
Marital Status                   35
Has Children?                    38
Vegetarian?                    1342
Weight (lb)                      97
Height (in)                      54
Average Amount Spent              2
Preferred Mode of Transport       7
Northwestern Student?             1
dtype: int64
```

```
Restaurant Name      0
Cuisine              0
Latitude             0
Longitude            0
Average Cost         0
Open After 8pm?      0
Brief Description    0
dtype: int64
Reviewer Name                  0.000000
Restaurant Name                0.000000
Rating                         0.000000
Review Text                    0.378116
Date of Review                 0.000000
Birth Year                     0.001385
Marital Status                 0.024238
Has Children?                  0.026316
Vegetarian?                    0.929363
Weight (lb)                    0.067175
Height (in)                    0.037396
Average Amount Spent           0.001385
Preferred Mode of Transport    0.004848
Northwestern Student?          0.000693
dtype: float64
```

In this part, I have identified duplicated columns and columns with entirely missing data in these two sheets but there is not one at all. Then I have identified outliers in these two sheets, and we can see that the percentage of outliers in the review sheet is 0.00554016620498615, and in the restaurant sheet 0.047619047619047616, which are both relatively small, so I have removed these from the dataset. Then I haved checked columns with missing values in these two sheets. There is not missing values in the restaurant sheet, but some in the review sheet. For the review sheet, I have calculated the portion of missing values in the sheet for each variable. The percentage of missing values in column "Vegetarian?" and "Review Text" are fairly large, so I replaced these with value "missing" for "Vegetarian?" and "" for "Review Text", since we are doing content analysis for that. Also, there are some columns with relatively less missing values, so I have replaced these columns 'Weight (lb)' and 'Height (in)' with the mean value of these columns. Also, there are some columns with very few missing value, even less than 3% of the data, 'Birth Year', 'Marital Status', 'Has Children?','Average Amount Spent', 'Preferred Mode of Transport', 'Northwestern Student?', so I have dropped these rows with missing values. And finally, I have added a mew dataset merge, which is merged by the review sheet and restaurant sheet.

1.2.2 2 histograms





I have tried to make histograms for 'has children', 'vegetarian', 'weight', 'prefered mode of transport', 'average amount spent', 'Northwestern student' and 'cusine'. We can see that the dataset is actually not properly balanced. In the histogram of 'Has Children?', we can see that the number of reviews from reviewers with children is almost double the number of reviews from reviewers without children, so it is really unbalanced. For the histogram of 'Vegetarian?', we can see that there are so many reviews from reviewers with unknown vegetarian preference, and there are more reviews from non-vegetarians than vegetarians, so the data is fairly incomplete in column 'Vegetarian?' and not so balanced with these data provide. For the the histogram of 'Weight (lb)', we can see that most of these reviews are from reviewers of weights between aroung 125lb to 220lb, and for these reviewers weighs fairly balanced between 110lb to 310lb, but there are some reviewers weighs aroung 350lb, and some especially more weighs around 210lb and 130lb. In the histogram of 'Preferred Mode of Transport', there are more reviews are from car owners than reviewers who prefers on foot than public transit. Thus, from the aspect of 'Preferred Mode of Transport', we can still say the data is fairly unbalanced. From the the histogram of 'Average Amount Spent', we can see that there are more reviews for restaurants of medium than low than hight average amount spent, and there are big differences in the amount of these reviews. Thus, the data is fairly unbalanced based on 'Average Amount Spent'. From the the histogram of 'Northwestern Student?', we can see that there are much more reviews for restaurants from non-Northwestern Student than Northwestern Student, so the data is really unbalanced based on 'Northwestern Student?'. From the the histogram of 'Cuisine', we can see that there are reviews for 22 types of cuisines, and there are many reviews

for American cuisine, and only a few for BBQ, Vegetarian, Irish, Breakfase, Jamaican and Brewery cusine. There are great differences in the number, so the data is really unbalanced.

1.2.3 3 clustering on the user demographic data

	Birth Year	Count
0	1989.000000	326
1	1961.376344	279
2	1945.740541	185
3	1999.921569	357
4	1976.429167	240

