SAR_v1

March 5, 2023

[308]:
```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

clean_df = pd.read_csv('../Reco/Data/workspaces_clean.csv', index_col=0)
clean_df
```

[308]:

	Workspace_Id	Name	Rating	\
0	1	Eugenio Trias Municipal Public Library	3.8	
1	2	Iván de Vargas Library	4.3	
2	3	Biblioteca Mario Vargas Llosa	3.8	
3	4	Pedro Salinas Library	4.0	
4	5	Acuna Public Library	2.9	
..	…	…	…	
113	261	Harina	3.9	
114	262	The Coffee Corner	4.3	
115	263	The Bear and the Madroño	4.4	
116	264	Cafés Pozo	4.6	
117	265	Cafés La Mexicana	4.6	

	Review_count	Price_range	Category	Address	\
0	800	0	Public library	P.º de Fernán Núñez, 24	
1	313	0	Public library	C. de San Justo, 5	
2	178	0	Public library	C. de Barceló, 4	
3	337	0	Public library	Gta. de la Prta de Toledo, 1	
4	118	0	Public library	C. de Quintana, 9	
..	…	…	…	…	
113	434	2	Coffee shop	C. de Velázquez, 61	
114	314	1	Coffee shop	Av. de Valladolid, 41	
115	590	1	Espresso bar	C. del Doce de Octubre, 16	
116	52	0	Coffee store	C. de Miguel Arredondo, 4	
117	112	0	Coffee store	Calle de Fuencarral, 115	

	Latitude	Longitude	Next_status
0	40.416705	-3.679161	Opens 8:30 AM Mon
1	40.413991	-3.709750	Opens 8:30 AM Mon
2	40.426713	-3.699394	Opens 8:30 AM Mon

```
3     40.407074  -3.710894                Opens 9 AM Mon
4     40.427932  -3.716937                Opens 9 AM Mon
..          …          …                           …
113   40.429262  -3.684050                   Closes 9 PM
114   40.428630  -3.729667                   Closes 9 PM
115   40.415687  -3.675956                Closes 10:30 PM
116   40.394994  -3.695993     Closes 2 PMReopens 5 PM
117   40.429825  -3.702889  Closes 1:40 PMReopens 5 PM

[264 rows x 10 columns]
```

-The code above imports the necessary libraries and reads the cleaned workspace data file 'workspaces_clean.csv' into a Pandas DataFrame called clean_df.

-The index_col=0 parameter specifies that the first column of the CSV file should be used as the index of the DataFrame.

-The clean_df DataFrame contains pre-processed and cleaned data for the workspace recommendation engine. This includes relevant attributes for each workspace, such as location, opening time, price, and ratings, as well as any additional user and workspace information needed for the SAR model.

[309]:
```
# Get 100 row indices labels following pattern:
# User_1, User_2, User_3 ... User_100
user_row_indices = []
for i in range(1, 101):
    user_row_indices.append(f"User_{i}")

# Dictionary to store weighted location ratings to add to dataframe
data = {
    "Workspace_Id": clean_df["Workspace_Id"],
    "Workspace": clean_df["Name"], # workspace locations
    "Category": clean_df["Category"]
}

# number of ratings to generate for each user
num_rows = len(clean_df)

# initialise random_seed to fixed value to always produce same results
random_seed = 2023
for row in user_row_indices:

    # set random seed
    np.random.seed(random_seed)

    # for each user, generate weights for each workspace location and add to␣
 ↪data dictionary
    data[row] = np.random.uniform(1, 5, num_rows).round(1)
```

```
    # increment random seed at each iteration so that each category has␣
 ↪different randomly generated values
    random_seed += 1

# create dataframe with weighted average for each location based on each user
weighted_clean_df = pd.DataFrame(data = data)
print("\nWeighted Average User Rating for each Workspace")
weighted_clean_df
```

```
Weighted Average User Rating for each Workspace
```

[309]:

	Workspace_Id	Workspace	Category	\
0	1	Eugenio Trias Municipal Public Library	Public library	
1	2	Iván de Vargas Library	Public library	
2	3	Biblioteca Mario Vargas Llosa	Public library	
3	4	Pedro Salinas Library	Public library	
4	5	Acuna Public Library	Public library	
..	…	…	…	
113	261	Harina	Coffee shop	
114	262	The Coffee Corner	Coffee shop	
115	263	The Bear and the Madroño	Espresso bar	
116	264	Cafés Pozo	Coffee store	
117	265	Cafés La Mexicana	Coffee store	

