CIND 820: Literature Review, Data Description, and Approach

By: Daria Yip

Ryerson University

Table of Contents

Abstract

In the past few years, with the advent of the Internet and the rise of instant gratification, companies who have been able to provide services on-demand have been the greatest winners. One of the companies that have led the transition to digital media consumption is Netflix. During the pandemic, they achieved even greater success through a significant increase in subscriptions. One of the technologies that streaming platforms like Netflix and its peers (i.e. Disney, Crave, Hulu) share is recommendation systems.

This paper looks at the logic behind recommendation systems and applying it to the Yelp dataset to provide personalized recommendations to the user based on data such as their reviews, ratings, location, and more. The dataset provides the information in .json form, and provides five files with information on users, businesses, reviews, check-ins, and tips. The sixth part of the dataset is in .jpg format, and will be disregarded for the purpose of this paper. Two common candidate generation approaches include content-based filtering, which is item-based filtering, and collaborative filtering, which uses user-based filtering (Google, 2022). The research questions that I will attempt to answer are: “For restaurant recommendations, do content-based filtering models or collaborative filtering models provide better suggestions?” and “What level of accuracy/performance can I achieve given these data points?”

To evaluate my models, I propose using the metric recall. As we are looking to create a Top-N recommendation list, recall is known to be effective in evaluating the ability of the model in finding relevant results (Cremonesi, P., & Turrin, R. (n.d.)). Currently, I am contemplating using the leave-one-out cross validation method (LOOCV), where it involves leaving a known rating to the side and later trying to predict it. Alternatives would be to use the k-fold cross-validation or the train-test split. As the Yelp dataset is very large, it may be computationally expensive to use k-fold cross-validation or the

LOOCV methods. However, I will attempt to use the k-fold cross validation first, and then use the LOOCV, then the train-test split respectively.

I will be using Python to code this project. Tools that I am proposing to use include Python's Pandas, Scikit-Learn, Numpy, Weka, and MatPlotLib libraries. With these tools, I will perform dimensionality reduction, data cleaning, exploratory data analysis, create training and test sets, and develop the content-based and collaborative. Then, I will evaluate this model with the aforementioned metric of recall.

Dataset Link

Dataset Link: https://www.yelp.com/dataset

GitHub Link

https://github.com/dariayip/Final-Capstone-Project (currently private)

References

Google. (2022, July 18). *Candidate Generation Overview | machine learning | google developers*. Google. Retrieved January 23, 2023, from https://developers.google.com/machine-learning/recommendation/overview/candidate-generation

Cremonesi, P., & Turrin, R. (n.d.). *An evaluation methodology for Collaborative Recommender Systems - polimi.it*. Retrieved January 24, 2023, from https://chrome.deib.polimi.it/images/8/8e/Cremonesi_2008_CROSSMEDIA.pdf

Literature Review

In a world where the burden of too many choices has become a widespread issue for consumers, recommender Systems (also known as RSs) have become a prevalent technology as a solution to this problem. RSs collect information on its users using explicit information such as ratings and reviews as well as implicit data such as monitoring users' behaviour and sites visited [1]. RSs have become applicable to many diverse industries, finding a home in areas such as music (Spotify), streaming entertainment (Netflix), and e-commerce (shopping sites), among others [1]. Not only do RSs help facilitate the decision-making process for users, they are also advantageous for businesses as they boost sales, particularly sales of niche items [3].

While recommendation systems have become more common as of recent years, RSs already have a history of more than 29 years [4]. The first collaborative filtering model was proposed in the 1990s [4]. The technology of recommendation systems can be divided into a data mining part that collects data about items and users, and a recommendation filtering model part [4]. The best type of filtering to use, collaborative or content-based, depends on the field [4]. Content-based filtering is where recommendations are generated based on similar items, whereas collaborative based filtering is where recommendations are generated based on similar users [4]. For instance, for online social network services (SNS) such as Yelp, Facebook, and Instagram, collaborative filtering models and hybrid recommendation models are common due to the availability of user data [4]. The vulnerabilities of collaborative filtering models include cold start (for new users), sparsity, and gray sheep (users with unique and difficult to predict tastes) [4]. In order to address these vulnerabilities, many studies have been conducted, such as Yang et al. who studied the use of a collaborative filtering model and the Matrix Factorization Technique for SNS recommendation systems [4]. Matrix factorization is where a user-item matrix will be split into two lower-ranked matrices, one for items and one for users to assist with prediction accuracy when there are issues with sparsity [4].

Content-based techniques develop their recommendations by analyzing an item’s attributes. This technique is best used when suggesting content such as web pages, publications, and news [5]. In order to find similarity between documents, it can use models such as the Naïve Bayes Classifier, Decision Trees, or Neural Networks [5]. One advantage of this technique is that it does not require the profile of other users as it does not generate predictions based on the profiles of similar users like collaborative filtering does [5]. One disadvantage is that it requires in-depth knowledge and description of the features of the items [5]. Within the recommendation system technology, recommendation techniques such as text mining, KNN (k-Nearest Neighbours), Clustering, Matrix Factorization, and Neural Network have been mainly utilized [4].

On the other hand, for collaborative filtering techniques, they have the ability to generate recommendations based on users with similar profiles enjoy, thus creating a “neighbourhood” [5]. The technique of collaborative filtering can be further broken down into memory-based and model-based categories [5]. Within memory-based techniques, one can further break down the categories into user-based techniques and item-based techniques [5]. User based collaborative filtering (UBCF) calculates the similarity between users based on their ratings on the same item, whereas item based collaborative filtering (IBCF) computes similarity between items and not users [5]. For model-based techniques, they utilize historic ratings for data mining or machine learning [5]. Examples include the Singular Value Decomposition (SVD) technique, and regression and clustering [5].

When comparing the performance of SVD, SVD++, and NMF (Non-negative matrix factorization) in a study, the RMSE of the SVD method demonstrated the lowest average error rate, while NMF had the largest value [2]. Using MAE as the performance metric, NMF maintained the highest value, but SVD++ showed the smallest value [2]. The results showed that using SVD for collaborative filtering in order to predict restaurant ratings was appropriate and efficient as it had the lowest RMSE and the second-lowest MAE [2]. The author continues to state that the combination of reduced data storage requirements and improved accuracy is a great justification for the use of SVD on the subject dataset [2].

The use of RMSE and MAE are popular as they are easy to compute and simple to understand [6]. However, the author states that the use of MAE for classification tasks is not meaningful [6]. The paper also talks about precision and recall in terms of classification metrics that are popular and appropriate for

Top-N models [6]. The author emphasizes that they should not be viewed on an absolute basis, but only on a comparative basis to compare different algorithms' results on the same dataset [6].

Data Description

The dataset is from the Yelp website, available for public download at https://www.yelp.com/dataset. At the time of download, the dataset contained 6,990,280 reviews; 150,346 businesses; 200,100 pictures; and 11 metropolitan areas. There were also 908,915 tips made by 1,987,897 users. Finally, there were also check-ins for each of the 131,930 businesses. The dataset is comprised of five .JSON files, named business.json, review.json, user.json, tip.json, and checkin.json.

Business.json

Within this .json file, there are 14 variables (Figure 2). Note, the variable is_open is a numeric variable displayed in Figure 1 as having a count and mean, amongst other variables, but it comprises of "0" and "1" in order to signify if the business is open ("1") or closed ("0"). Thus, the interquartile ranges as described below are not meaningful. There are five numeric variables, including latitude, longitude, stars, review_count, and is_open. Is_open may be better classified as binary. Contains location data, attributes, and categories for the business. According to the df.agg function, there are 150,346 records, and 14 attributes/columns.

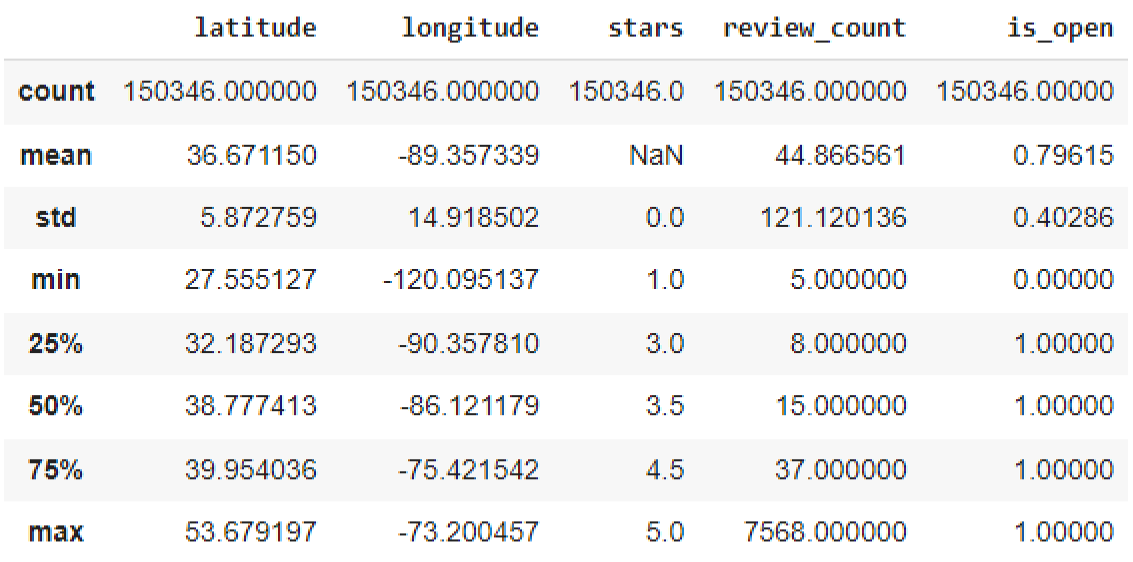
Latitude: Latitude of the business's location

Longitude: Longitude of the business's location

Stars: Out of 5 stars, with 5 being the highest rating and 1 being the lowest, the rating of the business

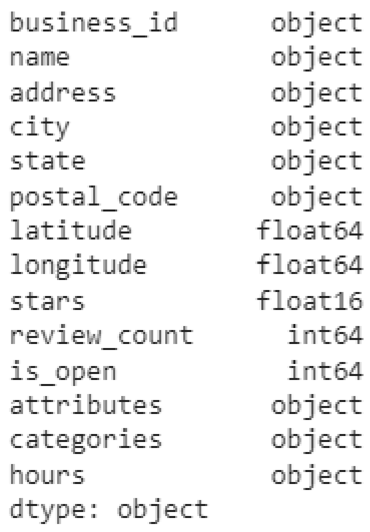
Review_Count: The amount of reviews the business has

Is_Open: "0" stands for closed, and "1" stands for open



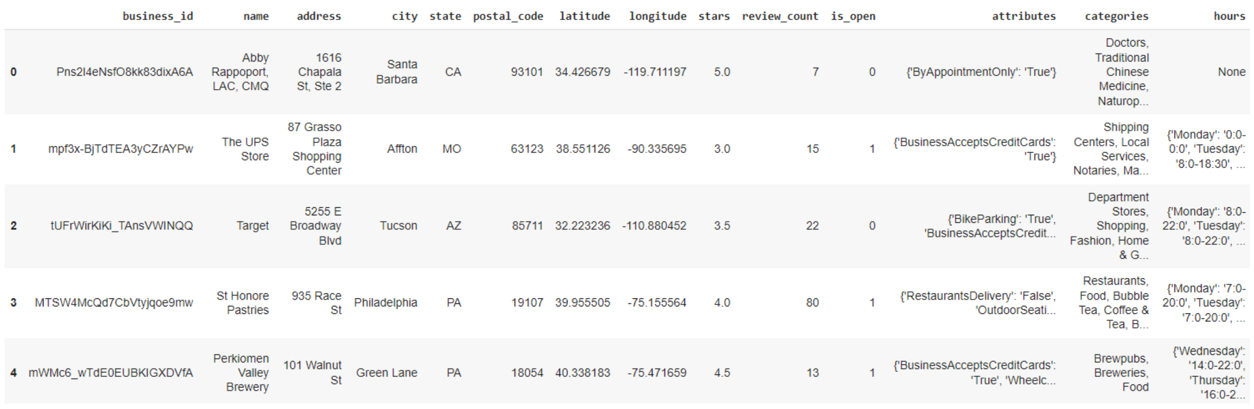
	latitude	longitude	stars	review_count	is_open
count	150346.000000	150346.000000	150346.0	150346.000000	150346.00000
mean	36.671150	-89.357339	NaN	44.866561	0.79615
std	5.872759	14.918502	0.0	121.120136	0.40286
min	27.555127	-120.095137	1.0	5.000000	0.00000
25%	32.187293	-90.357810	3.0	8.000000	1.00000
50%	38.777413	-86.121179	3.5	15.000000	1.00000
75%	39.954036	-75.421542	4.5	37.000000	1.00000
max	53.679197	-73.200457	5.0	7568.000000	1.00000