```
/Users/ellenh/anaconda3/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1533792113.py:1
4: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is
deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the
function.
  cluster_mean = rr.groupby('cluster')[cvars].mean()
```

There are 5 clusters(after optimization) by KMeans algorithm, and we can see that the mean birth year of these groups are . The mean birth year of clusters ranges for 54 years, so these are really distinct 5 clusters of users. Also, we can see that there are roughly more users in the group of later mean birth year than the earlier.

1.2.4 4 compute the average review score across the entire cluster

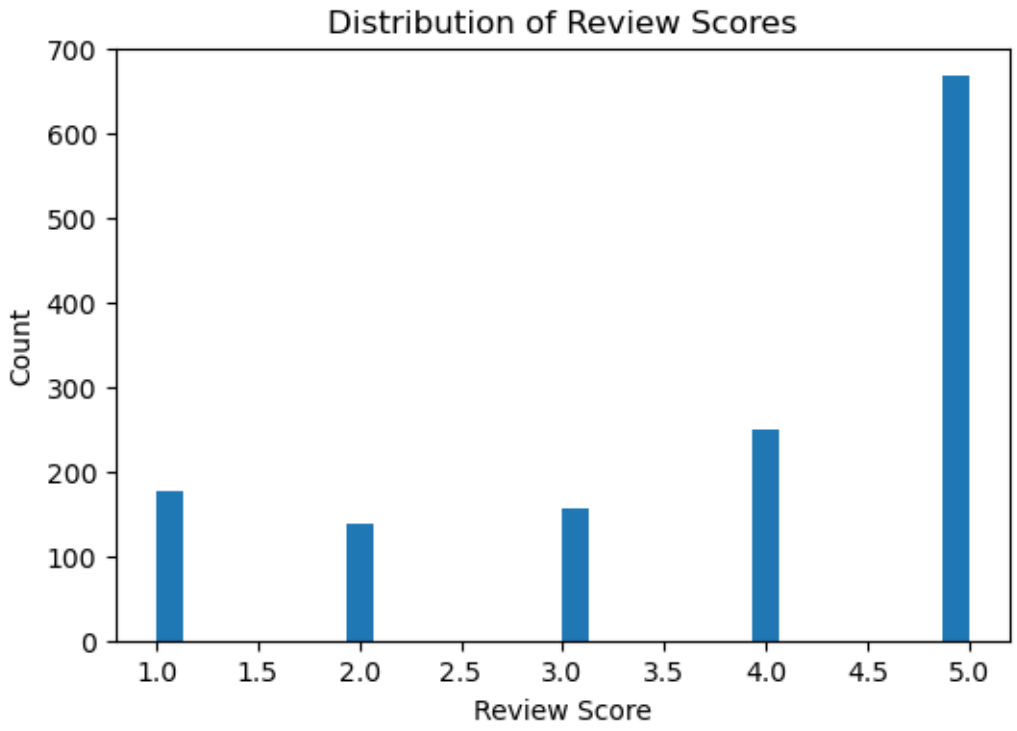
	Rating	Birth Year	Count
0	3.726994	1989.000000	326
1	3.799283	1961.376344	279
2	3.627027	1945.740541	185
3	3.955182	1999.921569	357
4	3.733333	1976.429167	240

After computing the average review score across the entire cluster, we can see that there is a trend in these clusters that as the mean birth year of clusters get later, there are more users in the cluster and the mean rating tends to be higher in the cluster, but still some minor downward trends in the rating as years get later.

1.3 Popularity matching

1.3.1 5

```
Most highly rated restaurant: Barn Steakhouse
Average review score: 3.79
Median review score: 4.00
```



From the histogram of the review score, we can see that the number of reviews with a review score 5 is the most. And the rating are all of integer values ranging from 1 to 5 as shown above.

1.3.2 6

```
Restaurant with the largest quantity of reviews: Campagnola
The median number of reviews received: 23.00
```

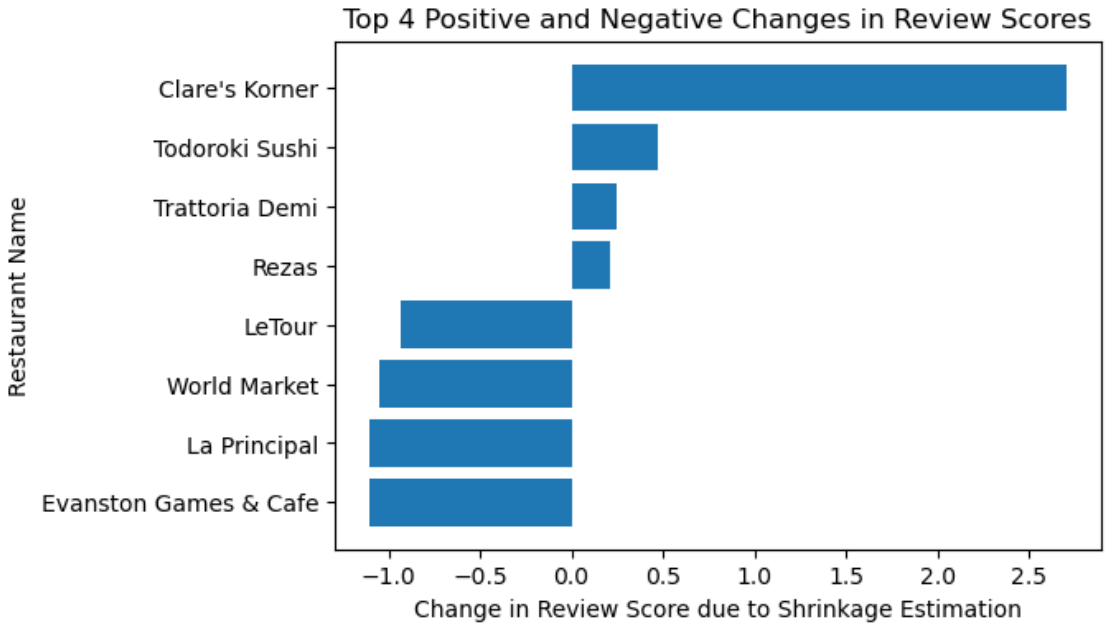
1.3.3 7

