Assignment: MLUL Group Project Final Report

Batch – AMPBA-2021S

Group - 15

12010011	Prashant	Arya	prashant_arya_ampba2021s@isb.edu
12010078	Abhilash	Gadepalli	abhilash_gadepalli_ampba2021s@isb.edu
12010066	Debjit	Ray	Debjit_Ray_AMPBA2021S@isb.edu

Project Goals

A. Based on past consumer ratings recommend new restaurants to existing consumers using user based Collaborative Filtering. This approach will utilize similar people like similar restaurants concept.
B. Based on past restaurants rated by the current consumer recommend new restaurants to the consumer using item based Collaborative Filtering. This approach will utilize the concept of if I like a set of items, I might like a new item similar to the ones I liked in the past.
C. Finally, we will use the consumer-rating matrix to use Matrix Factorization technique to come up with new recommendations.

Project Objectives

- Recommend new restaurants to users based on:
    - Users similar to the current user
    - Restaurants similar to the user's past likings
    - Combining both user's past likings and other users (similar to current user) past likings.

Data Collection

Instead of any web-scraping/ API calls/ other data collection methods, we decided to use the data already housed at the link:

https://archive.ics.uci.edu/ml/datasets/Restaurant+%26+consumer+data

- This URL has the following set of files, which are categorised as below:
    - **rating_final.csv –** This file has the user ratings provided by different users for different restaurants. The ratings captured are 3 dimensional:
        - **rating –** Overall user rating of the restaurant
        - **service_rating –** The user provided rating to the service provided by the restaurant.
        - **food_rating -** The user provided rating to the food provided by the restaurant.

      This is the primary file, which we will use for building our Collaborative Filtering and Matrix Factorization models.

    - **userprofile.csv** – This file stores different attributes of all the users like his location, smoking/ drinking habits, religion, budget, etc.

      We will be using this file for a contextual rating system.

    - Apart from the above two files, there are other files in the source URL, which can be used to build even better recommendation systems by considering other attributes about the restaurant and the user. These files can be categorized as below:
        - **Restaurant Details:**
            - **chefmozaccepts.csv –** The types of payments accepted by each place.
            - **chefmozcuisine.csv –** The types of cuisines available at each place.
            - **chefmozhours4.csv –** The operating hours and days of each restaurant.
            - **chefmozparking.csv –** The types of parking available at each restaurant.
            - **geoplaces2.csv –** The geographical location of each restaurant along with other attributes like whether alcohol, smoking area is available, Price, Dress Code, etc.

- **User Details:**
  - **usercuisine.csv –** The cuisines liked by each user.
  - **userpayment.csv –** The payment modes used/ preferred by each user.

The source URL provides a lot of information both from the user and restaurant's perspectives, to help us build recommendation engines, which range from the very basic (Item/User based Collaborative Filtering to Matrix Factorization based to Advanced Contextual).

Exploratory Data Analysis

- **File Name - rating_final.csv**

```
In [2]: 1 userRatings = pd.read_csv('./data/rating_final.csv')
```

```
In [23]: 1 userRatings.shape
Out[23]: (1161, 5)
```

```
In [3]: 1 userRatings.head()
```

Out[3]:

	userID	placeID	rating	food_rating	service_rating
0	U1077	135085	2	2	2
1	U1077	135038	2	2	1
2	U1077	132825	2	2	2
3	U1077	135060	1	2	2
4	U1068	135104	1	1	2

```
In [4]: 1 userRatings.describe()
```

Out[4]:

	placeID	rating	food_rating	service_rating
count	1161.000000	1161.000000	1161.000000	1161.000000
mean	134192.041344	1.199828	1.215332	1.090439
std	1100.916275	0.773282	0.792294	0.790844
min	132560.000000	0.000000	0.000000	0.000000
25%	132856.000000	1.000000	1.000000	0.000000
50%	135030.000000	1.000000	1.000000	1.000000
75%	135059.000000	2.000000	2.000000	2.000000
max	135109.000000	2.000000	2.000000	2.000000

- There are 1161 observations of 5 attributes in this file.
- There are no Nulls in the dataset and the ratings vary between 0 to 2.
- The overall distribution of the user ratings is as below:

```
In [30]: 1 uniqueCnts = lambda x: x.value_counts()
         2 userRatings[['rating','food_rating','service_rating']].apply(uniqueCnts)
```

Out[30]:

	rating	food_rating	service_rating
0	254	266	315
1	421	379	426
2	486	516	420

```
In [36]:
tot_cnt = userRatings.shape[0]
calcPct = lambda x:round ((x/tot_cnt)*100,2)
userRatings[['rating','food_rating','service_rating']].apply(uniqueCnts).apply(calcPct)
```

Out[36]:

	rating	food_rating	service_rating
0	21.88	22.91	27.13
1	36.26	32.64	36.69
2	41.86	44.44	36.18

- Also, we find that there are 138 unique users who have rated 130 unique restaurants.

```
In [10]:
print ("The number of unique users who have given ratings " + str(userRatings.userID.nunique()))
print ("The number of unique places which has been rated " + str(userRatings.placeID.nunique()))

The number of unique users who have given ratings 138
The number of unique places which has been rated 130
```

- Thus, we have only **6.47% of the total possible ratings** *(1161/(138 * 130) = 6.47%).*
- We also want to investigate about the users who have given the most and least number of ratings. This gives us a **good idea about who are the newer users and old users.**