Figure 1: Summary statistics for the numeric attributes in business.json



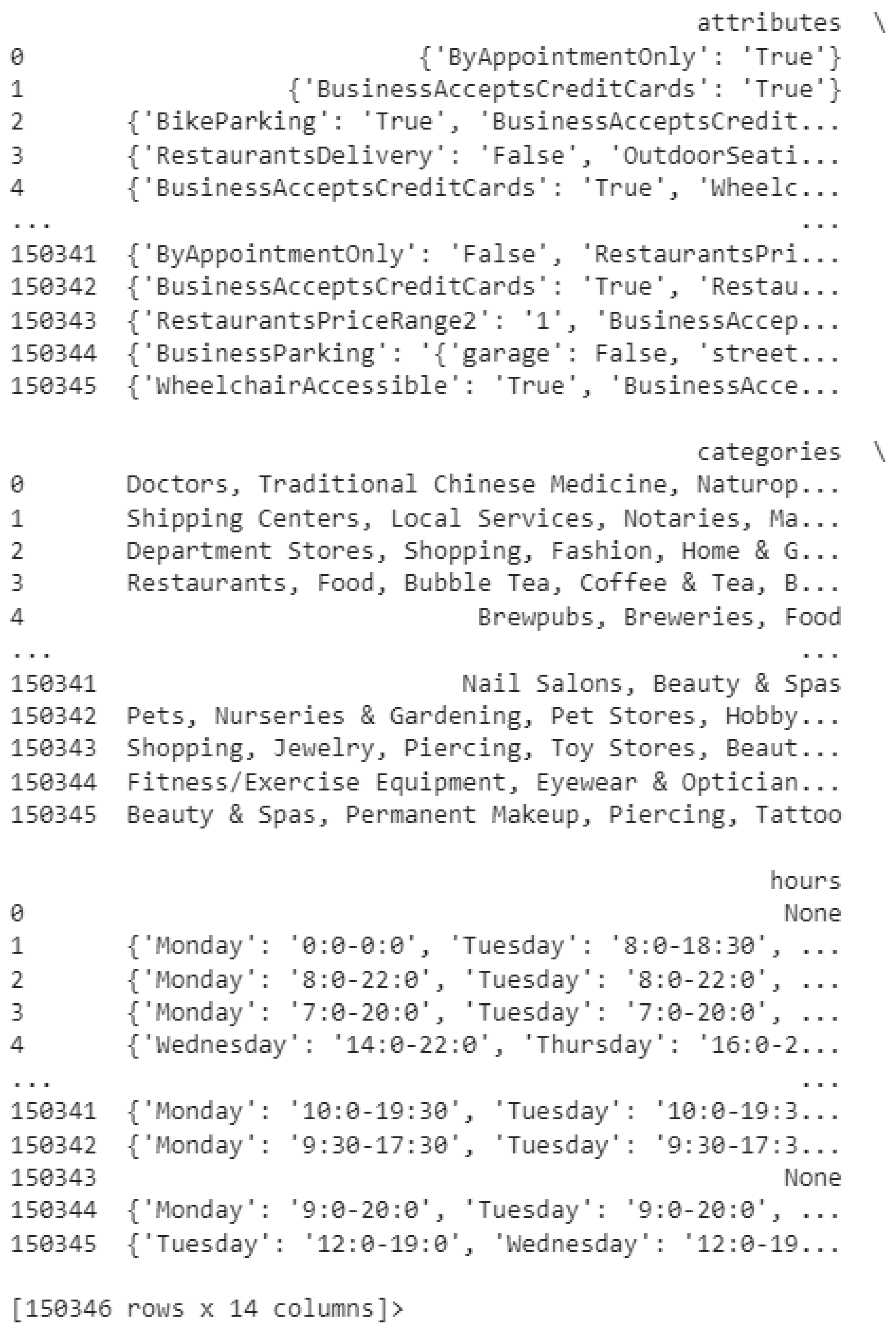
```
business_id      object
name             object
address          object
city             object
state            object
postal_code      object
latitude        float64
longitude       float64
stars           float16
review_count      int64
is_open           int64
attributes       object
categories       object
hours            object
dtype: object
```

Figure 2: Variables for business.json



	business_id	name	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open	attributes	categories	hours
0	Pns2l4eNsfO8kk83dixA6A	Abby Rappoport, LAC, CMQ	1616 Chapala St, Ste 2	Santa Barbara	CA	93101	34.426679	-119.711197	5.0	7	0	{'ByAppointmentOnly': 'True'}	Doctors, Traditional Chinese Medicine, Naturop...	None
1	mpf3x-BjTdTEA3yCZrAYPw	The UPS Store	87 Grasso Plaza Shopping Center	Affton	MO	63123	38.551126	-90.335695	3.0	15	1	{'BusinessAcceptsCreditCards': 'True'}	Shipping Centers, Local Services, Notaries, Ma...	{'Monday': '0:0-0:0', 'Tuesday': '8:0-18:30', ...
2	tUFrWirKiKi_TAnsVWINQQ	Target	5255 E Broadway Blvd	Tucson	AZ	85711	32.223236	-110.880452	3.5	22	0	{'BikeParking': 'True', 'BusinessAcceptsCredit...	Department Stores, Shopping, Fashion, Home & G...	{'Monday': '8:0-22:0', 'Tuesday': '8:0-22:0', ...
3	MTSW4McQd7CbVtyjqoe9mw	St Honore Pastries	935 Race St	Philadelphia	PA	19107	39.955505	-75.155564	4.0	80	1	{'RestaurantsDelivery': 'False', 'OutdoorSeati...	Restaurants, Food, Bubble Tea, Coffee & Tea, B...	{'Monday': '7:0-20:0', 'Tuesday': '7:0-20:0', ...
4	mWMc6_wTdE0EUBKIGXDVfA	Perkiomen Valley Brewery	101 Walnut St	Green Lane	PA	18054	40.338183	-75.471659	4.5	13	1	{'BusinessAcceptsCreditCards': 'True', 'Wheelc...	Brewpubs, Breweries, Food	{'Wednesday': '14:0-22:0', 'Thursday': '16:0-2...

Figure 3: First 5 entries within the business.json dataset



```
                                               attributes  \
0                         {'ByAppointmentOnly': 'True'}
1                 {'BusinessAcceptsCreditCards': 'True'}
2       {'BikeParking': 'True', 'BusinessAcceptsCredit...
3       {'RestaurantsDelivery': 'False', 'OutdoorSeati...
4       {'BusinessAcceptsCreditCards': 'True', 'Wheelc...
...                                                   ...
150341  {'ByAppointmentOnly': 'False', 'RestaurantsPri...
150342  {'BusinessAcceptsCreditCards': 'True', 'Restau...
150343  {'RestaurantsPriceRange2': '1', 'BusinessAccep...
150344  {'BusinessParking': '{'garage': False, 'street...
150345  {'WheelchairAccessible': 'True', 'BusinessAcce...

                                               categories  \
0       Doctors, Traditional Chinese Medicine, Naturop...
1       Shipping Centers, Local Services, Notaries, Ma...
2       Department Stores, Shopping, Fashion, Home & G...
3       Restaurants, Food, Bubble Tea, Coffee & Tea, B...
4                               Brewpubs, Breweries, Food
...                                                   ...
150341                              Nail Salons, Beauty & Spas
150342  Pets, Nurseries & Gardening, Pet Stores, Hobby...
150343  Shopping, Jewelry, Piercing, Toy Stores, Beaut...
150344  Fitness/Exercise Equipment, Eyewear & Optician...
150345  Beauty & Spas, Permanent Makeup, Piercing, Tattoo

                                                    hours
0                                                    None
1       {'Monday': '0:0-0:0', 'Tuesday': '8:0-18:30', ...
2       {'Monday': '8:0-22:0', 'Tuesday': '8:0-22:0', ...
3       {'Monday': '7:0-20:0', 'Tuesday': '7:0-20:0', ...
4       {'Wednesday': '14:0-22:0', 'Thursday': '16:0-2...
...                                                   ...
150341  {'Monday': '10:0-19:30', 'Tuesday': '10:0-19:3...
150342  {'Monday': '9:30-17:30', 'Tuesday': '9:30-17:3...
150343                                               None
150344  {'Monday': '9:0-20:0', 'Tuesday': '9:0-20:0', ...
150345  {'Tuesday': '12:0-19:0', 'Wednesday': '12:0-19...

[150346 rows x 14 columns]>
```

Figure 4: df.agg

Tip.json

Within this .json file, there are 5 variables (Figure 6). There is one numeric variable, compliment_count. According to the df.agg function, there are 908,915 records, and 5 attributes/columns. Contains tips written by a user for a business – these differ from reviews in that they are shorter and are there to convey fast suggestions.

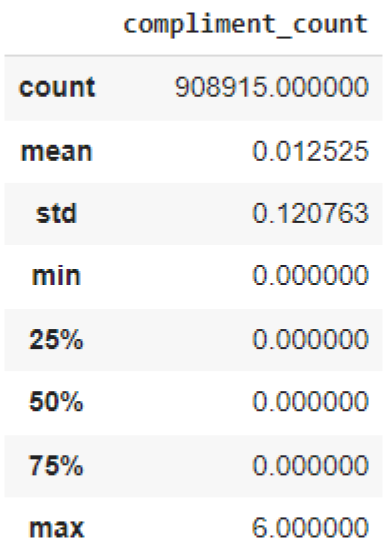
User_id: Refers to the user_id of the user who wrote the review

Business_id: Refers to the business_id of the respective business

Text: The full review text as written by the user

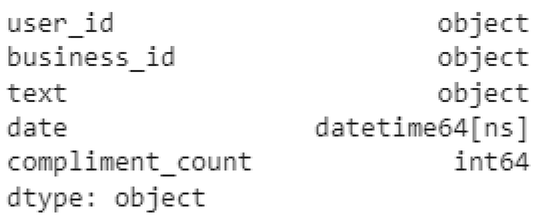
Date: Date Time format for the date and time the review was written, formatted as YYYY-MM-DD

Compliment_count: How many compliments it has



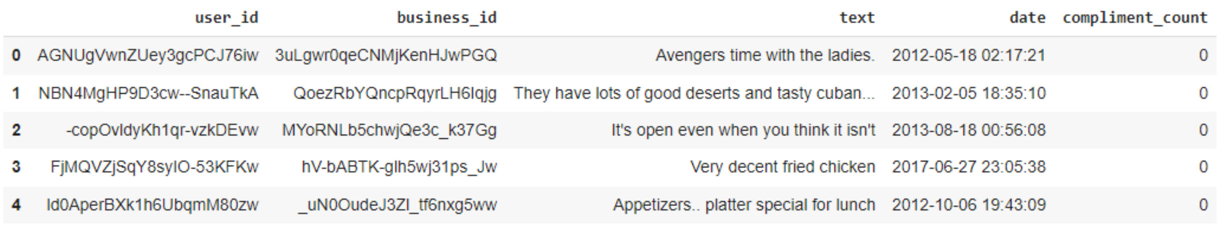
	compliment_count
count	908915.000000
mean	0.012525
std	0.120763
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	6.000000

Figure 5: Summary statistics for the numeric attributes in tip.json



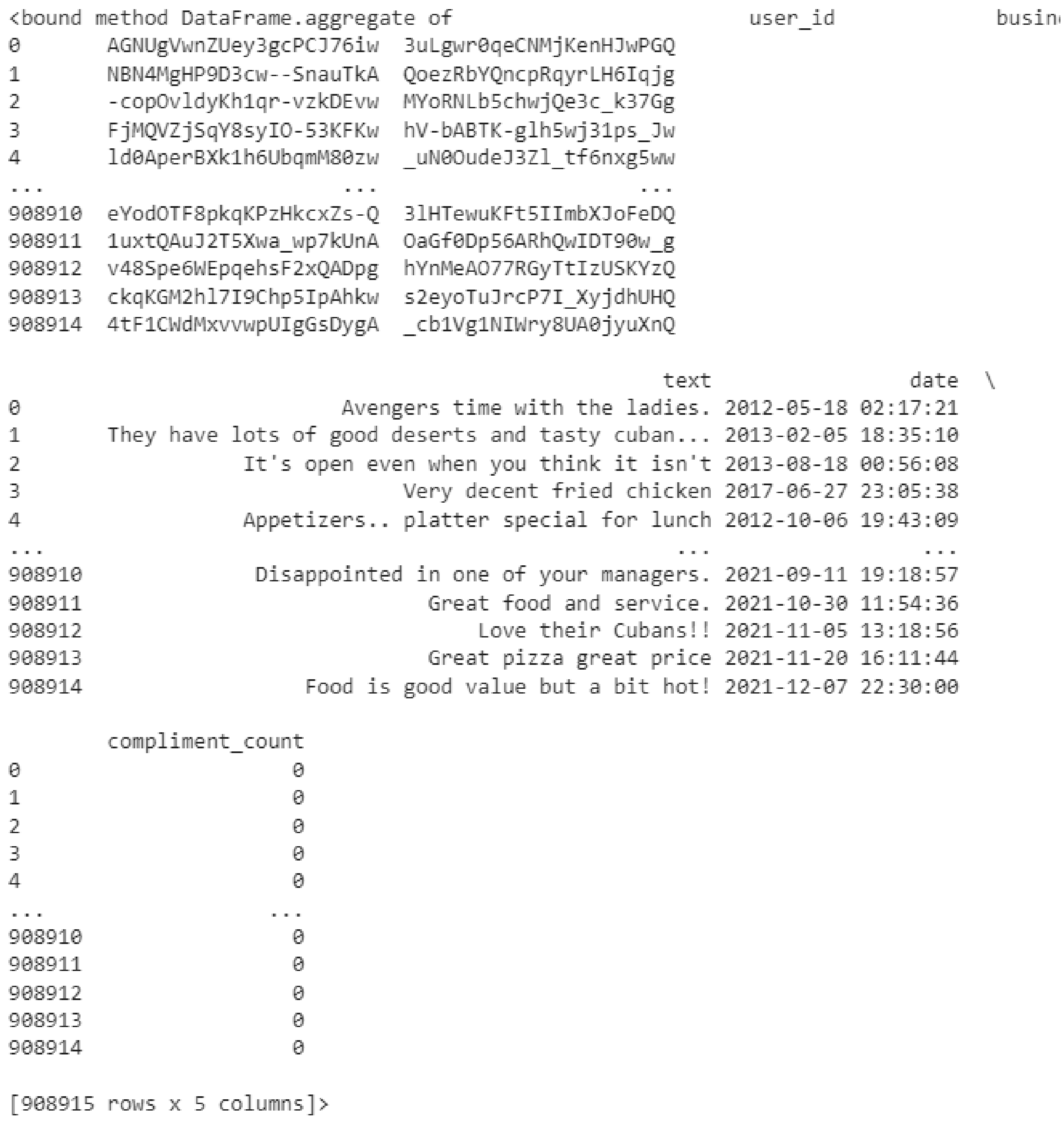
```
user_id                      object
business_id                  object
text                         object
date                 datetime64[ns]
compliment_count              int64
dtype: object
```

Figure 6: Variables for tip.json



	user_id	business_id	text	date	compliment_count
0	AGNUgVwnZUey3gcPCJ76iw	3uLgwr0qeCNMjKenHJwPGQ	Avengers time with the ladies.	2012-05-18 02:17:21	0
1	NBN4MgHP9D3cw--SnauTkA	QoezRbYQncpRqyrLH6Iqjg	They have lots of good deserts and tasty cuban...	2013-02-05 18:35:10	0
2	-copOvldyKh1qr-vzkDEvw	MYoRNLb5chwjQe3c_k37Gg	It's open even when you think it isn't	2013-08-18 00:56:08	0
3	FjMQVZjSqY8syIO-53KFKw	hV-bABTK-glh5wj31ps_Jw	Very decent fried chicken	2017-06-27 23:05:38	0
4	ld0AperBXk1h6UbqmM80zw	_uN0OudeJ3Zl_tf6nxg5ww	Appetizers.. platter special for lunch	2012-10-06 19:43:09	0