```
These are recommended restaurants for Spanish (in descending order): ['Tapas
Barcelona', '5411 Empanadas']
These are recommended restaurants for Chinese (in descending order): ['Joy Yee
Noodle', 'Peppercorns Kitchen', 'Lao Sze Chuan']
These are recommended restaurants for Mexican (in descending order): ['Fonda
Cantina', 'Zentli', 'Taco Diablo', 'Fridas', 'Chipotle']
These are recommended restaurants for Coffee (in descending order): ['Evanston
Games & Cafe', 'Philz Coffee', 'Brothers K Coffeehouse', 'Pâtisserie Coralie']
```

I have made the recommendation engine based on the mean vales of rating for each restaurant. A user can input a cusine type and receive a list of reccommendation in descending order based on popularity score.

1.3.4 8

```
Index(['Clare's Korner', 'Todoroki Sushi', 'Trattoria Demi', 'Rezas'],
dtype='object', name='Restaurant Name')
Index(['Evanston Games & Cafe', 'La Principal', 'World Market', 'LeTour'],
dtype='object', name='Restaurant Name')
```



I have implement a shrinkage estimator that shrinks reviews back towards the mean score, scaled by the number of reviews a restaurant has received from lecture slides right here. Top 4 restaurants that benefits the most from this shrinkage estimation are 'Clare's Korner', 'Todoroki Sushi', 'Trattoria Demi', 'Rezas'; top 4 restaurants hurt the most by it are 'Evanston Games & Cafe', 'La Principal', 'World Market', 'LeTour'. Also, I have made a plot that demonstrates changes in review scores due to shrinkage estimation for the top 4-positive and negative changes in a bar chart. We can see from the plot above that the restaurant 'Clare's Korner' is affected the most after shrinking and its rating is improved by this, while other restaurants does not seriously benefit from this. Also, there are more restaurant fairly negtively affected by shrinking, and they are affected a lot.

1.4 Content based filtering

1.4.1 9

```
[[0.         2.00000079 7.141429   … 2.45007571 2.44963616 7.28011246]
 [2.00000079 0.         7.28011037 … 2.45008167 2.44962076 7.28011211]
 [7.141429   7.28011037 0.         … 7.41637534 7.41623844 2.00000267]
 …
 [2.45007571 2.45008167 7.41637534 … 0.         2.00057594 7.41635632]
 [2.44963616 2.44962076 7.41623844 … 2.00057594 0.         7.41623089]
 [7.28011246 7.28011211 2.00000267 … 7.41635632 7.41623089 0.        ]]
```

```
/Users/ellenh/anaconda3/lib/python3.10/site-
packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

There are 4 columns of categorical variables in the sheet restaurants, so I have used one-hot encodings to deal with these 4 columns. And then I have computed the euclidean distance between every restaurant and printed the distance matrix above, but it takes spaces to display with restaurant names so I have formed one in code but display the one without restaurant name here.