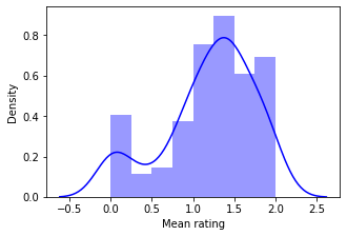
```
In [19]:
print ("The users which have given the most number of ratings:")
userRatings.groupby('userID')['placeID'].count().sort_values(ascending = False).head()

The users which have given the most number of ratings:

Out[19]:
userID
U1061    18
U1106    18
U1134    16
U1024    15
U1022    14
Name: placeID, dtype: int64

In [20]:
print ("The users which have given the least number of ratings:")
userRatings.groupby('userID')['placeID'].count().sort_values(ascending = True).head()

The users which have given the least number of ratings:

Out[20]:
userID
U1138    3
U1074    3
U1047    3
U1039    3
U1031    3
Name: placeID, dtype: int64
```

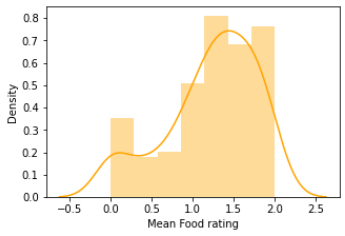
- Similarly, let us also see which are the places which have received the most and least number of ratings. **The high number of ratings in a way can signify the popularity and the low number of ratings might signify new restaurants or restaurants that might need some dedicated publicity drive.**

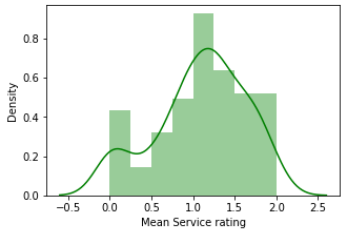
```
In [21]:
print ("The places which have received the most number of ratings:")
userRatings.groupby('placeID')['userID'].count().sort_values(ascending = False).head()

The places which have received the most number of ratings:

Out[21]: placeID
135085    36
132825    32
135032    28
132834    25
135052    25
Name: userID, dtype: int64

In [22]:
print ("The places which have received the most number of ratings:")
userRatings.groupby('placeID')['userID'].count().sort_values(ascending = True).head()

The places which have received the most number of ratings:

Out[22]: placeID
135011    3
132668    3
134975    3
132766    3
135016    3
Name: userID, dtype: int64
```

- Next, we will try to understand the user patterns while rating. For this we will use the mean ratings given out per user. This will help us identify the distribution of tough-raters vs lenient-raters. **As evident, from the distributions, there are a significant number of users who have rated 0 for all the places they have rated and hence, their mean rating is 0. Also, as expected the most number of mean ratings are around the 1- 1.5 mark.**

```
In [62]:
import warnings
warnings.filterwarnings('ignore')
fig = plt.figure(figsize=(12,8))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax = fig.add_subplot(2, 2, 1)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["userID"])["rating"].mean().to_frame(), color='blue', axlabel = "Mean rating")
ax = fig.add_subplot(2, 2, 2)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["userID"])["food_rating"].mean().to_frame(), color='orange', axlabel = "Mean Food rating")
ax = fig.add_subplot(2, 2, 3)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["userID"])["service_rating"].mean().to_frame(), color='g', axlabel = "Mean Service rating"

Out[62]: <AxesSubplot:xlabel='Mean Service rating', ylabel='Density'>
```

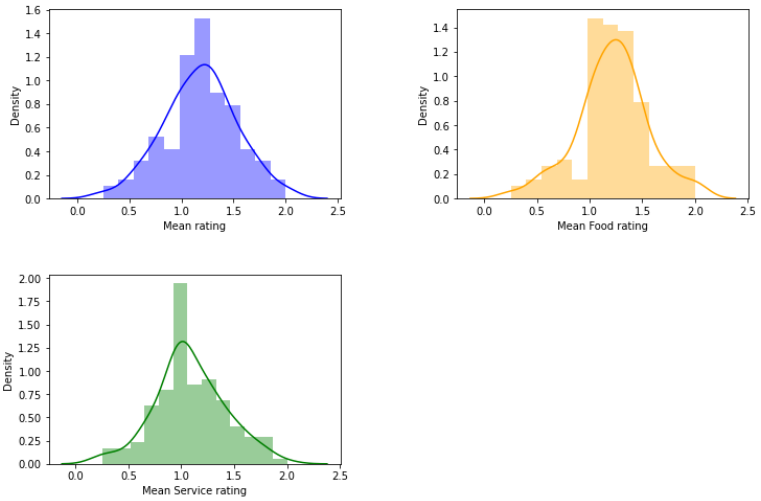






- Similarly, by aggregating the restaurants based on mean ratings, **we can identify the restaurants which need to improve drastically.**

```
In [63]:
import warnings
warnings.filterwarnings('ignore')
fig = plt.figure(figsize=(12,8))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax = fig.add_subplot(2, 2, 1)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["placeID"])["rating"].mean().to_frame(), color='blue', axlabel = "Mean rating")
ax = fig.add_subplot(2, 2, 2)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["placeID"])["food_rating"].mean().to_frame(), color='orange', axlabel = "Mean Food rating")
ax = fig.add_subplot(2, 2, 3)
## Distribution of Average ratings by each user
sns.distplot(userRatings.groupby(["placeID"])["service_rating"].mean().to_frame(), color='g', axlabel = "Mean Service rating")

Out[63]: <AxesSubplot:xlabel='Mean Service rating', ylabel='Density'>
```



- **File Name - userprofile.csv**

```
In [17]: userProfile = pd.read_csv('./data/userprofile.csv')

In [18]: userProfile.shape
Out[18]: (138, 19)

In [19]: userProfile.head()
```

Out[19]:

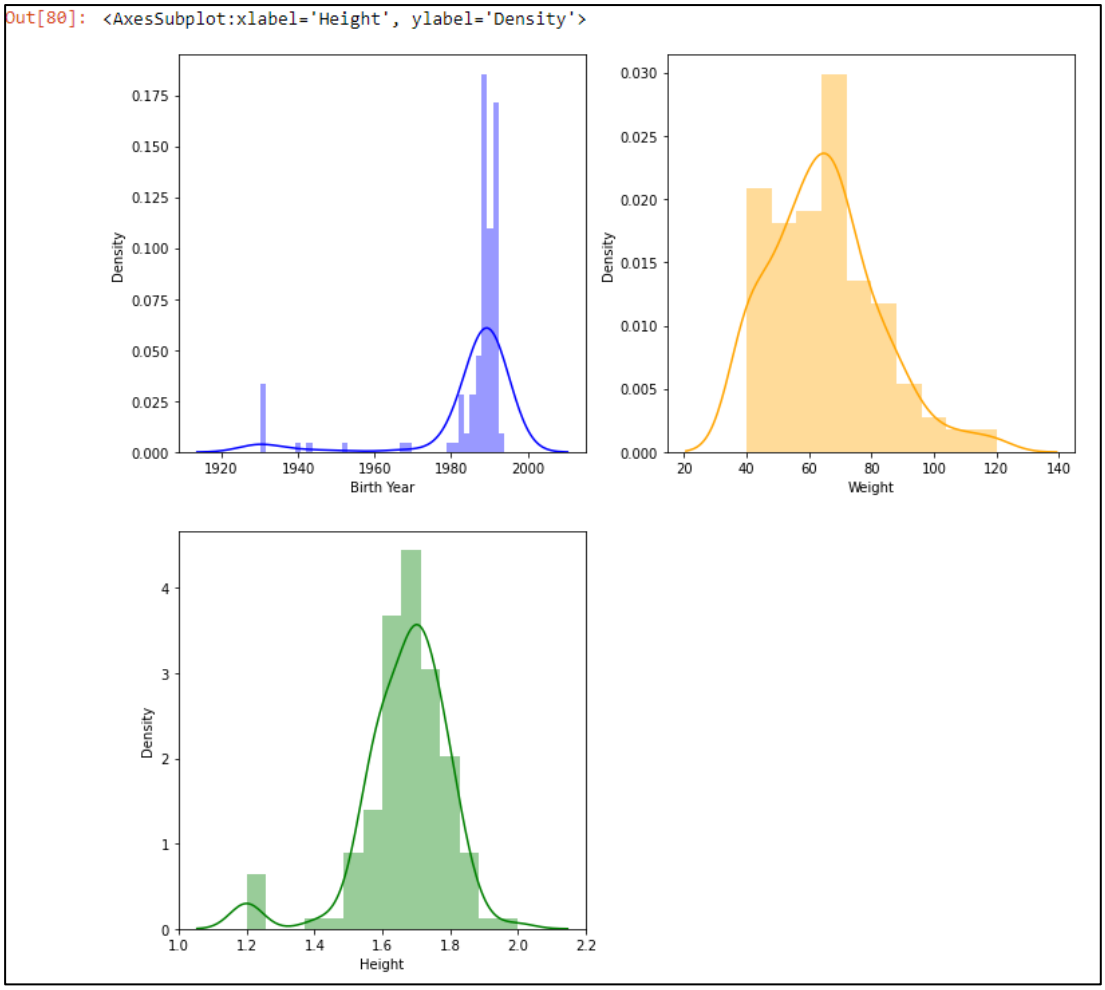
	userID	latitude	longitude	smoker	drink_level	dress_preference	ambience	transport	marital_status	hijos	birth_year	interest	personality
0	U1001	22.139997	-100.978803	false	abstemious	informal	family	on foot	single	independent	1989	variety	thrifty-protector
1	U1002	22.150087	-100.983325	false	abstemious	informal	family	public	single	independent	1990	technology	hunter-ostentatious
2	U1003	22.119847	-100.946527	false	social drinker	formal	family	public	single	independent	1989	none	hard-worker
3	U1004	18.867000	-99.183000	false	abstemious	informal	family	public	single	independent	1940	variety	hard-worker
4	U1005	22.183477	-100.959891	false	abstemious	no preference	family	public	single	independent	1992	none	thrifty-protector

```
In [20]: userProfile.describe()
```

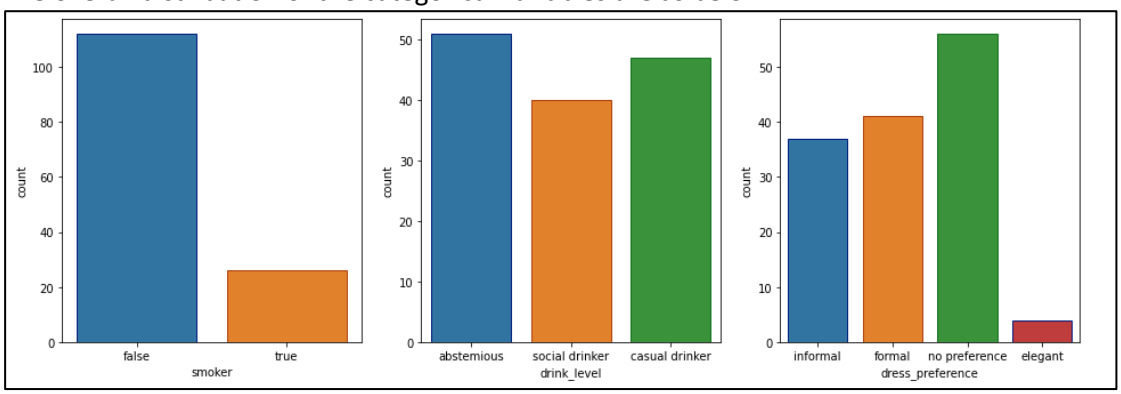
Out[20]:

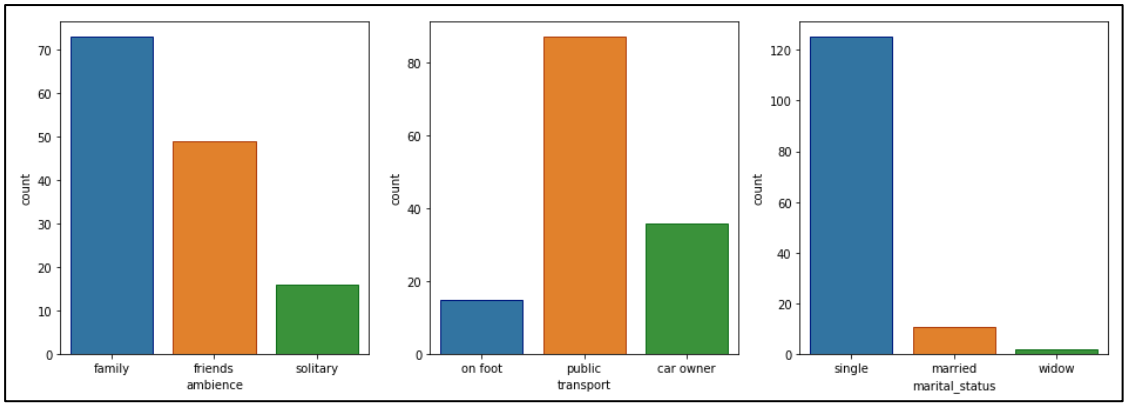
	latitude	longitude	birth_year	weight	height
count	138.000000	138.000000	138.000000	138.000000	138.000000
mean	21.810389	-100.291857	1984.702899	64.869565	1.667536
std	1.552529	0.869916	14.655364	17.214332	0.130473
min	18.813348	-101.054680	1930.000000	40.000000	1.200000
25%	22.126029	-100.983000	1987.000000	53.000000	1.600000
50%	22.150496	-100.937789	1989.000000	65.000000	1.690000
75%	22.186642	-99.183252	1991.000000	74.750000	1.750000
max	23.771030	-99.067106	1994.000000	120.000000	2.000000

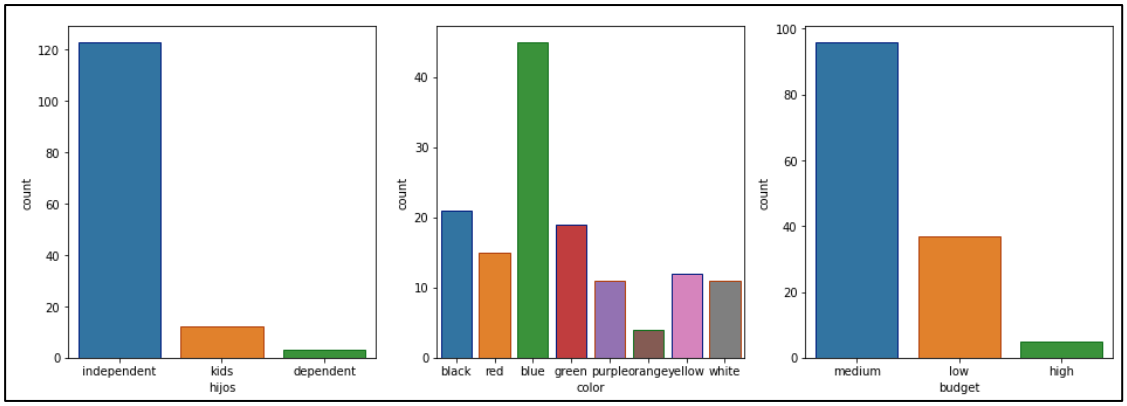
- There are 138 observations of 19 attributes in this file.
- Out of the 19 attributes other than ‘latitude’, ‘longitude’, ‘weight’ and ‘height’ all other are either categorical or strings.
- The users birth year vary between 1930 to 1994, **most users being born in 1989.**

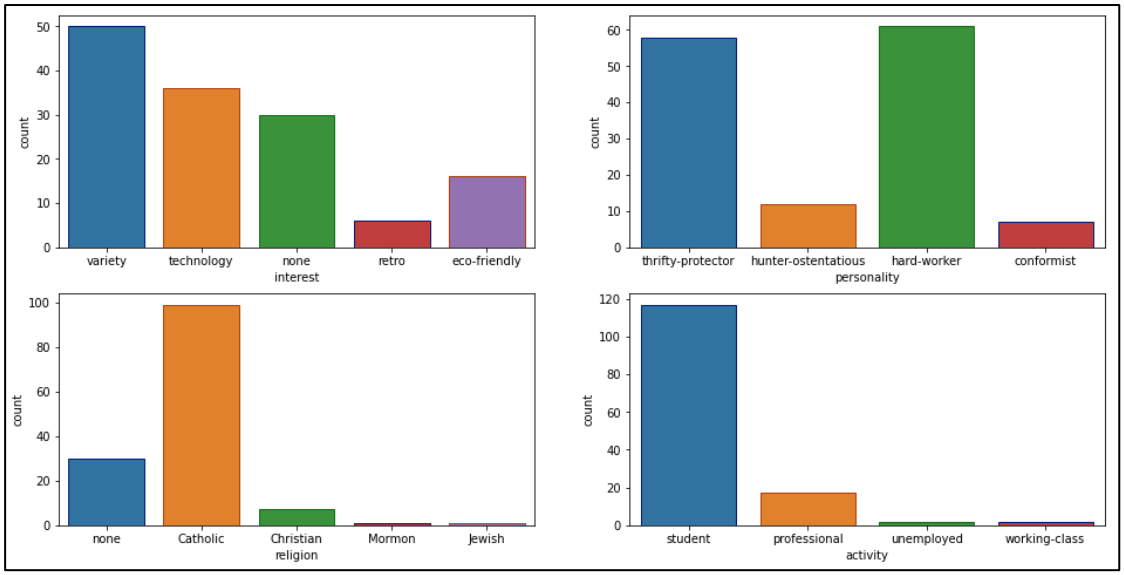


- **The height and weight can be used to derive the BMI and then conclude how many of these users are Obese. This can later help us in relating to their choices and help suggest healthy restaurants.**
- Some of the observations do have '?' in some attributes, which need to be replaced with Null.
- The overall distribution of the categorical variables are as below:

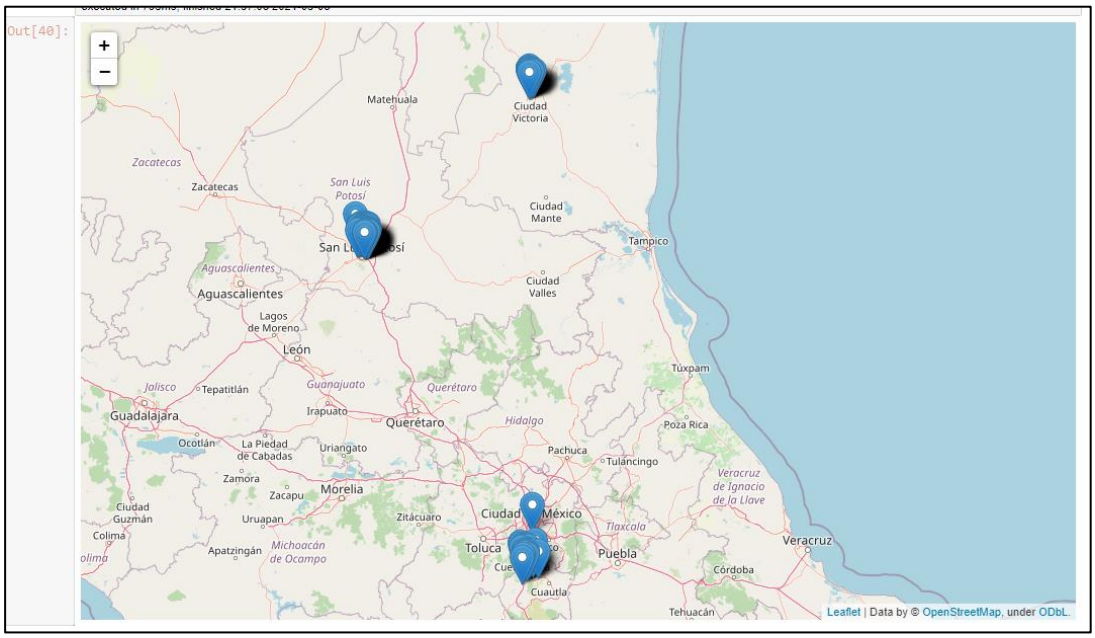



count
family
friends
solitary
ambience
on foot
public
car owner
transport
single
married
widow
marital_status

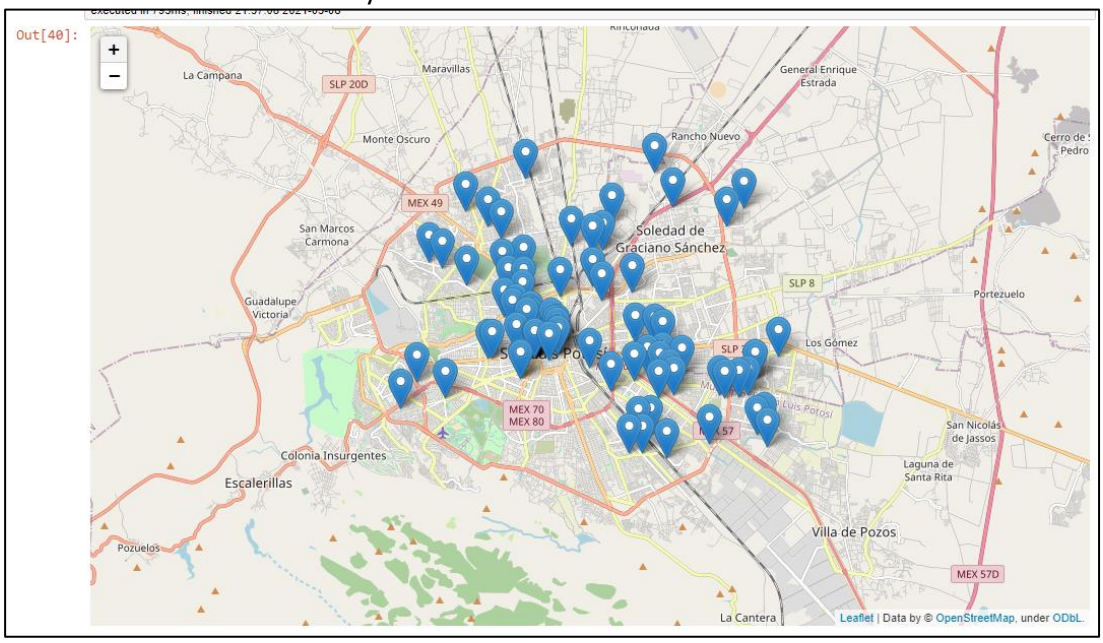

count
independent
kids
dependent
hijos
black
red
blue
green
purple
orange
yellow
white
color
medium
low
high
budget


count
variety
technology
none
retro
eco-friendly
interest
thrifty-protector
hunter-ostentatious
hard-worker
conformist
personality
none
Catholic
Christian
Mormon
Jewish
religion
student
professional
unemployed
working-class
activity

- From the longitude and latitude, we find that the users have been primarily selected from 3 Mexican cities.