Figure 7: First 5 entries within the tip.json dataset



```
<bound method DataFrame.aggregate of                         user_id             busin
0       AGNUgVwnZUey3gcPCJ76iw  3uLgwr0qeCNMjKenHJwPGQ
1       NBN4MgHP9D3cw--SnauTkA  QoezRbYQncpRqyrLH6Iqjg
2       -copOvldyKh1qr-vzkDEvw  MYoRNLb5chwjQe3c_k37Gg
3       FjMQVZjSqY8syIO-53KFKw  hV-bABTK-glh5wj31ps_Jw
4       ld0AperBXk1h6UbqmM80zw  _uN0OudeJ3Zl_tf6nxg5ww
...                        ...                     ...
908910  eYodOTF8pkqKPzHkcxZs-Q  3lHTewuKFt5IImbXJoFeDQ
908911  1uxtQAuJ2T5Xwa_wp7kUnA  OaGf0Dp56ARhQwIDT90w_g
908912  v48Spe6WEpqehsF2xQADpg  hYnMeAO77RGyTtIzUSKYzQ
908913  ckqKGM2hl7I9Chp5IpAhkw  s2eyoTuJrcP7I_XyjdhUHQ
908914  4tF1CWdMxvvwpUIgGsDygA  _cb1Vg1NIWry8UA0jyuXnQ

                                                     text                 date  \
0                          Avengers time with the ladies. 2012-05-18 02:17:21
1       They have lots of good deserts and tasty cuban... 2013-02-05 18:35:10
2                  It's open even when you think it isn't 2013-08-18 00:56:08
3                               Very decent fried chicken 2017-06-27 23:05:38
4                  Appetizers.. platter special for lunch 2012-10-06 19:43:09
...                                                   ...                 ...
908910              Disappointed in one of your managers. 2021-09-11 19:18:57
908911                            Great food and service. 2021-10-30 11:54:36
908912                                Love their Cubans!! 2021-11-05 13:18:56
908913                            Great pizza great price 2021-11-20 16:11:44
908914                  Food is good value but a bit hot! 2021-12-07 22:30:00

        compliment_count
0                      0
1                      0
2                      0
3                      0
4                      0
...                  ...
908910                 0
908911                 0
908912                 0
908913                 0
908914                 0

[908915 rows x 5 columns]>
```

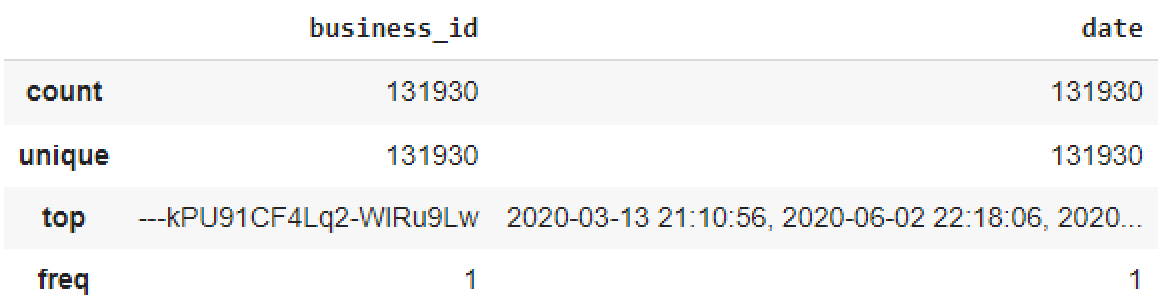
Figure 8: df.agg

Checkin.json

Within this .json file, there are 2 variables (Figure 10). There are no numeric variables, and only two string variables, business_id and date. According to the df.agg function, there are 131,930 records, and 2 attributes/columns. Contains the date and the business_id of the checkins.

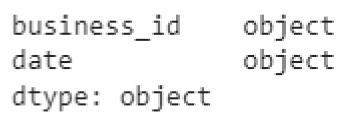
Business_id: Connected to business_id in business.json. A string variable representing the ID of the checked-in business

Date: Date of the check-in, a string variable, a comma-separated list of timestamps for each check-in in YYYY-MM-DD HH:MM:SS format



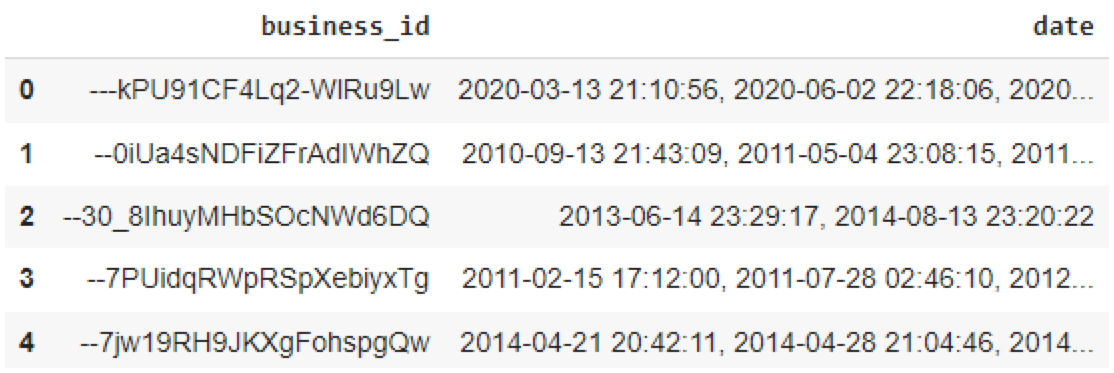
	business_id	date
count	131930	131930
unique	131930	131930
top	---kPU91CF4Lq2-WlRu9Lw	2020-03-13 21:10:56, 2020-06-02 22:18:06, 2020...
freq	1	1

Figure 9: Summary statistics for the attributes in checkin.json



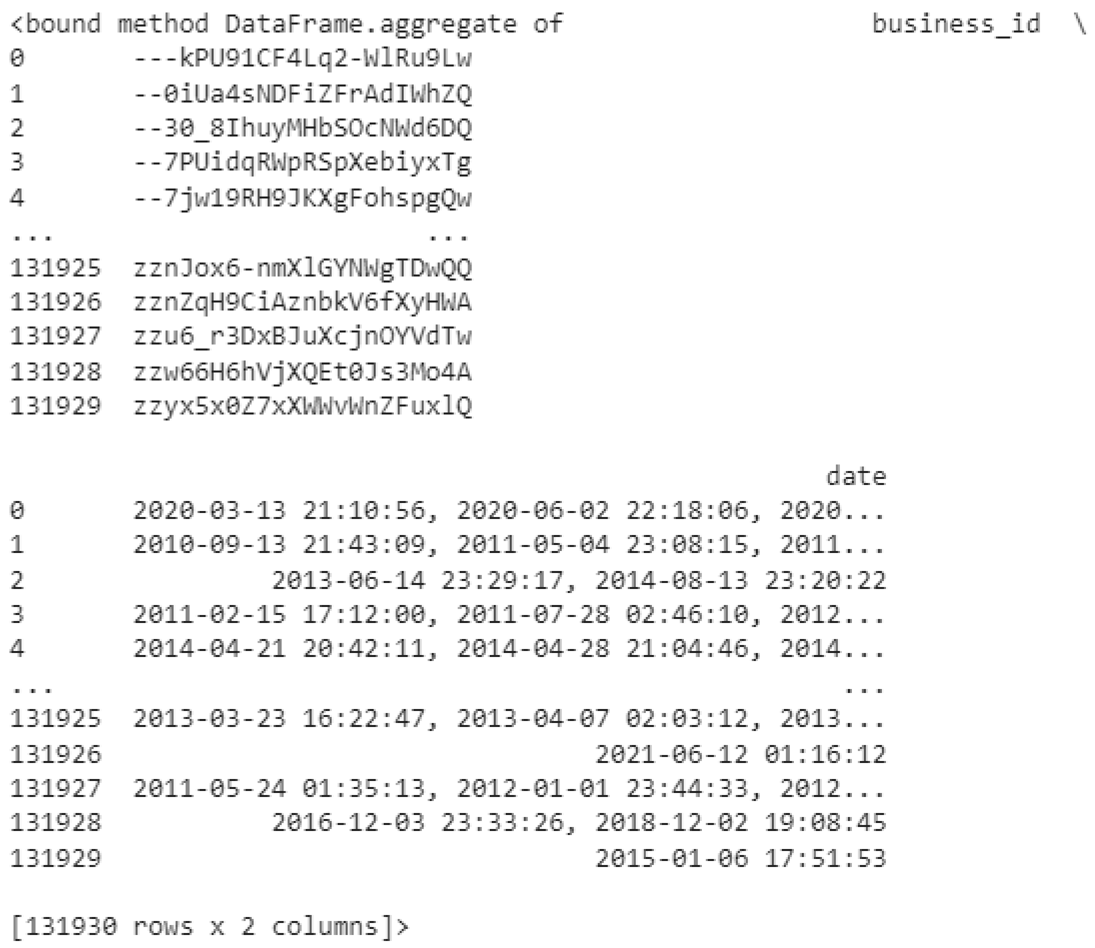
```
business_id    object
date           object
dtype: object
```

Figure 10: Variables for checkin.json



	business_id	date
0	---kPU91CF4Lq2-WlRu9Lw	2020-03-13 21:10:56, 2020-06-02 22:18:06, 2020...
1	--0iUa4sNDFiZFrAdIWhZQ	2010-09-13 21:43:09, 2011-05-04 23:08:15, 2011...
2	--30_8IhuyMHbSOcNWd6DQ	2013-06-14 23:29:17, 2014-08-13 23:20:22
3	--7PUidqRWpRSpXebiyxTg	2011-02-15 17:12:00, 2011-07-28 02:46:10, 2012...
4	--7jw19RH9JKXgFohspgQw	2014-04-21 20:42:11, 2014-04-28 21:04:46, 2014...

Figure 11: First 5 entries within the checkin.json dataset



```
<bound method DataFrame.aggregate of                   business_id  \
0       ---kPU91CF4Lq2-WlRu9Lw
1       --0iUa4sNDFiZFrAdIWhZQ
2       --30_8IhuyMHbSOcNWd6DQ
3       --7PUidqRWpRSpXebiyxTg
4       --7jw19RH9JKXgFohspgQw
...                        ...
131925  zznJox6-nmXlGYNWgTDwQQ
131926  zznZqH9CiAznbkV6fXyHWA
131927  zzu6_r3DxBJuXcjnOYVdTw
131928  zzw66H6hVjXQEt0Js3Mo4A
131929  zzyx5x0Z7xXWWvWnZFuxlQ

                                                      date
0       2020-03-13 21:10:56, 2020-06-02 22:18:06, 2020...
1       2010-09-13 21:43:09, 2011-05-04 23:08:15, 2011...
2                2013-06-14 23:29:17, 2014-08-13 23:20:22
3       2011-02-15 17:12:00, 2011-07-28 02:46:10, 2012...
4       2014-04-21 20:42:11, 2014-04-28 21:04:46, 2014...
...                                                   ...
131925  2013-03-23 16:22:47, 2013-04-07 02:03:12, 2013...
131926                                2021-06-12 01:16:12
131927  2011-05-24 01:35:13, 2012-01-01 23:44:33, 2012...
131928           2016-12-03 23:33:26, 2018-12-02 19:08:45
131929                                2015-01-06 17:51:53

[131930 rows x 2 columns]>
```

Figure 12: df.agg

User.json

Within this .json file, there are 22 variables (Figure 14). There are many numeric variables, such as the ones shown in Figure 13 (ie. Review_count). According to the df.agg function, there are 1,987,897 records, and 22 attributes/columns. User.json contains information and metadata about the user.

user_id: A string variable representing the ID of the user

name: A string variable representing the first name of the user

review_count: An integer that stands for the amount of reviews the user has written

Yelping_since: A string that holds a "YYYY-MM-DD" format date, representing when the user joined Yelp

Friends: An array of strings, representing the user's friends written as user_ids

Useful: An integer, the number of votes the user has sent that are "useful"

Funny: An integer, the number of votes the user has sent that are "funny"

Cool: An integer, the number of votes the user has sent that are "cool"

Fans: An integer, the amount of fans that the user has

Elite: An array of integers, contains the years that the user was "elite"

Average Stars: A float figure, representing the average rating of all reviews

Compliment_Hot: An integer representing the number of times the user received the compliment "hot"

Compliment_More: An integer representing the number of times the user has received the compliment "more"

Compliment_Profile: An integer representing the number of times the user has received a compliment on their profile

Compliment_Cute: An integer representing the number of times the user has received the compliment "cute"

Compliment_List: An integer representing the number of times the user has received a list compliment

Compliment_Note: An integer representing the number of note compliment received by the user

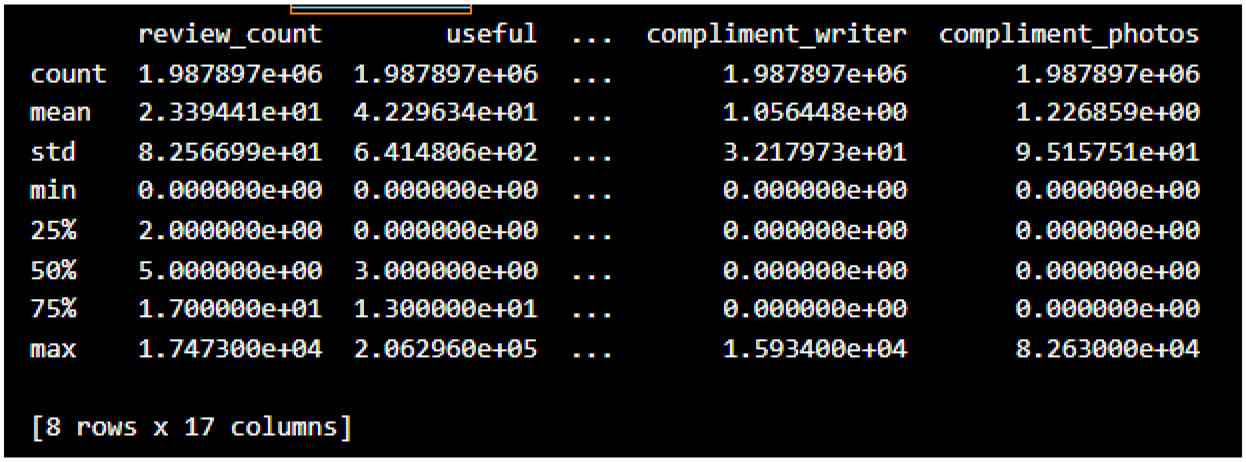
Compliment_Plain: An integer representing the number of times the user received "plain" compliments

Compliment_Cool: An integer representing the number of times the user received "cool" compliments

Compliment_Funny: An integer representing the number of times the user received "funny" compliments