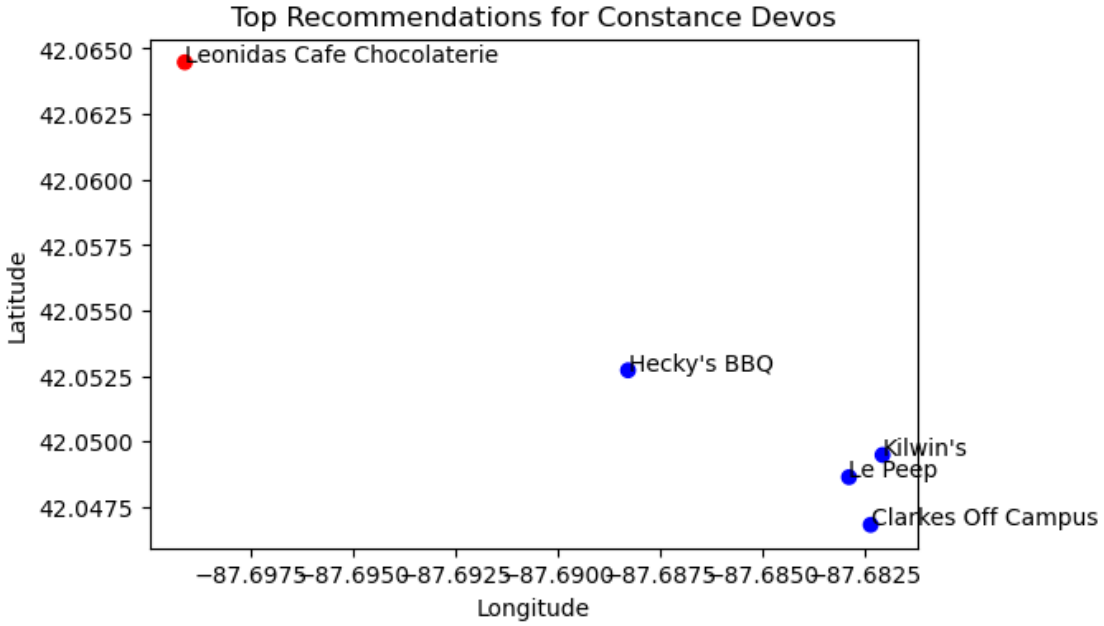
1.4.2 10

```
[[0.         0.00020287 0.00254742 … 0.00030428 0.00030423 0.00265035]
 [0.00020287 0.         0.0026499  … 0.00030428 0.00030423 0.00265018]
 [0.00254742 0.0026499  0.         … 0.00274755 0.00274898 0.00020772]
 …
 [0.00030428 0.00030428 0.00274755 … 0.         0.00020286 0.00274782]
 [0.00030423 0.00030423 0.00274898 … 0.00020286 0.         0.00274926]
 [0.00265035 0.00265018 0.00020772 … 0.00274782 0.00274926 0.        ]]
```

Silimar to the process above, this time I have computed the cosine distance between every restaurant and printed the distance matrix above, but it takes spaces to display with restaurant names so I have formed one in code but display the one without restaurant name here.

1.4.3 11

```
['Leonidas Cafe Chocolaterie', "Hecky's BBQ", 'Le Peep', "Kilwin's", 'Clarkes
Off Campus']
```



I have written a function that takes a user and returns a reccommendation using content based filtering based on cosine distance above. And tried to find the recommended restaurants for the user 'Constance Devos', and get the list of recommended restaurant 'Leonidas Cafe Chocolaterie', "Hecky's BBQ", 'Le Peep', "Kilwin's", 'Clarkes Off Campus'. Also for Constance Devos, I have plotted the top reccommendations by the system.

1.5 Natural language analysis

1.5.1 12

I have augemented this description by attaching the restaurant's cusine type to the end of the description as a column 'Augmented Description'.

1.5.2 13

```
[[1.         0.1        0.14285714 … 0.21428571 0.125      0.25      ]
 [0.125      1.         0.21428571 … 0.14285714 0.1875     0.25      ]
 [0.125      0.15       1.         … 0.07142857 0.3125     0.125     ]
 …
 [0.1875     0.1        0.07142857 … 1.         0.0625     0.5       ]
 [0.125      0.15       0.35714286 … 0.07142857 1.         0.125     ]
 [0.125      0.1        0.07142857 … 0.28571429 0.0625     1.        ]]
```

The computed Jaccard matrix using the elements of Augemented Description is shown above, but same as before, it takes spaces to display with restaurant names so I have formed one in code but display the one without restaurant name here.

1.5.3 14

```
The restaurant with the highest TF-IDF score for the word cozy is: Taste of
Nepal
The restaurant with the highest TF-IDF score for the word chinese is: Lao Sze
Chuan
```

I have compute the TF-IDF score for each restaurant's Augmented Description above first and made a function for that. By this function, we can see that the restaurant with the highest TF-IDF score for the word 'cozy' is Taste of Nepal, for 'chinese' is Lao Sze Chuan.