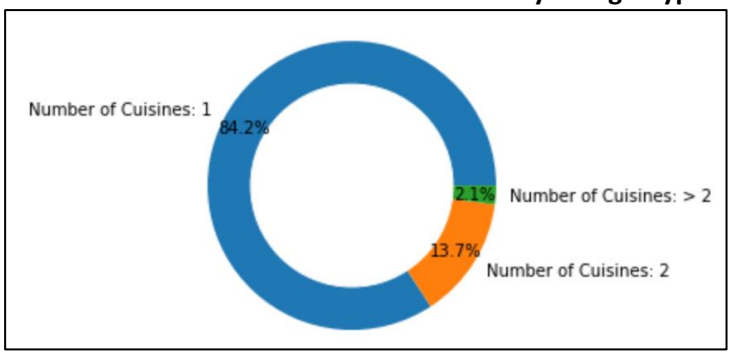


- A zoomed view of one such city shows the location of the users.

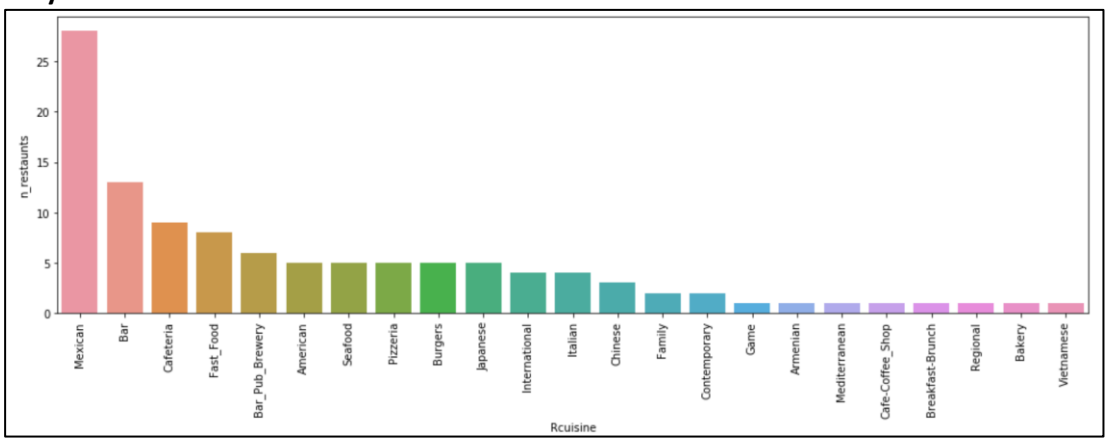


- Using the geo-coordinates provided, it is observed that there are 3 geographic clusters where both users and restaurants are concentrated. Although a user may rate a restaurant from a different geographic cluster, most users rate restaurants within the same geographic cluster.

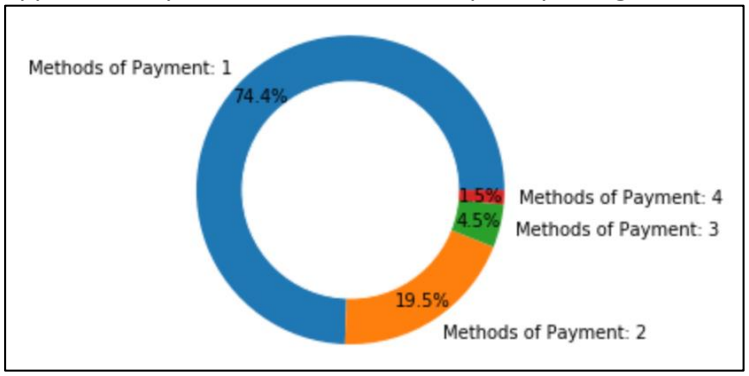
- **Some interesting insights from other data files.**
    - **More than 84% of the restaurants serve only a single type of cuisine.**



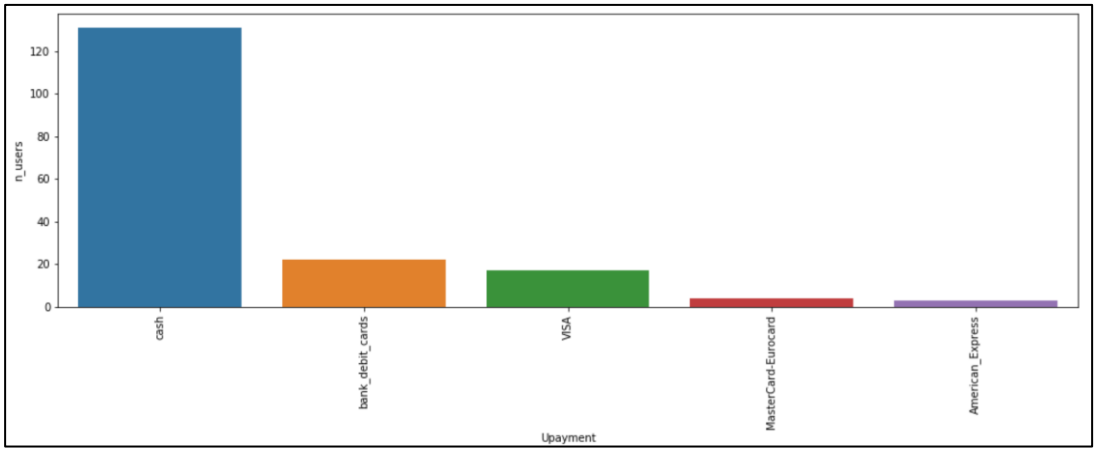
    - **Only 25-30 restaurants** serve **Mexican cuisine**



    - Approximately **75%** of restaurants accept only a **single mode of payment**.