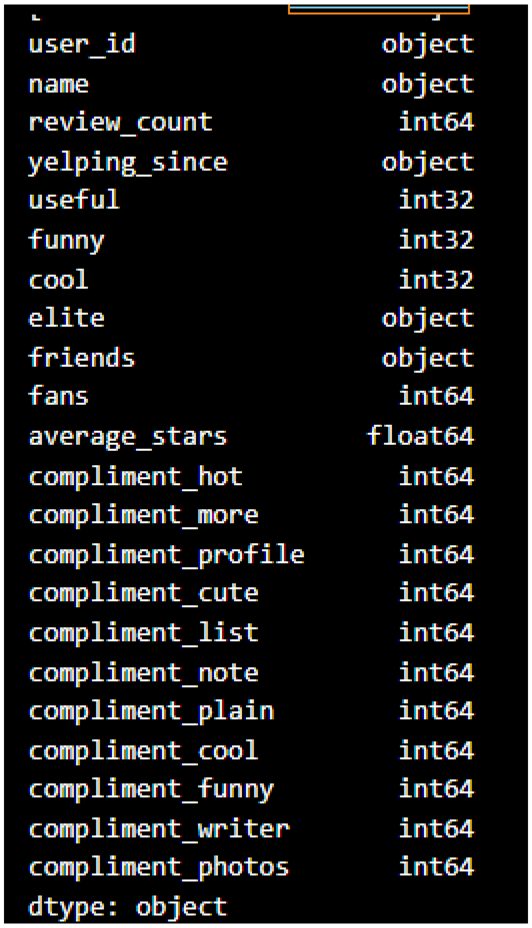
Compliment_Writer: An integer representing the number of times the user received "writer" compliments

Compliment_Photos: An integer representing the number of times the user received photo compliments



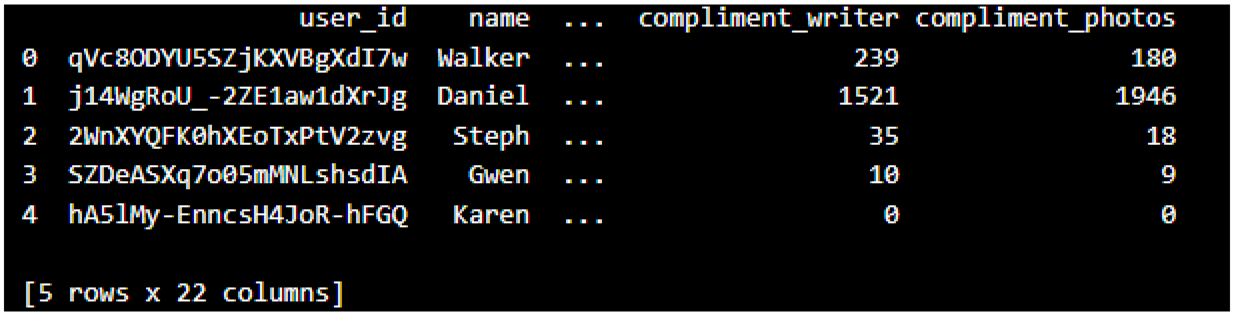
```
       review_count        useful  ...  compliment_writer  compliment_photos
count  1.987897e+06  1.987897e+06  ...       1.987897e+06       1.987897e+06
mean   2.339441e+01  4.229634e+01  ...       1.056448e+00       1.226859e+00
std    8.256699e+01  6.414806e+02  ...       3.217973e+01       9.515751e+01
min    0.000000e+00  0.000000e+00  ...       0.000000e+00       0.000000e+00
25%    2.000000e+00  0.000000e+00  ...       0.000000e+00       0.000000e+00
50%    5.000000e+00  3.000000e+00  ...       0.000000e+00       0.000000e+00
75%    1.700000e+01  1.300000e+01  ...       0.000000e+00       0.000000e+00
max    1.747300e+04  2.062960e+05  ...       1.593400e+04       8.263000e+04

[8 rows x 17 columns]
```

Figure 13: Summary statistics for the attributes in user.json



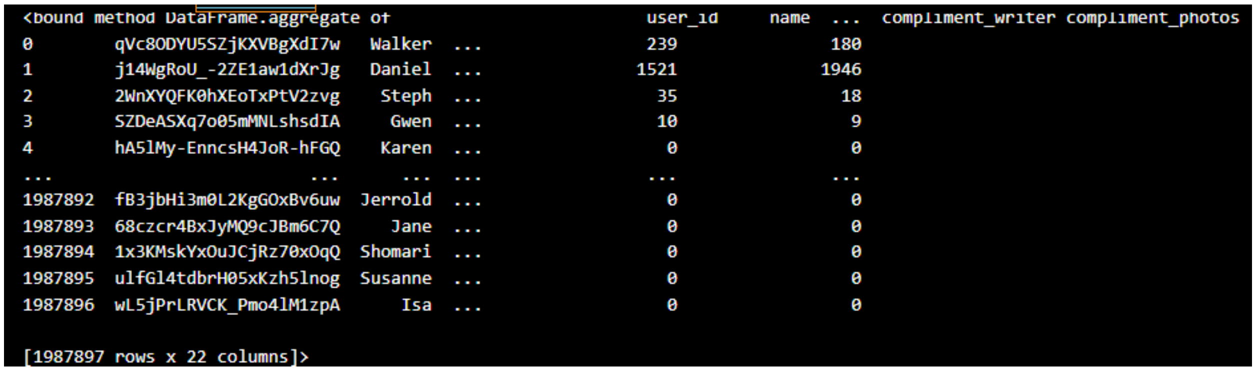
```
user_id                  object
name                     object
review_count              int64
yelping_since            object
useful                    int32
funny                     int32
cool                      int32
elite                    object
friends                  object
fans                      int64
average_stars           float64
compliment_hot            int64
compliment_more           int64
compliment_profile        int64
compliment_cute           int64
compliment_list           int64
compliment_note           int64
compliment_plain          int64
compliment_cool           int64
compliment_funny          int64
compliment_writer         int64
compliment_photos         int64
dtype: object
```

Figure 14: Variables for User.json



```
                  user_id    name  ...  compliment_writer compliment_photos
0  qVc8ODYU5SZjKXVBgXdI7w  Walker  ...                239               180
1  j14WgRoU_-2ZE1aw1dXrJg  Daniel  ...               1521              1946
2  2WnXYQFK0hXEoTxPtV2zvg   Steph  ...                 35                18
3  SZDeASXq7o05mMNLshsdIA    Gwen  ...                 10                 9
4  hA5lMy-EnncsH4JoR-hFGQ   Karen  ...                  0                 0

[5 rows x 22 columns]
```

Figure 15: First 5 entries within the user.json dataset



```
<bound method DataFrame.aggregate of                       user_id     name  ...  compliment_writer compliment_photos
0        qVc8ODYU5SZjKXVBgXdI7w   Walker  ...                239               180
1        j14WgRoU_-2ZE1aw1dXrJg   Daniel  ...               1521              1946
2        2WnXYQFK0hXEoTxPtV2zvg    Steph  ...                 35                18
3        SZDeASXq7o05mMNLshsdIA     Gwen  ...                 10                 9
4        hA5lMy-EnncsH4JoR-hFGQ    Karen  ...                  0                 0
...                         ...      ...  ...                ...               ...
1987892  fB3jbHi3m0L2KgGOxBv6uw  Jerrold  ...                  0                 0
1987893  68czcr4BxJyMQ9cJBm6C7Q     Jane  ...                  0                 0
1987894  1x3KMskYxOuJCjRz70xOqQ  Shomari  ...                  0                 0
1987895  ulfGl4tdbrH05xKzh5lnog  Susanne  ...                  0                 0
1987896  wL5jPrLRVCK_Pmo4lM1zpA      Isa  ...                  0                 0

[1987897 rows x 22 columns]>
```

Figure 16: df.agg

Review.json

Within this .json file, there are 9 variables (Figure 18). There are four numeric variables, such as the ones shown in Figure 17 (stars, useful, funny, and cool). According to the df.agg function, there are 6,990,280 records, and 9 attributes/columns. Review.json contains metadata about the review including the full text of the review, and contains information such as relevant user_id and business_id the review is written for.

Review_id: A string representing the unique ID of the review

User_id: Maps to the user_id in the user.json file. A string unique to the user who wrote the review

Business_id: Maps to the business_id in the business.json file. A string unique to the business who the review is written for

Stars: An integer, the star rating in the review

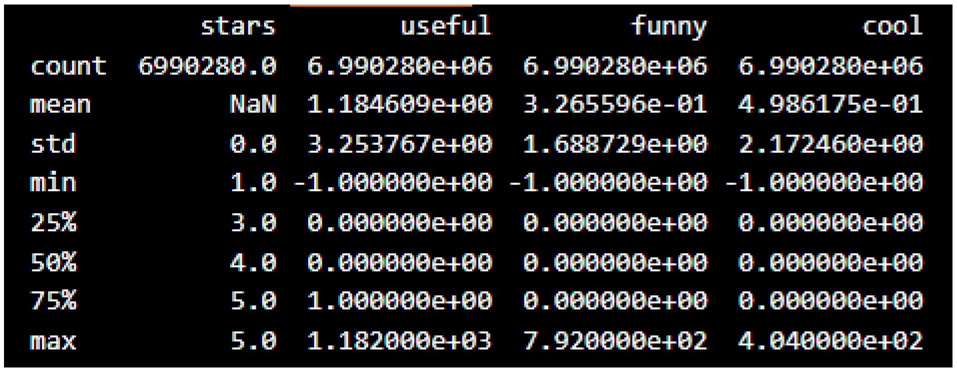
Date: A string, formatted as "YYYY-MM-DD" to represent the date the review was posted

Text: A string, the review itself

Useful: An integer representing the number of votes for "useful" the review received

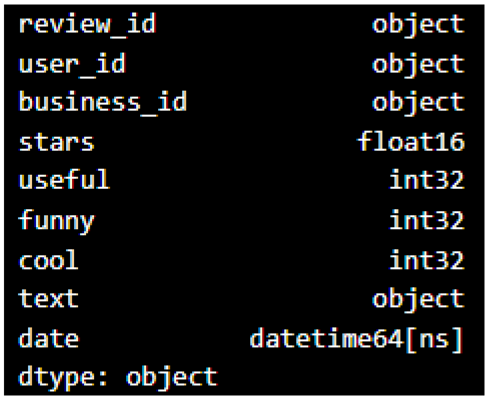
Funny: An integer representing the number of votes for "funny" the review received

Cool: An integer representing the number of votes for "cool" the review received



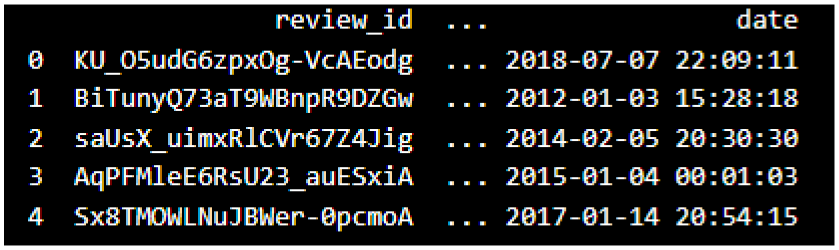
```
           stars        useful         funny          cool
count  6990280.0  6.990280e+06  6.990280e+06  6.990280e+06
mean         NaN  1.184609e+00  3.265596e-01  4.986175e-01
std          0.0  3.253767e+00  1.688729e+00  2.172460e+00
min          1.0 -1.000000e+00 -1.000000e+00 -1.000000e+00
25%          3.0  0.000000e+00  0.000000e+00  0.000000e+00
50%          4.0  0.000000e+00  0.000000e+00  0.000000e+00
75%          5.0  1.000000e+00  0.000000e+00  0.000000e+00
max          5.0  1.182000e+03  7.920000e+02  4.040000e+02
```

Figure 17: Summary Statistics for the attributes in review.json



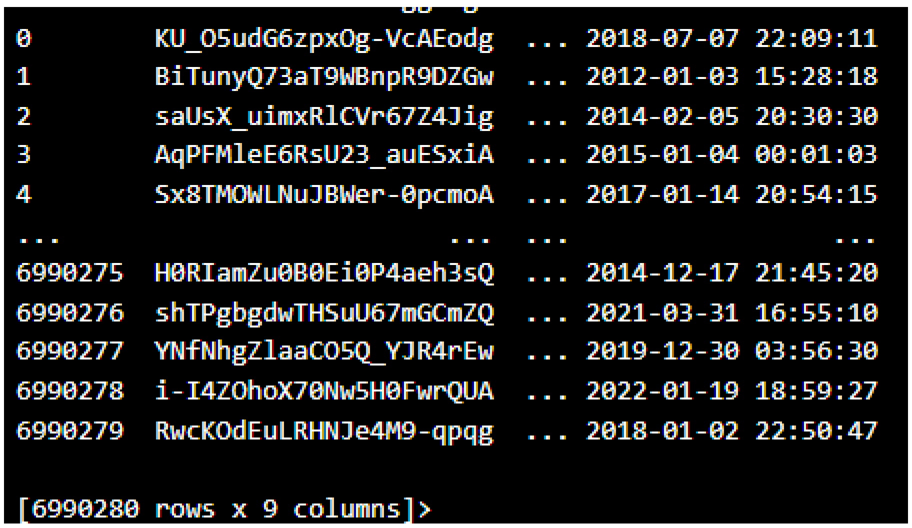
```
review_id               object
user_id                 object
business_id             object
stars                  float16
useful                   int32
funny                    int32
cool                     int32
text                    object
date            datetime64[ns]
dtype: object
```

Figure 18: Variables for review.json



```
                review_id  ...                date
0  KU_O5udG6zpxOg-VcAEodg  ... 2018-07-07 22:09:11
1  BiTunyQ73aT9WBnpR9DZGw  ... 2012-01-03 15:28:18
2  saUsX_uimxRlCVr67Z4Jig  ... 2014-02-05 20:30:30
3  AqPFMleE6RsU23_auESxiA  ... 2015-01-04 00:01:03
4  Sx8TMOWLNuJBWer-0pcmoA  ... 2017-01-14 20:54:15
```