1.5.4 15

```
['known', 'for', 'in', 'and', 'with', 'chain', 'plus', 'space', 'american',
'restaurant', 'setting', 'coffee', 'fare', 'food', 'italian', 'of', 'burgers',
'warm', 'asian', 'bowls', 'casual', 'dishes', 'eatery', 'including', 'lunch',
'mexican', 'served', 'small', 'traditional', 'an', 'cafe', 'chinese', 'classic',
'drinks', 'japanese', 'local', 'mediterranean', 'other', 'south', 'vegetarian',
'bar', 'beers', 'breakfast', 'bright', 'byob', 'cocktails', 'contemporary',
'cozy', 'cuisine', 'extensive', 'fries', 'is', 'joint', 'korean', 'laid',
'meats', 'menu', 'or', 'outpost', 'pizza', 'pizzas', 'popular', 'quaint',
'sandwiches', 'simple', 'stylish', 'sushi', 'tacos', 'to', 'wood', 'bubble',
```

```
'chocolate', 'french', 'fresh', 'goods', 'grub', 'hot', 'ingredients',
'intimate', 'it', 'plates', 'prepared', 'ramen', 'relaxed', 'spanish', 'spot',
'storefront', 'such', 'takeout', 'tea', 'thai', 'the', 'upscale', 'whimsical',
'wine', 'your', 'indian', 'irish', 'jamaican', 'vegan']
   Restaurant Name  known  for   in       and  with  chain  plus  space  \
0  Tapas Barcelona    0.0  0.0  0.0  0.000000   0.0    0.0   0.0    0.0
1    Lao Sze Chuan    0.0  0.0  0.0  0.263346   0.0    0.0   0.0    0.0
2   5411 Empanadas    0.0  0.0  0.0  0.000000   0.0    0.0   0.0    0.0

   american  …  thai  the  upscale  whimsical  wine  your  indian  irish  \
0  0.260395  …   0.0  0.0      0.0        0.0   0.0   0.0     0.0    0.0
1  0.000000  …   0.0  0.0      0.0        0.0   0.0   0.0     0.0    0.0
2  0.000000  …   0.0  0.0      0.0        0.0   0.0   0.0     0.0    0.0

   jamaican  vegan
0       0.0    0.0
1       0.0    0.0
2       0.0    0.0

[3 rows x 101 columns]
```

The list of the 100 most popular words in the Augmented Description column is shown above, and the computed TF-IDF score for that word as a 64 TF-IDF vectors of length 100, one for each restaurant is also formed, and I have displayed the first 3 restaurant above.

1.5.5 16

```
Restaurant Name  Tapas Barcelona  Lao Sze Chuan  5411 Empanadas  \
Restaurant Name
Tapas Barcelona         0.000000       0.840362        0.834028
Lao Sze Chuan           0.840362       0.000000        0.975756
5411 Empanadas          0.834028       0.975756        0.000000

Restaurant Name  Hokkaido Ramen  Tomo Japanese Street Food  \
Restaurant Name
Tapas Barcelona        0.988324                   0.964442
Lao Sze Chuan          0.989275                   0.991437
5411 Empanadas         0.986613                   0.989311

Restaurant Name  Kuni's Japanese Restaurant   Kansaku  Taco Diablo   Fridas  \
Restaurant Name
Tapas Barcelona                    0.830653  0.957596     0.988848  1.00000
Lao Sze Chuan                      0.886685  0.946900     0.989757  1.00000
5411 Empanadas                     0.972835  0.970692     0.987213  0.96643

Restaurant Name  Edzo's Burger Shop  …  Pâtisserie Coralie  \
Restaurant Name                      …
Tapas Barcelona            0.989877  …            0.990946
```

```
Lao Sze Chuan              0.976595  …           0.932107
5411 Empanadas             0.933922  …           0.900309

Restaurant Name  Cozy Noodles and Rice  Nakorn  Prairie Moon  Hecky's BBQ  \
Restaurant Name
Tapas Barcelona               0.891482  1.0000      0.989949     0.985592
Lao Sze Chuan                 0.977654  0.9448      0.951693     0.986766
5411 Empanadas                0.972106  1.0000      0.943416     0.983480

Restaurant Name  Soban Korea  Burger King  Sarah's Brick Oven  \
Restaurant Name
Tapas Barcelona     0.989090     0.982640            0.963974
Lao Sze Chuan       0.989979     0.984055            0.991324
5411 Empanadas      0.907567     0.980096            0.989170

Restaurant Name  Leonidas Cafe Chocolaterie  Evanston Games & Cafe
Restaurant Name
Tapas Barcelona                   0.990783               0.986872
Lao Sze Chuan                     0.978688               0.987942
5411 Empanadas                    0.973397               0.984948

[3 rows x 60 columns]
```

The first 3 rows of the TF-IDF distance matrix with the distance between the TF-IDF vectors for restaurants is shown above.