- Almost **all restaurants accept cash.**



Approach

To build our predictive models, we have taken up multiple approaches:

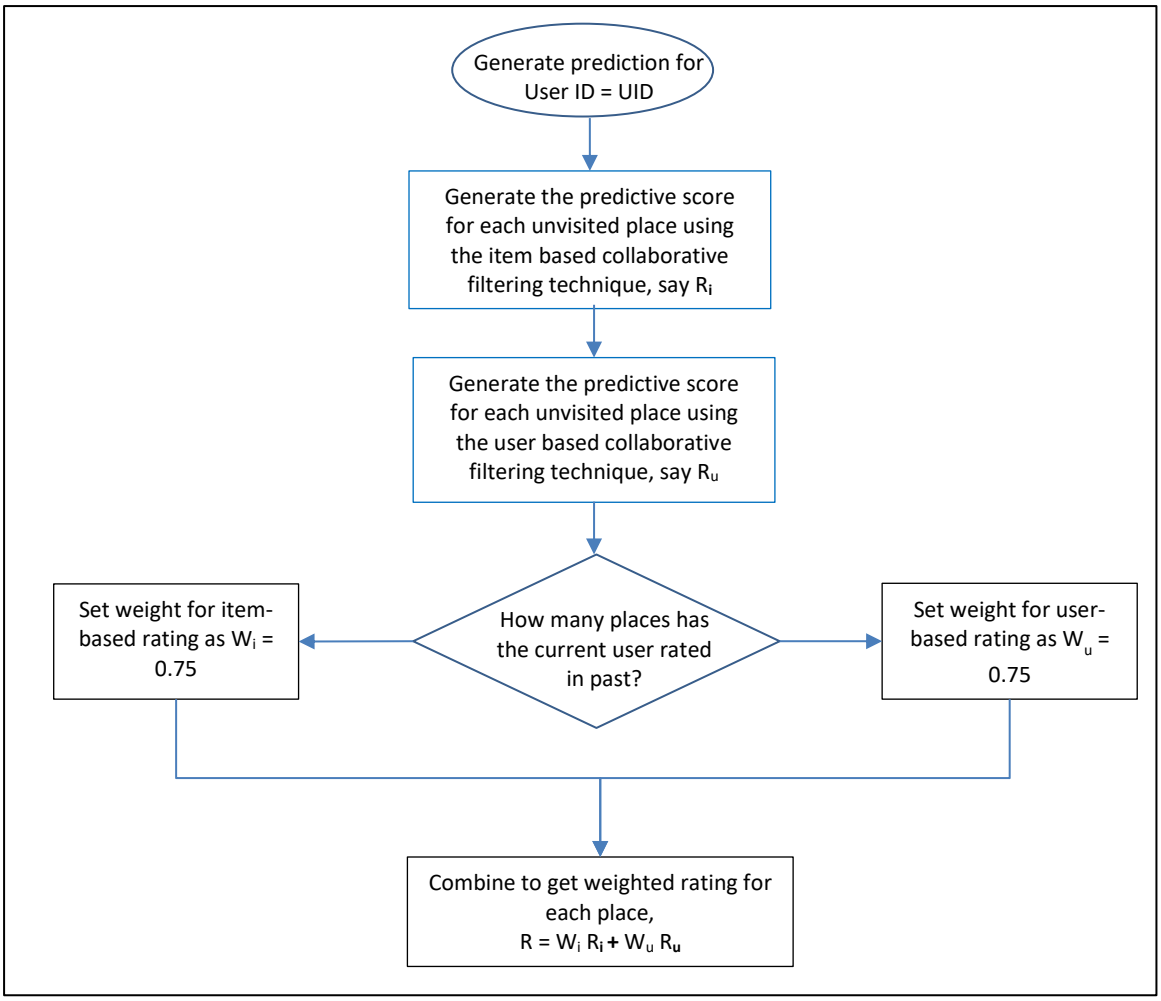
A. **Standard Item Based Collaborative Filtering** – Using a KNNWithMeans model and using cosine as the distance metric, we have built a Item based collaborative filtering model.
B. **Standard User Based Collaborative Filtering** – Again, using a KNNWithMeans model and using cosine as the distance metric, we have built an User based collaborative filtering model.

**KNNWithMeans along with Cosine as distance metric was selected based on the least RMSE.*

metric	model	threshold	RMSE_item_based	RMSE_user_based	RMSE_weighted_pred	RMSE_non_weighted_pred
cosine	KNNBasic	1	1.727974	2.409663	2.082450	2.409663
		2	1.727974	2.409663	2.082450	2.409663
		3	1.727974	2.409663	2.082693	2.410084
		4	1.727974	2.409663	2.075907	2.398345
		5	1.727974	2.409663	2.085210	2.414433
	KNNWithMeans	1	1.883388	1.731350	1.720173	1.731350
		2	1.883388	1.731350	1.720173	1.731350
		3	1.883388	1.731350	1.730080	1.750983
		4	1.883388	1.731350	1.734673	1.760047
		5	1.883388	1.731350	1.724043	1.739032
pearson	KNNBasic	1	1.830910	2.563408	2.239503	2.563408
		2	1.830910	2.563408	2.239503	2.563408
		3	1.830910	2.563408	2.237366	2.559672
		4	1.830910	2.563408	2.234577	2.554795
		5	1.830910	2.563408	2.251629	2.584564
	KNNWithMeans	1	2.014955	1.871697	1.801316	1.871697
		2	2.014955	1.871697	1.801316	1.871697
		3	2.014955	1.871697	1.815122	1.898184
		4	2.014955	1.871697	1.824881	1.916808
		5	2.014955	1.871697	1.817392	1.902523
msd	KNNBasic	1	1.721757	2.310914	2.027946	2.310914
		2	1.721757	2.310914	2.027946	2.310914
		3	1.721757	2.310914	2.014744	2.287703
		4	1.721757	2.310914	2.005774	2.271882
		5	1.721757	2.310914	2.009676	2.278769
	KNNWithMeans	1	1.860368	1.793722	1.759842	1.793722
		2	1.860368	1.793722	1.759842	1.793722
		3	1.860368	1.793722	1.764762	1.803364
		4	1.860368	1.793722	1.768336	1.810353
		5	1.860368	1.793722	1.753889	1.782023

Considering, we have limited amount of data and our data matrix is sparse (6.47% populated), we thought of another Hybrid approach.