Figure 19: The first 5 entries in review.json



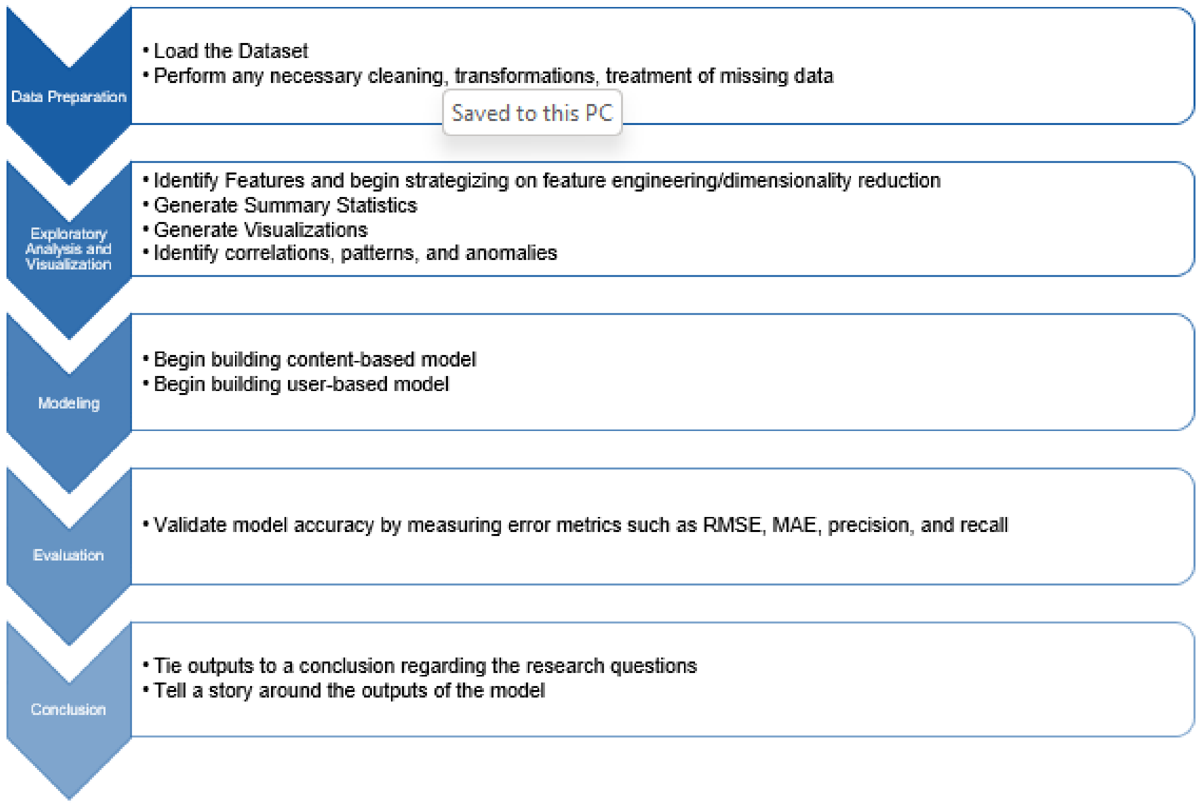
```
0        KU_O5udG6zpxOg-VcAEodg  ... 2018-07-07 22:09:11
1        BiTunyQ73aT9WBnpR9DZGw  ... 2012-01-03 15:28:18
2        saUsX_uimxRlCVr67Z4Jig  ... 2014-02-05 20:30:30
3        AqPFMleE6RsU23_auESxiA  ... 2015-01-04 00:01:03
4        Sx8TMOWLNuJBWer-0pcmoA  ... 2017-01-14 20:54:15
...                         ...  ...                 ...
6990275  H0RIamZu0B0Ei0P4aeh3sQ  ... 2014-12-17 21:45:20
6990276  shTPgbgdwTHSuU67mGCmZQ  ... 2021-03-31 16:55:10
6990277  YNfNhgZlaaCO5Q_YJR4rEw  ... 2019-12-30 03:56:30
6990278  i-I4ZOhoX70Nw5H0FwrQUA  ... 2022-01-19 18:59:27
6990279  RwcKOdEuLRHNJe4M9-qpqg  ... 2018-01-02 22:50:47

[6990280 rows x 9 columns]>
```

Figure 20: df.agg

Data Approach

The approach taken can be broken down into the following steps, with the smaller sub steps to the right. The graphic below describes this in detail, and the steps are explained in detail following the diagram.

Data Approach Process


Data Preparation
• Load the Dataset
• Perform any necessary cleaning, transformations, treatment of missing data
Saved to this PC
Exploratory Analysis and Visualization
• Identify Features and begin strategizing on feature engineering/dimensionality reduction
• Generate Summary Statistics
• Generate Visualizations
• Identify correlations, patterns, and anomalies
Modeling
• Begin building content-based model
• Begin building user-based model
Evaluation
• Validate model accuracy by measuring error metrics such as RMSE, MAE, precision, and recall
Conclusion
• Tie outputs to a conclusion regarding the research questions
• Tell a story around the outputs of the model

Procedure

1. **Step 1: Data Preparation**
   The process of data preparation can be split into loading the dataset and performing any necessary cleaning, transformations, and performing treatment of missing data. Other possibilities include handling special characters and creating strategies for error handling. Specific to the Yelp Dataset, it was provided in five files, with each file created as a .JSON file, one JSON object per line. As Pandas provides a way to read .JSON files through the pandas.read_json functionality, I will keep the files in .JSON rather than converting to .CSV. Through this, I can load the dataset as a data frame in Python. Once loaded, one of the things that I am considering includes filtering the data set into restaurants only as well as focusing on reviews and users within a certain city. As there is approximately 8GB of semi-structured data, it is essential to strategize the model building in this step and keep the models and their required data in mind as I continue.

2. **Exploratory Analysis and Visualization**

After finishing the data cleaning and processing, the next step is exploratory data analysis and visualization. This step is instrumental in understanding the data better and beginning feature engineering/dimensionality reduction. Sub steps that I will be taking include generating summary statistics along with visualizations. Within this step, I can also perform correlation analysis to determine which variables/features are most important to the models in terms of predictive ability.

3. **Modeling**
   Within this step, I will begin building out the models in order to answer my research questions. To answer my first question regarding whether content-based filtering models are "better" or collaborative filtering models are "better", I will need to build both types of models and compare their metrics and recommendation lists to evaluate and answer this question. For my second research question regarding which level of accuracy/performance I can achieve given this dataset, I have decided to extend upon my original abstract and include the metrics of precision, recall, RMSE, and MAE within my analysis. According to my literature review, not only are these metrics popular and simple to understand, but I will also be able to measure the performance of my models on a deeper level and on different facets through the inclusion of these other metrics.

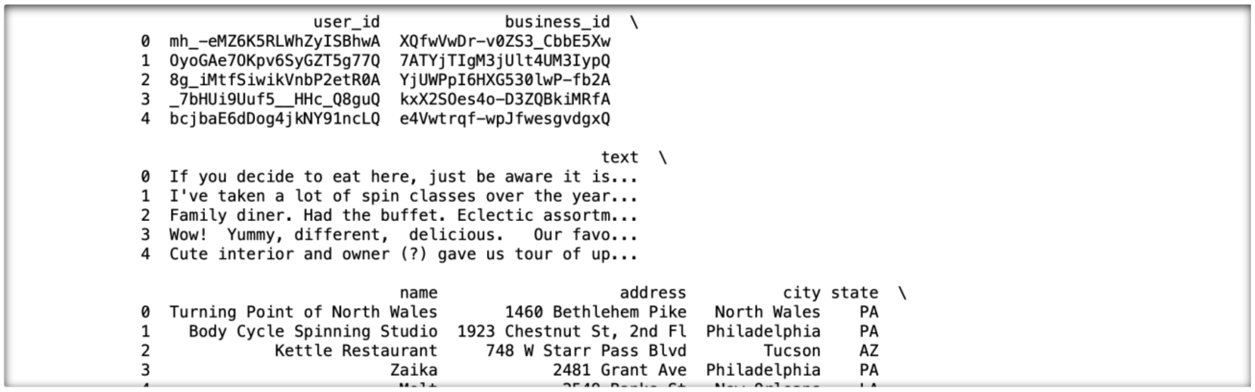
4. **Evaluation**
   While this step may need to be performed several times along with step 3 (modeling) in order to evaluate the performance of the models accurately (or alternatively, with different variables/scenarios), the aforementioned four metrics will be integral in validating the models.

Data Preparation

The dataset files were saved locally in my drive. They are also available for download from the Yelp Dataset website, dated as of February 5, 2023 and last modified in January 19, 2022. When loading the dataset, one of the most important considerations that had to be made was the method of handling the JSON files due to their size and number. This is because loading the dataset into the Jupyter Notebook resulted in many occurrences of dead kernels and long loading times, even for the files with a few hundred MBs. Thus, when selecting from the five JSON files, two were chosen: business.json and review.json that contained information regarding the businesses and the reviews.

For the two JSON files, different loading methods were utilized. For business.json, as it was smaller (118.9MB), it was possible to import the Pandas library and load the business.json file as a dataframe. As there were 14 variables/columns, it was necessary to declare them before loading to ensure Pandas would not use unnecessary time and memory to infer the datatypes. For the review.json file, as it was much larger, simply loading it was not feasible and resulted in many dead kernels due to its size (5GB). Hence, it was pre-determined that three columns (user_ID, business_ID, and stars) were necessary and the rest could be excluded from loading. The loading method chosen was to read the JSON object line by line using a for loop and the values for the three columns were saved in a variable "lst" and then appended to another variable called "chunk". Finally, a Pandas dataframe is created by setting the data parameter to "chunk" and the columns parameter to the list of column names saved in "columns".

The second part of the data preparation was performing any necessary cleaning, transformations, and treatment of missing data. After loading the two JSON files into the aptly named "business" and "review" dataframes, a left join was performed to create a new dataframe using the Pandas "merge" command on the "business_ID" variable to get the businesses with reviews and add the columns from the "business" dataframe to the "review" dataframe. This dataframe was named "businesseswithreviews" (shown below).



```
                 user_id             business_id  \
0  mh_-eMZ6K5RLWhZyISBhwA  XQfwVwDr-v0ZS3_CbbE5Xw
1  OyoGAe7OKpv6SyGZT5g77Q  7ATYjTIgM3jUlt4UM3IypQ
2  8g_iMtfSiwikVnbP2etR0A  YjUWPpI6HXG530lwP-fb2A
3  _7bHUi9Uuf5__HHc_Q8guQ  kxX2SOes4o-D3ZQBkiMRfA
4  bcjbaE6dDog4jkNY91ncLQ  e4Vwtrqf-wpJfwesgvdgxQ

                                               text  \
0  If you decide to eat here, just be aware it is...
1  I've taken a lot of spin classes over the year...
2  Family diner. Had the buffet. Eclectic assortm...
3  Wow!  Yummy, different,  delicious.   Our favo...
4  Cute interior and owner (?) gave us tour of up...

                          name                 address         city state  \
0  Turning Point of North Wales       1460 Bethlehem Pike  North Wales    PA
1    Body Cycle Spinning Studio  1923 Chestnut St, 2nd Fl  Philadelphia    PA
2            Kettle Restaurant     748 W Starr Pass Blvd       Tucson    AZ
3                        Zaika           2481 Grant Ave  Philadelphia    PA
```

Figure 21: businesseswithreviews.head(5) to see the first five entries of each of the columns in the dataframe

The further concentration of the dataset is detailed in the Exploratory Data Analysis and Visualization section.

Exploratory Data Analysis and Visualization

To understand the distribution better, the next step was to create visualizations. The first visualization created was Top 50 Cities by Businesses to plot a histogram of the Top 50 cities with the most businesses. In Figure 22, you can see the top two cities with the most businesses are Philadelphia and New Orleans respectively. In my Github repository, there is also a Pandas Profile that was created on the “businesseswithreviews” dataset to explore the results of univariate and bivariate analyses of the dataset (see figure below). According to the heatmap below, there were no strong correlations. The strongest correlation (other than the variable with itself) was with review_count and postal_code, but nothing material.