	User_1	User_2	User_3	User_4	User_5	User_6	User_7	…	User_91	\
0	2.3	3.4	1.5	1.9	4.2	1.3	1.6	…	2.8	
1	4.6	3.8	4.6	2.7	4.6	1.7	2.1	…	1.3	
2	3.4	1.8	4.7	4.9	4.1	3.7	4.5	…	1.4	
3	1.5	1.2	2.8	1.4	4.5	2.1	3.5	…	4.4	
4	1.6	1.8	2.6	2.9	3.1	3.4	2.8	…	1.8	
..	…	…	…	…	…	…	…	…		
113	4.1	3.7	3.1	4.3	1.6	4.6	2.9	…	4.2	
114	1.4	2.0	2.3	3.4	1.7	2.6	2.2	…	4.9	
115	4.3	4.7	4.7	5.0	2.9	4.5	1.6	…	4.0	
116	1.4	2.0	4.0	1.6	2.2	2.1	4.3	…	2.0	
117	3.5	2.5	4.0	4.5	2.8	3.2	2.9	…	3.2	

	User_92	User_93	User_94	User_95	User_96	User_97	User_98	User_99	\
0	4.4	2.9	2.3	1.9	3.8	4.1	3.6	2.0	
1	2.9	4.2	2.4	1.5	4.0	4.6	1.8	4.9	
2	2.9	1.5	3.5	3.5	4.3	1.9	2.2	4.3	
3	2.5	1.9	3.9	2.4	4.9	4.4	4.8	4.6	
4	3.1	2.2	3.3	1.6	2.7	4.8	2.3	3.5	
..	…	…	…	…	…	…	…	…	
113	4.4	2.2	4.0	4.5	1.2	2.2	3.7	4.3	

```
114      2.5      4.1      4.0      4.9      2.5      1.9      3.5      1.0
115      2.6      3.2      4.0      1.4      1.2      4.0      1.5      1.3
116      1.3      3.0      1.6      1.1      2.3      4.2      1.9      3.1
117      4.3      3.1      1.7      1.6      1.3      1.6      2.6      2.4

     User_100
0         4.9
1         3.9
2         3.2
3         1.7
4         3.1
..        …
113       3.6
114       4.8
115       1.5
116       4.4
117       3.9

[264 rows x 103 columns]
```

-Here, we generate a synthetic dataset for the SAR model, which consists of 100 users and their ratings of the workspaces in the clean_df DataFrame. The users are identified by labels in the format of "User_1" up to "User_100".

-The data dictionary is initialized to store the workspace information from the clean_df DataFrame, including the workspace ID, name, and category.

-The **num_rows** variable is set to the number of rows in the clean_df DataFrame.

-For each user, the code generates random weights for each workspace location using **np.random.uniform(1, 5, num_rows)**, which generates random floating-point numbers between 1 and 5. These weights are rounded to 1 decimal place using the **.round(1)** method. The weights are added to the data dictionary under the key of the user's label.

-The **weighted_clean_df** DataFrame is then created by passing the data dictionary to the **pd.DataFrame()** constructor. This DataFrame shows the weighted average user rating for each workspace, based on each user's randomly generated ratings.

[310]:
```
category_averages_df = weighted_clean_df.groupby("Category").mean().round(1).T
print("\nUser Average Rating for Workspace Categories")
category_averages_df
```

```
User Average Rating for Workspace Categories

/var/folders/xx/l33d299s24b396mf6n2yj2600000gn/T/ipykernel_98499/1309936256.py:1
: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is
deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the
function.
```

```
category_averages_df = weighted_clean_df.groupby("Category").mean().round(1).T
```

[310]:

Category	Bakery	Brunch	Business center	Cafe	Cafeteria
Workspace_Id	245.0	236.5	171.0	198.1	207.5
User_1	3.6	2.3	2.7	3.4	3.1
User_2	3.4	3.3	2.4	3.5	3.4
User_3	2.2	3.1	2.5	3.1	3.4
User_4	3.4	2.2	2.4	2.6	3.0
…	…	…	…	…	…
User_96	1.5	4.3	3.9	3.2	3.4
User_97	3.3	1.3	3.4	2.7	2.8
User_98	1.1	2.9	1.2	3.0	3.2
User_99	4.7	1.8	4.5	3.3	3.3
User_100	2.7	1.8	4.1	3.5	2.7

Category	Coffee roasters	Coffee shop	Coffee store	Coworking space
Workspace_Id	173.0	196.6	248.6	139.0
User_1	3.2	3.1	2.6	2.9
User_2	4.5	3.0	3.3	3.1
User_3	2.2	3.2	3.3	2.8
User_4	3.0	3.1	3.3	3.2
…	…	…	…	…
User_96	3.1	2.7	2.8	3.1
User_97	1.2	3.0	2.7	2.9
User_98	2.6	3.1	2.8	3.0
User_99	3.7	3.0	2.5	2.7
User_100	3.6	3.1	3.1	3.1