1.5.6 17

```
Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.seq_relationship.weight', 'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.weight', 'cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

                Tapas Barcelona  Lao Sze Chuan  5411 Empanadas  \
Tapas Barcelona        0.000000       0.015169        0.013951
Lao Sze Chuan          0.015169       0.000000        0.012449
5411 Empanadas         0.013951       0.012449        0.000000

                Hokkaido Ramen  Tomo Japanese Street Food  \
Tapas Barcelona       0.018831                   0.014805
Lao Sze Chuan         0.011399                   0.014911
```

```
5411 Empanadas         0.015158                   0.007110

                Kuni's Japanese Restaurant   Kansaku  Taco Diablo    Fridas  \
Tapas Barcelona                   0.023336  0.015831     0.011077  0.007787
Lao Sze Chuan                     0.010407  0.012575     0.012718  0.013041
5411 Empanadas                    0.023593  0.003776     0.003483  0.008620

                Edzo's Burger Shop  …  Pâtisserie Coralie  \
Tapas Barcelona            0.008817  …            0.011240
Lao Sze Chuan              0.012955  …            0.007956
5411 Empanadas             0.008013  …            0.016668

                Cozy Noodles and Rice    Nakorn  Prairie Moon  Hecky's BBQ  \
Tapas Barcelona               0.013105  0.010155      0.008740     0.015954
Lao Sze Chuan                 0.012939  0.025369      0.008228     0.020336
5411 Empanadas                0.013530  0.022065      0.008266     0.005125

                Soban Korea  Burger King  Sarah's Brick Oven  \
Tapas Barcelona    0.020201     0.015521            0.016169
Lao Sze Chuan      0.026944     0.014226            0.013827
5411 Empanadas     0.008354     0.004741            0.006008

                Leonidas Cafe Chocolaterie  Evanston Games & Cafe
Tapas Barcelona                    0.023723               0.018790
Lao Sze Chuan                      0.009274               0.019204
5411 Empanadas                     0.016647               0.007384

[3 rows x 60 columns]
```

The first 3 rows of the Embedding-Distance matrix with the distance between embedding vectors of restaurants is shown above.

1.5.7 18

```
Jaccard: 3.9656301851894824
TF-IDF: 3.819726211811076
Embedding-Distance: 3.8577545115602216
```

Based on matrices from question 13, 16, 17, I have tried to do k recommendations as the method mentioned in the lecture notes for restaurant 'Lao Sze Chuan' based on these matrices. Since the mean rating are all assigned as a value of 1 as they are greater than 3, I calculated the actual values without bounded here, and we can see that these mean scores are so close, so it is really hard to tell which one is better, but Jaccard is a little better than TF-IDF and Embedding-Distance for recommendation here.

1.6 Collaborative Filtering

1.6.1 19

```
        Reviewer Name                                        vector
0         Connie Neal                [0.0, 0.0, 234.0, 1.0, 2.0, 0.0]
1   Jacquelyn Rigatti  [0.0, 0.0, 202.65272591486183, 2.0, 2.0, 0.0]
12        Sarah Hardy                [0.0, 0.0, 243.0, 3.0, 1.0, 0.0]
13  Jennifer Armagost  [0.0, 0.0, 202.65272591486183, 2.0, 1.0, 0.0]
14       Ruth Waynick                [1.0, 0.0, 135.0, 2.0, 1.0, 0.0]
```

```
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['Has Children?'] = features['Has Children?'].apply(lambda x: 1 if x
== 'Yes' else 0)
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:1
0: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['Vegetarian?'] = features['Vegetarian?'].apply(lambda x: 1 if x ==
'Yes' else 0)
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:1
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['Preferred Mode of Transport'] = features['Preferred Mode of
Transport'].apply(lambda x: 1 if x == 'Car Owner' else (2 if x == 'On Foot' else
3))
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:1
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['Average Amount Spent'] = features['Average Amount
Spent'].apply(lambda x: 1 if x == 'Low' else (2 if x == 'Medium' else 3))
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:1
```

```
3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['Northwestern Student?'] = features['Northwestern
Student?'].apply(lambda x: 1 if x == 'Yes' else 0)
/var/folders/6m/72gsx1kn1cd5rj99dq1_44yh0000gn/T/ipykernel_65798/1769911310.py:1
7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  features['vector'] = features[feas].values.tolist()
```

Considering double counting, which means many reviewers have reviewed multiple restaurants, I have dropped all duplicated reviews from a same reviewers. Then I have formed a vector including numeric representations of traits 'Has Children?', 'Vegetarian?', 'Weight (lb)', 'Preferred Mode of Transport', 'Average Amount Spent', and 'Northwestern Student?' for each reviewer. Also, I have printed out the first 5 vectors above. Right here 1 in the first element means the reviewer has children, 0 no children; 1 in the second element means the reviewer is vegetarian, 0 not; the 3rd element represents the Weight of the reviewer; the 4th element represents the Preferred Mode of Transport, 1 if the review is a car owner, 2 if the reviewer prefers on foot, 3 if the reviewer prefers public transit; 1 in the 5th element means the reviewer's Average Amount Spent in the restaurant is low, 2 medium, and 3 high; 1 in the 5th element means the review is Northwestern student, 0 not.

1.6.2 20

```
Index(['Nakorn', 'Kabul House', 'Kuni's Japanese Restaurant', 'Zentli'],
dtype='object', name='Restaurant Name')
```

Based on the vectors from the previous step, I have written a function that takes a user and computes the distance from that user to every other user, and based on this function, I have created a reccommendation algorithm that takes a user andnumber k and outputs k reccommendation made by a similar user, include the user and the suggested reccommendations in your write-up. What is the distance between the user and the user that you used to make reccommendations?