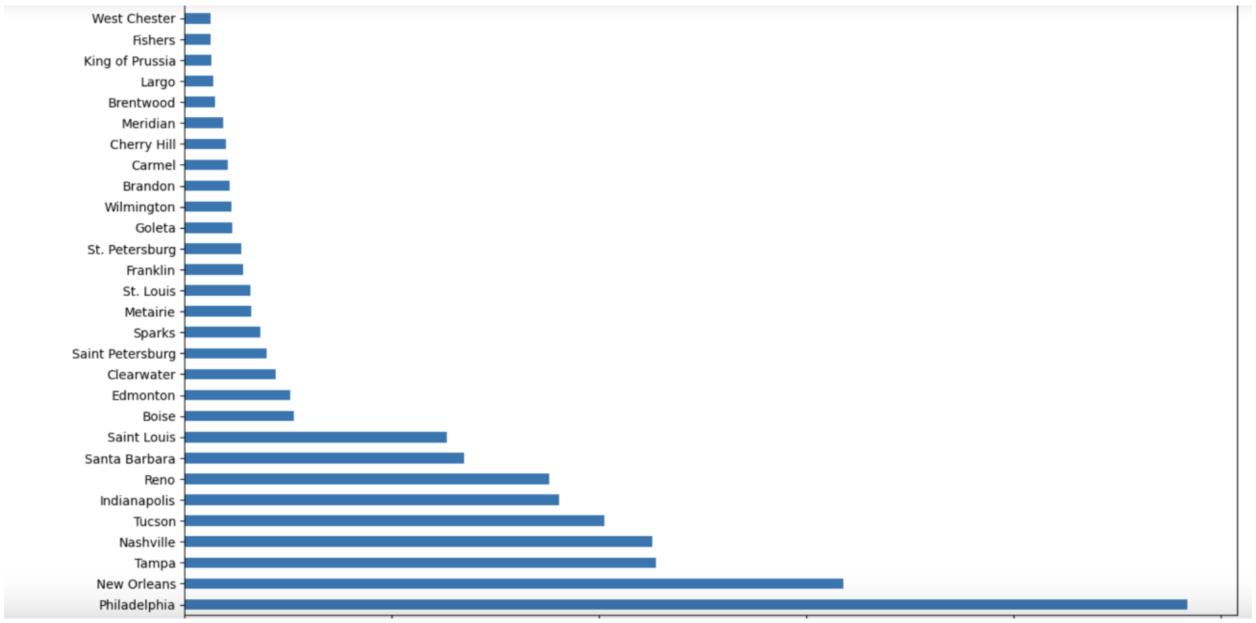


Figure 22: Snippet of the Top 50 Cities by Businesses Histogram

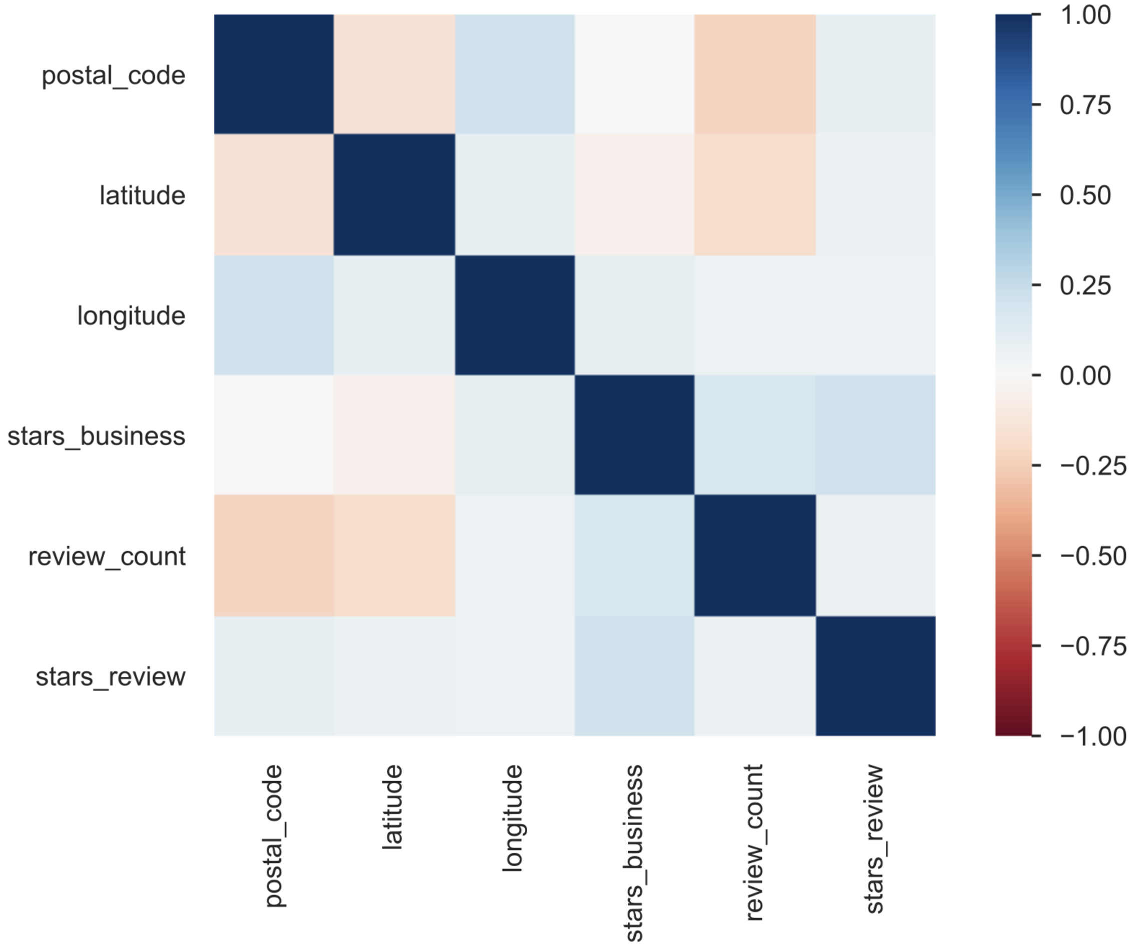


Figure 23: Heatmap of "businesseswithreviews" dataset correlations

The next visualization on the next page used to understand the data better was the "Top 50 Cities by Reviews" histogram. This was to truly understand visually if this would differ from the "Top 50 Cities with the Most Businesses" and to paint a different perspective of the data. This is shown on Figure 24 (see below).

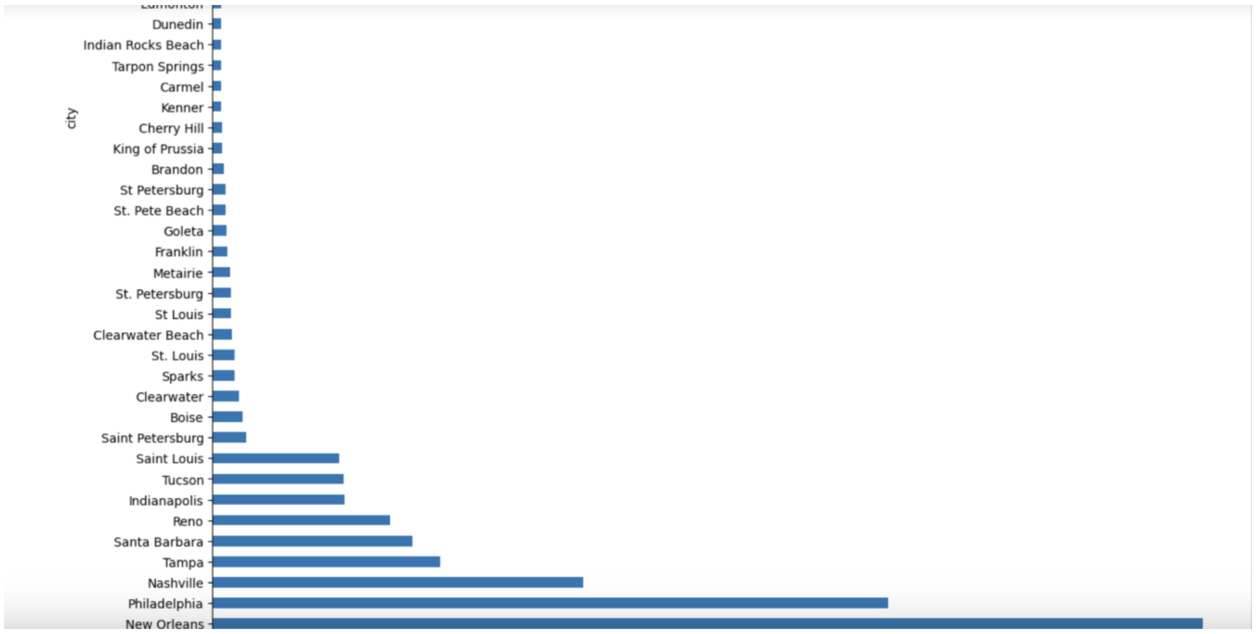
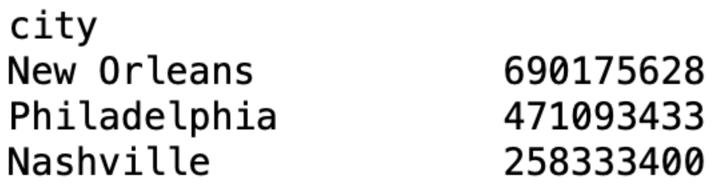


Figure 24: Snippet of the Top 50 Cities by Reviews histogram

In this chart, New Orleans has 690,175,628 reviews as the top city by reviews. In second place is Philadelphia with 471,093,433 (see Figure 25).

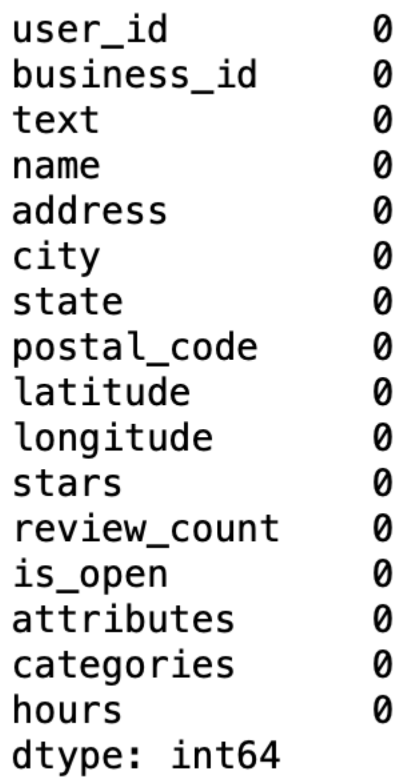


```
city
New Orleans            690175628
Philadelphia           471093433
Nashville              258333400
```

Figure 25: Snippet of Top 50 Cities by Reviews dataset

In order to downsample the dataset “businesseswithreviews”, multiple criteria were used to narrow the focus of the dataset. For example, businesses that were closed were dropped from the dataset to avoid recommendations for places that were no longer in operation. Restaurants were chosen as the focus of the dataset as it is not unreasonable to assume that the tastes and recommendation criteria would differ depending on the store category. Philadelphia was chosen as the city to concentrate on as the top city by number of businesses. Any businesses with 20 or less reviews were removed from the dataset as well. Next, systematic sampling was performed by removing every 10th row. Despite all this narrowing, 448,906 records remained. In order to further break down this dataset to ensure performance of the model, random sampling was performed to get 20% of the dataset. This left 89,781 rows and 16 columns, still a sizable amount of data.

Missing data was checked for, but none existed (Figure 26), thus no treatment was required.



```
user_id          0
business_id      0
text             0
name             0
address          0
city             0
state            0
postal_code      0
latitude         0
longitude        0
stars            0
review_count     0
is_open          0
attributes       0
categories       0
hours            0
dtype: int64
```

Figure 26: Missing Data from the "businesseswithreviews" dataset

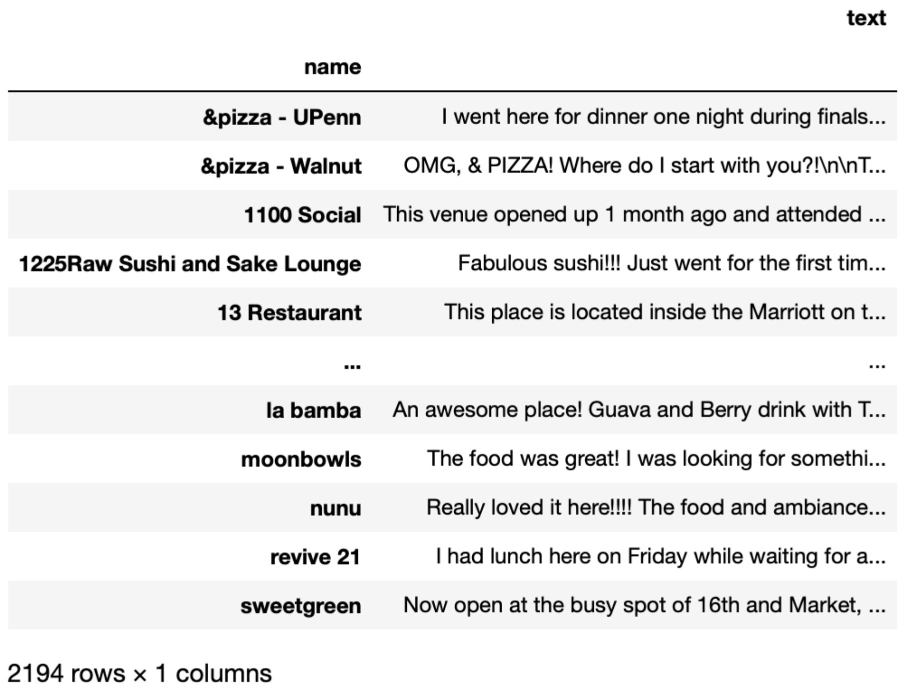
After the downsampling, the dataset was split into train and test sets before any preprocessing was done for the TF-IDF model (content-based filtering model) or Matrix Factorization model (collaborative filtering model). This was done on a common 80/20 split. After the split, the train dataset had 71,824 rows and 16 columns and the test dataset had 17,957 rows and 16 columns.

Modeling

Content-Based Model (TF-IDF Model)

The content-based model was created to use the TF-IDF methodology. The goal of this model is to generate recommendations based on similar businesses. Similar businesses are generated using a similarity measure on a “bag of words” related to the business. TF-IDF stands for Term Frequency Inverse Document Frequency and is based on the idea that if a word is important, it will appear frequently throughout the document. It then calculates a score for each word in the document based on TF (Term Frequency) and IDF (Inverse Document Frequency). Term Frequency is the ratio of the number of times a term appears in a document/total number of terms in a document. Inverse Document Frequency measures the importance of a word across all documents, calculated as the log of the total number of documents/the number of documents that contain the term. Therefore, the final TF-IDF score is calculated by multiplying TF and IDF together. The higher the score, the more important the term is in the document. In this case, it is used to rank documents by relevance to a query.

In this model, if a business is inputted as a query or sample business, the Top 10 most similar businesses are generated. In order to achieve this, an item-by-item matrix was generated as the first step by creating a pivot table where the name of the business was the row and the review “text” was the column. To ensure all reviews were accounted for, the JOIN function was used to combine the review “text” strings into one string (Figure 27).



	text
name	
&pizza - UPenn	I went here for dinner one night during finals...
&pizza - Walnut	OMG, & PIZZA! Where do I start with you?!\n\nT...
1100 Social	This venue opened up 1 month ago and attended ...
1225Raw Sushi and Sake Lounge	Fabulous sushi!!! Just went for the first tim...
13 Restaurant	This place is located inside the Marriott on t...
...	...
la bamba	An awesome place! Guava and Berry drink with T...
moonbowls	The food was great! I was looking for somethi...
nunu	Really loved it here!!!! The food and ambiance...
revive 21	I had lunch here on Friday while waiting for a...
sweetgreen	Now open at the busy spot of 16th and Market, ...

2194 rows × 1 columns

Figure 27: Pivot Table of Business Name and Review “Text”

The next step taken was to preprocess the data to remove stop words such as “a”, “the”, “is”, “are”. Using the TfidfVectorizer, the TF-IDF matrix uses the fit_transform function to get the vectors used for

comparison. Next, the cosine similarity matrix is generated based on item-to-item similarity (in this case, restaurant-to-restaurant similarity) to generate the scores of which restaurants are most similar.
It is not pictured above, but another test was run to generate similarity scores based on categories, rather than review “text” (Figure 29). After running the two, it seems that categories is a more accurate predictor (higher similarity scores) compared to review “text” to determine if two restaurants are similar.

Finally, in preparation to be fed into the recommender system function which will generate the Top 10 recommendations given a restaurant name (query business name), the dataframe “business_reviews”, which grouped categories (column) by restaurant names (rows), is modified to add row numbers as the index/row names. The restaurant names are then moved to become another column and the corresponding review “text” is added as the second column to the dataframe. This is so when the recommender system goes through to find the query business name’s index, it can find the row number based on the index, and in turn the corresponding business name. Then, it will look at the cosine similarities matrix using that same index number and sort the cosine similarities into a sorted list (descending, with the highest scores at the top). The corresponding scores are also fetched and both values are put into a dataframe to house the Top 10 recommendations for the user, based on the top 10 highest cosine similarity scores to a query restaurant.

One change that was made on the TF-IDF code since the initial code phase was the introduction of bigram instead of unigram analysis. For instance, if a sentence were to be “I have three sisters”, unigram analysis would have it read as “I” “have” “three” “sisters”. However, since the way humans understand language is through context with a string of words, bigram analysis is more realistic and hence, may be more accurate. Thus, the aforementioned sentence would read “I have” “have three” “three sisters”.
Results are discussed in the evaluation section.