C. **Hybrid Collaborative Filtering** – This model is a mix of the User based and Item based collaborative filtering techniques. Here, for users who have already rated more than 3 restaurants, we will give more weightage to the User based collaborative filtering and for other users, we will give more weightage to the Item based collaborative filtering.



D. **Non-negative Matrix Factorization –** We next built a non-negative matrix factorization-based model. Considering, ratings cannot be negative, NMF is suitable for generating the unknown ratings for user-place pair, which users are yet to visit.

Considering we have a file with user characteristics, we thought if we can use the similarity between different users and then help us to predict the ratings for a new pair.

E. **User Characteristics Similarity Based Model –** For this model, we first find the similarities between users using Cosine similarity score and then utilize it to predict the ratings. Mathematically, this can be expressed as:

$$[\mathbf{S}][\mathbf{R}] = [\mathbf{P}]$$

where, S=User-User Cosine Similarity Matrix
R = User-Place Rating Matrix
P = Predictive Matrix

In our use case, S has a dimension of 138 x 138 as there are 138 distinct users.
R has a dimension of 138 x 130 as there are 130 distinct places.
P will be generated as a matrix with dimensions 138 x 130, thus containing the predictive ratings for each user id and place id pair.

To ensure,

- The prediction ratings are in the same scale as the original rating
- Attribute randomness of the data
- And, cross validate our model against the original scaled ratings,

we introduce a weights matrix to form the below equation.

$$[\mathbf{S}][\mathbf{R}][\mathbf{W}] = [\mathbf{P}'] \sim [\mathbf{O}]$$

where, W=Weights Matrix
P' = Updated Predictive Matrix
O = Original User-Place Rating Matrix

In our use case, W has a dimension of 130 x 130
P' will be generated as a matrix with dimensions 138 x 130
O is a matrix with dimensions 138 x 130, containing the original ratings for each user id and place id pair.

Now to calculate the weights matrix, we initiate it with random numbers and then, after each iteration, we calculate the Error Matrix, E as

$$[\mathbf{E}] = [\mathbf{O}] - [\mathbf{P}']$$

Over each iteration, we try to reduce the total error (summation of each cell of Error Matrix, E) by updating the weights matrix, [W]. Once, we have reached a decent stability in the total error, we finalize the weights matrix, [W] and we have generated a dense matrix with the predictive ratings for each user-place pair by using the below formula.

$$[\mathbf{S}][\mathbf{R}][\mathbf{W}] = [\mathbf{P}']$$

Sample predictions generated for the same user based on different methods described above.

```
In [20]: get_top_n('U1099',by='userID',n =10)
         executed in 44ms, finished 22:28:34 2021-06-27
```

Out[20]:

	by_est_item	by_est_user	by_weighted_prediction	by_non_weighted_prediction	by_cosine_predictive_rating	by_nmf_predictive_rating
0	132958	132847	132755	132847	132875	135032
1	132955	135070	132847	135070	134999	135050
2	132922	132846	135055	132846	132851	135041
3	134986	132755	135034	132755	132937	135081
4	135034	135055	132922	135055	132955	135062
5	132755	135057	135057	135057	132921	135052
6	135048	135034	132958	135034	132825	135063
7	135080	132862	132955	132862	132862	135106
8	135013	135054	135070	135054	132755	135079
9	132954	135035	132846	135035	132922	135053

Conclusions and Recommendations

To wrap it up, we have combined all the predictive ratings to a single large dataframe, to allow us to predict:

- The top 'n' suggestive places for any user
- The metric to use for sorting the predictive ratings, can be based on the scenario:
    - If it is a new user and we are aware of his/ her attributes, we can use the Cosine similarity-based model to predict.
    - If over time our user-item matrix becomes far more densely populated, we can use one of the Collaborative Filtering scores to suggest.
    - For all other cases (like a new user), NMF or Cosine similarity based models can be used for predictions.

We have not considered a lot of other attributes available in the datasets, which can be used to further enhance our Cosine based model. These are:

- User Cuisine preference

- Restaurant Cuisine/ payment/ operating hours
- Geographical location of user versus that of the Restaurant
- Matching the user preference attributes against the restaurant attributes like parking availability, dress code, budget, etc.
- Similar to user similarities, we can have similarities between the restaurants and have our predictive model enhanced.

Considering Prediction is always subject to human experience, a lot of times the recommendations might go wrong. However, utilizing the similarity metrics to past data can help us to make predictions which can really bring a change in the Business.

In our use case, the model predictions can be used in the following manner:

- Suggesting an existing user with recommended places to visit
- Providing restaurants with list of users, who might like their restaurants and this can be used by restaurants to run directed marketing campaigns.
- Any new place opening up, can get a view of the characteristics rated highly by users and utilize in their planning phase.

Even though, Predictions might not be the core function of a platform, in today's fast paced world, without a prediction engine; Businesses will most likely lose customer base and additional revenue opportunities.