1.6.3 21

```
                  Barn Steakhouse Brothers K Coffeehouse Clarkes Off Campus  \
Jacquelyn Rigatti             5.0                    4.0                5.0
Kim Hamilton                  5.0                    5.0                5.0
Katina Whelton                5.0                    5.0                5.0

                  Edzo's Burger Shop Elephant & Vine Graduate Homestead Room  \
Jacquelyn Rigatti                5.0             4.0                      2.0
```

```
Kim Hamilton                    5.0            5.0                    5.0
Katina Whelton                  5.0            5.0                    5.0

                 Lao Sze Chuan Panino's Pizzeria Philz Coffee  \
Jacquelyn Rigatti           3.0              2.0         4.0
Kim Hamilton                5.0              5.0         5.0
Katina Whelton              5.0              5.0         5.0

                 Mumbai Indian Grill  … Zentli Peppercorns Kitchen  \
Jacquelyn Rigatti                0.0  …    5.0                 0.0
Kim Hamilton                     1.0  …    5.0                 5.0
Katina Whelton                   0.0  …    5.0                 5.0

                 Prairie Moon Taste of Nepal Pâtisserie Coralie Le Peep  \
Jacquelyn Rigatti          4.0            3.0                0.0     5.0
Kim Hamilton               4.0            3.0                0.0     5.0
Katina Whelton             4.0            3.0                0.0     5.0

                 Kung Fu Tea Evanston Games & Cafe Soban Korea Clare's Korner
Jacquelyn Rigatti         4.0                   0.0         0.0            0.0
Kim Hamilton              4.0                   0.0         0.0            0.0
Katina Whelton            4.0                   0.0         0.0            0.0

[3 rows x 59 columns]
```

We have selected users who have given at least 4 reviews. Then I have formed a 64 dimensioanl vector where entry i is the user's review of restaurant i here. This vector also have many blank entries and I have filled them with the mean of their corresponding row, since the mean of each row is based on the rating habits of each reviewer. And the first 3 rows of the final vector is shown above.

1.6.4 22

```
Index(['Zentli', 'Trattoria Demi', 'Clare's Korner', 'Lao Sze Chuan',
       'Evanston Chicken Shack'],
      dtype='object', name='Restaurant Name')
```

I have computed the 64-dimensional review vector for every user above. Now I have written the function find_sim() that takes a user and finds other users with similar review vectors. Also, I have tried to find the similar user of the reviewer 'Jacquelyn Rigatti', and the similar users are 'James Gutierrez', 'Jennifer Richardson', 'Ione Tollison', 'Gloria Donato', 'George Varela'.

1.6.5 23

```
mean rating for restaurant in 20: 4.227667153818049
mean rating for restaurant in 22: 2.950432815266575
```

Based on the mean rating for 5 restaurants recommended by the top 5 similar user of these two algorithm for 'Nancy Hayes', we can see they the mean rating for these recommended restaurants

in 20 is much higher than that in 22. Thus, we can see that the quality of recommendations made based on demographic data matrix is much better than user's review here.

1.7 Predictive modeling

1.7.1 24

```
[2.90163765]
error:  [0.90163765]
```

The linear model that takes demographic data, along with the cusine type for a restaurant is fitted, and I have tried to predict the restaurant score provided by a single non-vegetarian reviewer born in 1998 for American cuisine, which corresponds to the value in row 8 from the review sheet. The predicted rating for this is 2.90163765, while the actual rating is 2, so there is an error of 0.90163765.

1.7.2 25

```
RMSE: 1.3256164197966678
[2.7980957]
error:  [0.7980957]
```

```
/Users/ellenh/anaconda3/lib/python3.10/site-packages/sklearn/base.py:432:
UserWarning: X has feature names, but LinearRegression was fitted without
feature names
  warnings.warn(
```

Again, I have fitted the linear model using a train/test split in this problem. By testing with the test data, I have calculated the RMSE of the model 1.3256164197966678. Then I have tried to predict the restaurant score provided by a single non-vegetarian reviewer born in 1998 for American cuisine, which corresponds to the value in row 8 from the review sheet same as problem 24. The predicted rating for this is 2.7980957, while the actual rating is 2, so there is an error of 0.7980957, which is smaller than the model in the last problem, so it is more accurate. The reason for a better performance with train/split can be taht it helps to avoid overfitting and allows us to assess the generalization performance of the model.