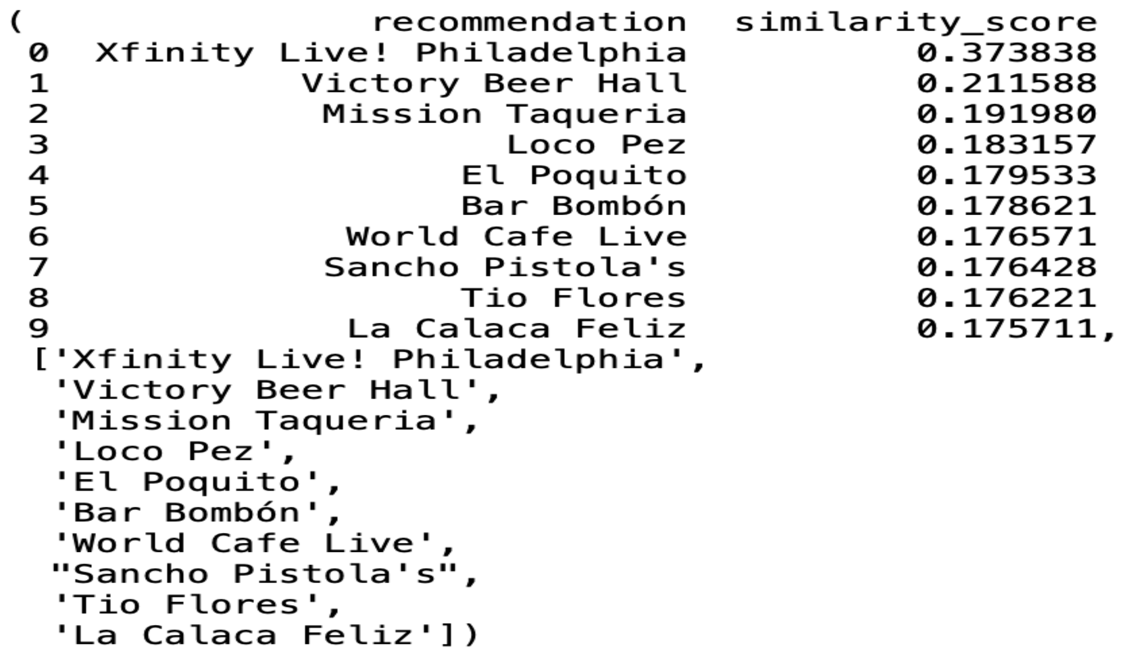
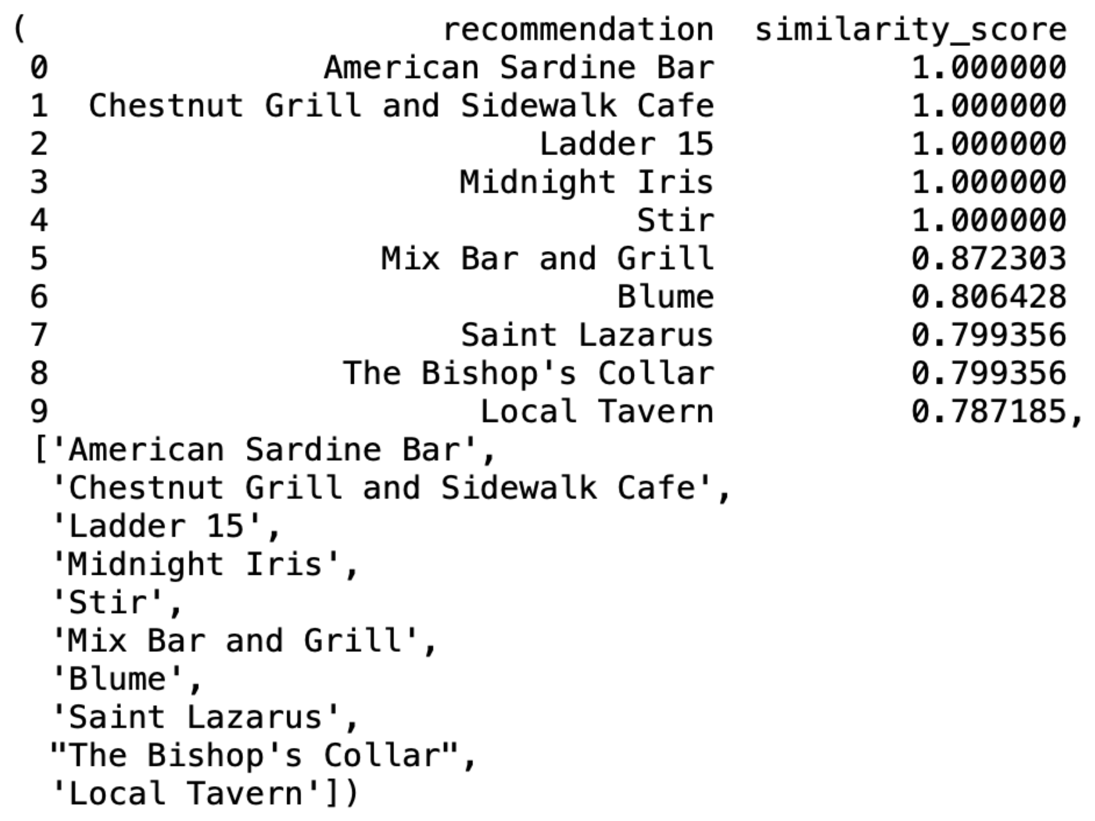

```
(               recommendation  similarity_score
 0  Xfinity Live! Philadelphia          0.373838
 1            Victory Beer Hall          0.211588
 2            Mission Taqueria          0.191980
 3                    Loco Pez          0.183157
 4                  El Poquito          0.179533
 5                  Bar Bombón          0.178621
 6             World Cafe Live          0.176571
 7            Sancho Pistola's          0.176428
 8                  Tio Flores          0.176221
 9             La Calaca Feliz          0.175711,
 ['Xfinity Live! Philadelphia',
  'Victory Beer Hall',
  'Mission Taqueria',
  'Loco Pez',
  'El Poquito',
  'Bar Bombón',
  'World Cafe Live',
  "Sancho Pistola's",
  'Tio Flores',
  'La Calaca Feliz'])
```

Figure 28: Top 10 Recommendations for the restaurant named “1100 Social” with review text



```
(                      recommendation  similarity_score
 0                 American Sardine Bar          1.000000
 1  Chestnut Grill and Sidewalk Cafe          1.000000
 2                            Ladder 15          1.000000
 3                        Midnight Iris          1.000000
 4                                 Stir          1.000000
 5                     Mix Bar and Grill          0.872303
 6                                Blume          0.806428
 7                        Saint Lazarus          0.799356
 8                   The Bishop's Collar          0.799356
 9                         Local Tavern          0.787185,
 ['American Sardine Bar',
  'Chestnut Grill and Sidewalk Cafe',
  'Ladder 15',
  'Midnight Iris',
  'Stir',
  'Mix Bar and Grill',
  'Blume',
  'Saint Lazarus',
  "The Bishop's Collar",
  'Local Tavern'])
```

Figure 29: Top 10 Recommendations for the restaurant named "1100 Social" with categories

Collaborative Filtering Model (Matrix Factorization Model)

For the collaborative filtering model, this was built using the Surprise library. One of the major differences was the use of the "stars" variable from the review.json object, rather than the review "text" variable from the review.json object. Since stars range from 1-5, with 1 being the lowest rating and 5 being the highest, the dataset was normalized using Surprise's Reader function to keep the user item rating (user restaurant rating in this case) on a scale of 1 to 5 for its output recommendation later. The dataset is also loaded with the Surprise library for the variables "user_ID", "name", and "stars_review" and the dataset split into training and testing sets with an 80/20 split.

Next, the Surprise library imports SVD or Singular Value Decomposition, a technique of matrix factorization. Matrix Factorization works by trying to break down a larger matrix into its smaller parts. This is done to understand the properties of it and thus make better predictions of what should fill the sparse matrix. SVD was chosen as we had a large and sparse matrix, where sparse means a matrix with a large number of zero elements relative to the total number of elements. As we have many "NaN" values (many users who do not provide recommendations for every business), we therefore have a sparse matrix. The model is then trained using the SVD method on the training set. Then, the recommendations are generated on the test set.

Summary of Models

The content-based recommendation system was created using the TF-IDF (Term Frequency Inverse Document Frequency) technique. This technique was performed on the "categories" column (alternatively, also the "text" column in the review.json object) in the business.json file (files last modified January 2022). Similarities scores between businesses were calculated using cosine similarity. Recommendations are then generated based on restaurants with the highest similarity scores. This model was evaluated using RMSE, MAE, and alternatively Precision at K. The "ratings" column in review.json was used to pull the actual ratings from a randomly picked user in the test dataset and compared to predicted ratings (a weighted average of the cosine similarity score and the "query business").

The collaborative filtering recommendations system was created using the Surprise library. Two approaches were considered: SVD (Singular Value Decomposition) and ALS (Alternating Least Squares). Both are variations of Matrix Factorization, but employ slightly different methods. When the two were compared in this model, SVD came out as the winner with lower RMSE and MAE scores. Evaluation was performed with K-Fold Cross Validation (5 folds) on RMSE and MAE metrics. For this model, the review.json and business.json objects were used as well.

Evaluation

For evaluation of the content-based model (TF-IDF model), a user-item matrix was created in order to calculate Precision. This was required in order to see which users have reviewed which restaurants, with review count as the values of the matrix. That is to say, if the value is not "NaN", then the user has reviewed the restaurant. Using this logic, we calculate Precision at K with the formula (# of recommended items that were relevant/K total items). K = 10 as we are looking to generate the top 10 recommendations. The # of recommended items that were relevant is calculated as the number of "hits" that we generate. That is to say, if other users who tried (in this case) "1100 Social" also enjoyed the restaurants on the generated Top 10 list, that would count as a "hit" for every restaurant that was enjoyed by a "similar" user who also enjoyed "1100 Social". Then, the average was calculated for the average precision at K, which was generated for each user that also tried "1100 Social".

For the content-based filtering model, one approach mentioned in the previous section was the introduction of bigram analysis in the TF-IDF model as compared to unigram analysis. When incorporating this method, it seems as if the RMSE and MAE scores increased (4.0621920231798 and 4.0 respectively) rather than intuitively decreasing. Therefore, it seems as if unigram analysis promotes a better and more accurate outcome for the model. Upon second thought, this also makes sense as the words that were being captured were "categories", not review "text", where bigram may have been more useful (see Figure 40).

Another approach that was taken to make its results more comparable with the collaborative filtering model was by tweaking the method to use RMSE and MAE as the evaluation metrics. This was done by importing the "stars" column instead of the "text" column from review.json and running the TF-IDF model on the categories. Also, the user-item matrix was created in a similar manner to the one mentioned two paragraphs above, but with stars as the values instead of review count. Then, after computing the cosine similarity matrix, evaluation was performed by extracting a random user_ID in the "df_test" dataframe and generating a list of reviewed restaurants for that specific user_ID (after filtering out user_IDs with less than 5 reviews). The "query business" was the business with the highest rating of all he reviewed restaurants, as recommender systems are built on the assumption of the user looking for other great recommendations. Next, the actual ratings for those reviewed restaurants were generated and put into a dataframe named "df_actual". The predicted ratings were then calculated through a weighted average formula that was put into a dataframe called "df_predicted" which uses the product of the cosine similarity score and the rating from the "query business". Finally, the RMSE and MAE were calculated on the difference between the actual and predicted scores. Results are shown in the conclusion section.

For evaluation of the collaborative filtering model (Matrix Factorization model, the Surprise library has a built-in capability to calculate MAE and RMSE. MAE stands for Mean Absolute Error and RMSE stands for Root Mean Squared Error. While both calculate the average of the absolute differences between predicted values and the actual values, since RMSE incorporates a square root in its calculation, it penalizes errors more heavily than MAE. It can be thus considered a more stringent metric, and RMSE is more sensitive to outliers than MAE. Another technique used was to perform K-Fold cross validation as a method of evaluation on the ALS (Alternating Least Squares technique) as opposed to on the SVD (Singular Value Decomposition).

Both ALS and SVD are commonly used matrix factorization techniques in collaborative filtering systems, but they work differently. ALS works through a trial and error approach by alternating optimizing the user and item factors and basing it on other factors such as past preferences and similar users. SVD is more of a direct approach by factorizing a matrix into its constituent parts using a mathematical approach, but it can be computationally expensive. Results are discussed in the conclusion.

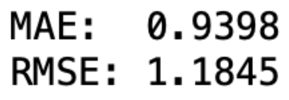
Conclusions & Considerations

The two research questions that these models aimed to answer: "For restaurant recommendations, do content-based filtering models or collaborative filtering models provide better suggestions?" and "What level of accuracy/performance can I achieve given these data points?" To answer the first one, if we take matrix factorization to be a representative and accurate technique, then it seems as if collaborative filtering models provide better suggestions. This is because our precision at K scores for the content-based filtering model had an average of 0 and the collaborative filtering model was able to provide a RMSE and MAE shown below. While RMSE and MAE were not stellar and have room for improvement, we could predict the correct rating within about a 1-star difference. For instance, in the figure shown below, a MAE of 0.9398 means that there is a mean average error between the predicted and actual value of 0.9398, or there is about a 1-star difference for the prediction versus actual values. For RMSE, since we are looking at stars that range from 1-5, an RMSE of 1.1845 is a large difference between predicted and actual star values. Using the ALS algorithm and cross validation, average RMSE across five iterations was 1.192 and average MAE was 0.9469 (figure below). Based on these results, it looks like using SVD was a better choice for this dataset as the RMSE and MAE values were lower using SVD.

For the second question, we were able to achieve a best level of performance within about a 1 star rating difference. Therefore, if we were to predict a user would rate a restaurant as a 5-star restaurant, then we could consider high rated restaurants to be 4-5 star ratings, despite the approximate 1-star RMSE and MAE error terms, we would still be able to accurately predict that they would enjoy this restaurant. Upon averaging three runs of the RMSE and MAE on three separate occasions, the arithmetic average is 1.193 and 0.948 respectively. Therefore, the highest level of accuracy/performance is within a 1-star.

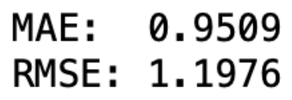
In short, based on this model, taking the TF-IDF model representative of the content-based filtering system performance and the matrix factorization model representative of collaborative filtering recommendation system performance, collaborative filtering outperformed content-based filtering when generating restaurant recommendations, evaluated on RMSE and MAE scores. The main contribution of my work compared to past research is the exploration of the Yelp Dataset from a recommendation standpoint by exploring the TF-IDF and matrix factorization models, pitting their performances together, and evaluating it based on RMSE and MAE scores.

Some shortcomings of this experiment could include taking reviews as an accurate measure of restaurant similarity. That is to say, the keywords that found in reviews using the TF-IDF model, are they most accurate proxy to base similarity scores on? Another consideration is that this experiment takes item-to-item content-based filtering with TF-IDF as representative of content-based filtering as a whole and takes matrix factorization as representative of collaborative filtering models; to answer the first research question comprehensively would require using other techniques from both content-based and collaborative filtering. If these are not the best techniques to suit this dataset, then this thesis of collaborative filtering being a more accurate restaurant recommender would not hold. Next, if this experiment is run again, it may be a more prudent approach to include tips.json as another complement to reviews. Finally, another possibility is to use Euclidean Distance or Jaccard Similarity instead of Cosine Similarity to see how that affects the similarity scores.



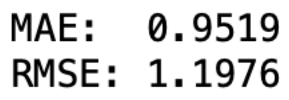
```
MAE:  0.9398
RMSE: 1.1845
```

Figure 30: MAE and RMSE of first run of SVD algorithm



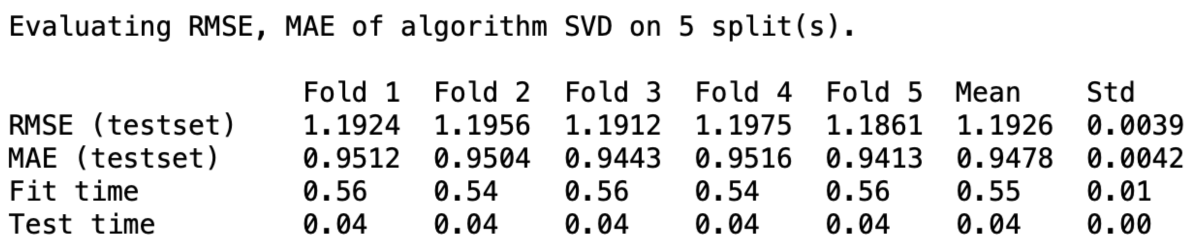
```
MAE:  0.9509
RMSE: 1.1976
```

Figure 31: MAE and RMSE of second run of SVD algorithm



```
MAE:  0.9519
RMSE: 1.1976
```

Figure 32: MAE and RMSE of third run of SVD algorithm



```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.1924  1.1956  1.1912  1.1975  1.1861  1.1926  0.0039
MAE (testset)     0.9512  0.9504  0.9443  0.9516  0.9413  0.9478  0.0042
Fit time          0.56    0.54    0.56    0.54    0.56    0.55    0.01
Test time         0.04    0.04    0.04    0.04    0.04    0.04    0.00
```

Figure 33: Cross Validation of the ALS algorithm

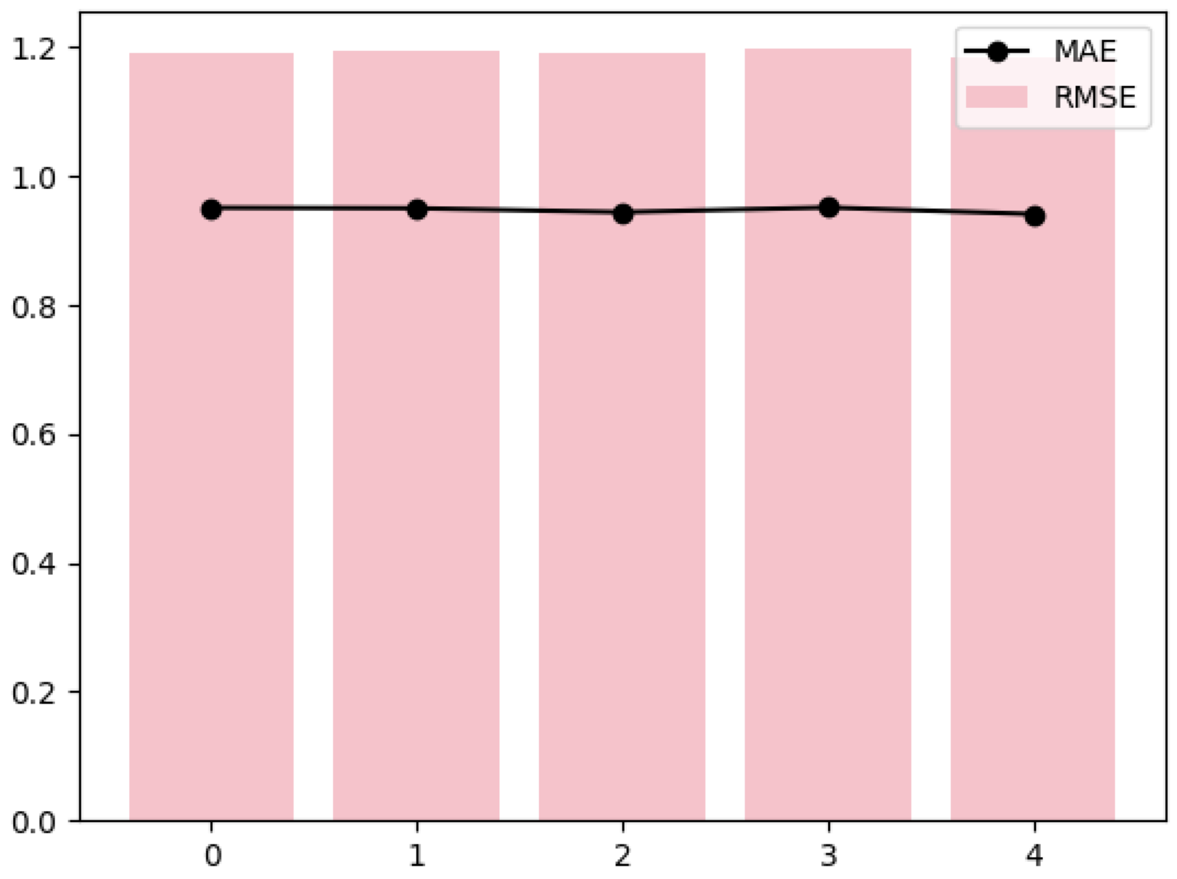
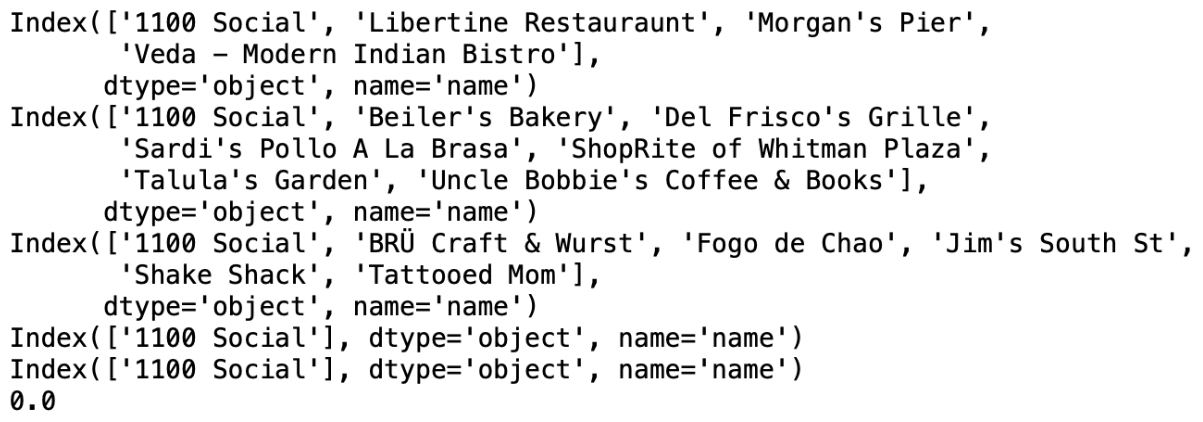
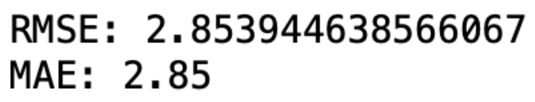


Figure 34: RMSE and MAE scores for Cross Validation of ALS algorithm



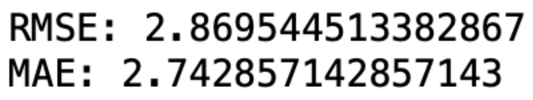
```
Index(['1100 Social', 'Libertine Restauraunt', 'Morgan's Pier',
       'Veda - Modern Indian Bistro'],
      dtype='object', name='name')
Index(['1100 Social', 'Beiler's Bakery', 'Del Frisco's Grille',
       'Sardi's Pollo A La Brasa', 'ShopRite of Whitman Plaza',
       'Talula's Garden', 'Uncle Bobbie's Coffee & Books'],
      dtype='object', name='name')
Index(['1100 Social', 'BRÜ Craft & Wurst', 'Fogo de Chao', 'Jim's South St',
       'Shake Shack', 'Tattooed Mom'],
      dtype='object', name='name')
Index(['1100 Social'], dtype='object', name='name')
Index(['1100 Social'], dtype='object', name='name')
0.0
```

Figure 35: Actual restaurants that users who also went to 1100 Social went to. Last zero represents the precision at K score



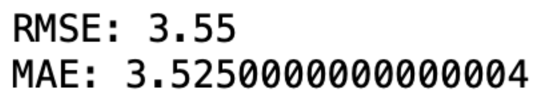
```
RMSE: 2.853944638566067
MAE: 2.85
```

Figure 36: RMSE and MAE for first run of TF-IDF model



```
RMSE: 2.869544513382867
MAE: 2.742857142857143
```

Figure 37: RMSE and MAE for second run of TF-IDF model



```
RMSE: 3.55
MAE: 3.5250000000000004
```

Figure 38: RMSE and MAE for third run of TF-IDF model

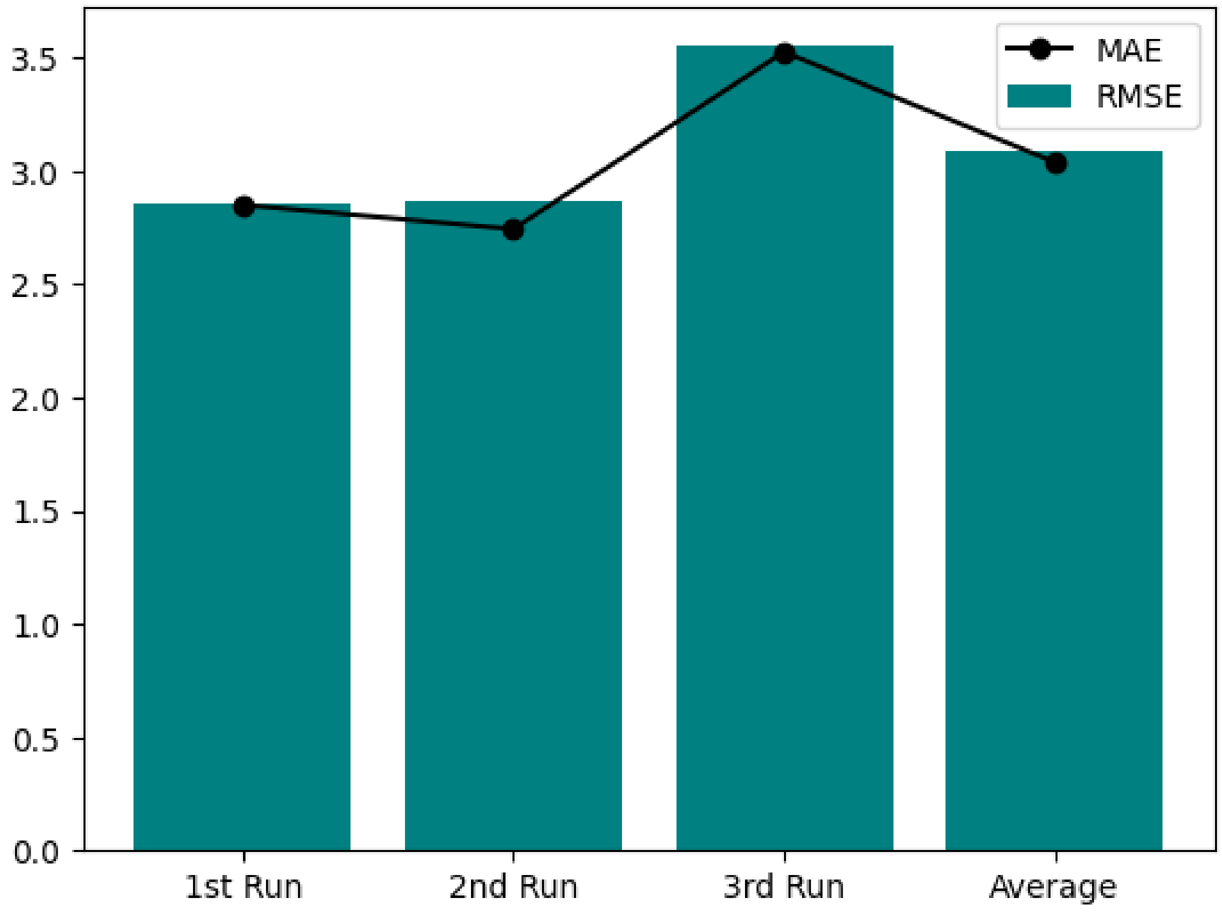
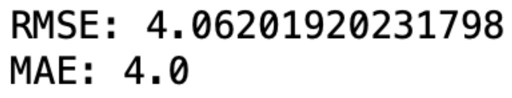


Figure 39:RMSE and MAE scores for first three runs and average across all three runs



```
RMSE: 4.06201920231798
MAE: 4.0
```

Figure 40: RMSE and MAE scores for first run with bigram analysis (TF-IDF model)

Citations

[1] Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender Systems survey. Knowledge-Based Systems, 46, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[2] Al-Nafjan, A., Alrashoudi, N., & Alrasheed, H. (2022). Recommendation system algorithms on location-based social networks: Comparative study. *Information*, *13*(4), 188. https://doi.org/10.3390/info13040188

[3] Silveira, T., Zhang, M., Lin, X., Liu, Y., & Ma, S. (2017). How good your recommender system is? A survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, *10*(5), 813–831. https://doi.org/10.1007/s13042-017-0762-9

[4] Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A survey of recommendation systems: Recommendation models, techniques, and Application Fields. *Electronics*, *11*(1), 141. https://doi.org/10.3390/electronics11010141

[5] Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

[6] Cremonesi, P., & Turrin, R. (n.d.). *An evaluation methodology for Collaborative Recommender Systems - polimi.it*. Retrieved January 24, 2023, from https://chrome.deib.polimi.it/images/8/8e/Cremonesi_2008_CROSSMEDIA.pdf