1.7.3 26

```
RMSE (Lasso): 1.3680825052091492

Coefficients (Lasso):
Birth Year 0.0035802838836107346
Marital Status_Married -0.0
Marital Status_SIngle -0.0
Marital Status_Single 0.0
Marital Status_Widow -0.0
Vegetarian?_No -0.0
Vegetarian?_Yes 0.0
Vegetarian?_missing 0.0
Cuisine_American -0.0
Cuisine_BBQ -0.0
Cuisine_Breakfast 0.0
```

```
Cuisine_Brewery 0.0
Cuisine_Bubble Tea 0.0
Cuisine_Burgers -0.0
Cuisine_Chinese -0.0
Cuisine_Chocolate 0.0
Cuisine_Coffee 0.0
Cuisine_Irish 0.0
Cuisine_Italian 0.0
Cuisine_Jamaican 0.0
Cuisine_Japanese 0.0
Cuisine_Korean 0.0
Cuisine_Mediterranean -0.0
Cuisine_Mexican -0.0
Cuisine_Pizza 0.0
Cuisine_Salad 0.0
Cuisine_South Asian -0.0
Cuisine_Spanish 0.0
Cuisine_Thai -0.0
Cuisine_Vegetarian 0.0
Cuisine_nan 0.0
```

The RMSE based on this linear regression model with L1 penalty is 1.3680825052091492, which is larger than the RMSE 1.3256164197966678 of the standard linear regression model in question 25, shows the standard linear regression model is better and more accurate, but there is no much difference. In this model, we can see that only the "Birth Year" feature has a non-zero coefficient, indicating that it is the only feature that is selected on by this L1 model and it is predictive of review score and its weight of the linear model is large. The other features have a coefficient of 0, indicating that they are not predictive of review score.

1.7.4 27

```
RMSE: 1.2310567807320958
```

Considering the column 'Review Text', I have Embed the review text into a vector with TF-IDF and used this vector to predict the review score with a linear model fitted above. And based on this model, I used train/split to get the RMSE of the model 1.2310567807320958, which is smaller than both of these linear models fitted above, so it is more accurate.

1.7.5 28

```
/Users/ellenh/anaconda3/lib/python3.10/site-packages/sklearn/base.py:432:
UserWarning: X has feature names, but LinearRegression was fitted without
feature names
  warnings.warn(

[1.23463498]
error:  [-0.76536502]
```

I have included the vector for the embedded review text from step 27 here to fit a new model with demographic data, review texts, along with the cusine type for a restaurant. Compare to RMSE

1.3256164197966678 of the model in step 24 and 25, we get an smaller RMSE of 1.23463498 in this model with the embedded review text, so we can say that including the embdeed review text improves the predictive power of the model.

1.7.6 29

```
Columns with missing values: Index(['Marital Status', 'Vegetarian?'],
dtype='object')
[4.57204625]
error:  [0.57204625]
```

Finally, I have used demographic features to predict coffee scores for 3 coffee shops 'Philz Coffee', 'Brothers K Coffeehouse', 'Evanston Games & Cafe' in the dataset. There is actually one more marked as Coffee cuisine but more like a bakery actually, so I haved excluded it here. Thus, for these three restaurants only, I have fitted a linear model that takes demographic data and predicts the score here. And I have tried to predict for the restaurant score provided by a single non-vegetarian reviewer born in 1998 with this model and get a predicted score of 4.57204625, with an error of 0.57204625, which is relatively small.

1.7.7 30

```
Coefficients:
Birth Year 0.006852187196188132
Marital Status_Married -0.06693190993875936
Marital Status_Single -0.2619855555569427
Marital Status_nan 0.32891746549570305
Vegetarian?_No 0.02323104244849224
Vegetarian?_nan -0.023231042448492157
```

The weights produced by the linear model in step 29 is shown above, and we can see that higher birth year and non-vegetarian food items are associated with higher review scores, and reviewers with missing marital status tend to have higher review scores compared to those who are married or single. Also, missing values for vegetarian status are associated with lower review scores. To sum up, all three of these domographical features are selected on, and the weighs of Birth Year, unknown Marital Status, and vegetarian are positive, but married or single, and non-vegetarian weighs negative. Thus, older, unknown-marital-status, vegetarian like coffee more than others based on the model.

1.8 Final

1.8.1 31

```
Coefficients:
Birth Year 0.0038569933579799633
Marital Status_Married 0.11548446311717343
Marital Status_SIngle 0.01522400971499082
Marital Status_Single 0.1314252294815861
Marital Status_Widow -0.26213370231375055
Vegetarian?_No -0.5670259487724395
Vegetarian?_Yes 0.44651103351367527
```

```
Vegetarian?_missing 0.12051491525876505
Northwestern Student?_No 0.018757384682734843
Northwestern Student?_Yes -0.018757384682734954
```

In this part, I have tried to fit a model with 'Northwestern Student?' to predict the rating of the review. For the model coefficients above, we can see that whether the reviewer is a northwestern student or not a northwestern student weighs almost the same but weighs opposite in the model, so whether a reviewer is a northwestern student almost affect the rating for this model. Northwestern students tend to grade lower than non-Northwestern students, which is interesting.