Category	Dog cafe	Donuts	Espresso bar	Library	Public library
Workspace_Id	173.0	238.0	244.2	64.2	36.9
User_1	2.0	3.7	2.7	2.9	2.9
User_2	4.1	3.0	3.5	3.0	3.0
User_3	2.8	2.5	4.0	3.0	3.1
User_4	4.9	2.9	2.6	3.0	2.8
…	…	…	…	…	…
User_96	2.3	1.9	2.3	3.3	3.1
User_97	1.9	1.2	3.2	3.0	2.9
User_98	3.0	4.3	3.2	2.6	3.0
User_99	4.2	2.7	3.1	3.0	3.0
User_100	2.9	4.4	2.6	2.8	2.7

Category	Records storage facility	Restaurant	Tea store
Workspace_Id	81.0	256.0	251.0
User_1	4.3	3.0	2.0
User_2	3.0	2.9	1.1
User_3	4.4	2.3	1.9
User_4	2.1	4.4	2.2

```
…                                …          …         …
User_96                          1.3        3.8       4.4
User_97                          1.9        1.0       2.1
User_98                          3.2        1.1       2.6
User_99                          4.9        2.1       1.9
User_100                         1.6        4.7       2.4

Category      University library
Workspace_Id                74.6
User_1                       3.2
User_2                       3.3
User_3                       2.3
User_4                       4.2
…                            …
User_96                      2.9
User_97                      3.0
User_98                      3.1
User_99                      2.9
User_100                     2.8

[101 rows x 18 columns]
```

-In this code, we calculate the average ratings for each workspace category across all users, using the **groupby()** method on the weighted_clean_df DataFrame. The groupby() method groups the rows in the DataFrame by the "Category" column, and the **.mean()** method calculates the mean of the ratings for each category. The resulting DataFrame has the category names as the index and the mean ratings as the columns.

-Then, the **.T** method is called to transpose the DataFrame, so that the categories are now the columns and the mean ratings are the rows.

-Finally, we print the **category_averages_df** , which shows the user average rating for each workspace category based on the synthetic dataset generated in the previous code.

[311]:
```
# Drop Workspace_Id row
category_averages_df.drop("Workspace_Id", axis=0, inplace=True)
category_averages_df
```

[311]:
```
Category  Bakery  Brunch  Business center  Cafe  Cafeteria  Coffee roasters  \
User_1       3.6     2.3              2.7   3.4        3.1              3.2
User_2       3.4     3.3              2.4   3.5        3.4              4.5
User_3       2.2     3.1              2.5   3.1        3.4              2.2
User_4       3.4     2.2              2.4   2.6        3.0              3.0
User_5       1.5     3.1              3.4   2.8        2.8              3.6
…            …       …                …     …          …                …
User_96      1.5     4.3              3.9   3.2        3.4              3.1
User_97      3.3     1.3              3.4   2.7        2.8              1.2
User_98      1.1     2.9              1.2   3.0        3.2              2.6
```

```
User_99      4.7     1.8             4.5   3.3        3.3             3.7
User_100     2.7     1.8             4.1   3.5        2.7             3.6

Category  Coffee shop  Coffee store  Coworking space  Dog cafe  Donuts  \
User_1            3.1           2.6              2.9       2.0     3.7
User_2            3.0           3.3              3.1       4.1     3.0
User_3            3.2           3.3              2.8       2.8     2.5
User_4            3.1           3.3              3.2       4.9     2.9
User_5            3.2           2.4              3.2       4.4     1.0
…                 …             …                …         …       …
User_96           2.7           2.8              3.1       2.3     1.9
User_97           3.0           2.7              2.9       1.9     1.2
User_98           3.1           2.8              3.0       3.0     4.3
User_99           3.0           2.5              2.7       4.2     2.7
User_100          3.1           3.1              3.1       2.9     4.4

Category  Espresso bar  Library  Public library  Records storage facility  \
User_1             2.7      2.9             2.9                       4.3
User_2             3.5      3.0             3.0                       3.0
User_3             4.0      3.0             3.1                       4.4
User_4             2.6      3.0             2.8                       2.1
User_5             2.8      3.2             3.0                       1.7
…                  …        …               …                         …
User_96            2.3      3.3             3.1                       1.3
User_97            3.2      3.0             2.9                       1.9
User_98            3.2      2.6             3.0                       3.2
User_99            3.1      3.0             3.0                       4.9
User_100           2.6      2.8             2.7                       1.6

Category  Restaurant  Tea store  University library
User_1           3.0        2.0                3.2
User_2           2.9        1.1                3.3
User_3           2.3        1.9                2.3
User_4           4.4        2.2                4.2
User_5           4.1        3.8                2.6
…                …          …                  …
User_96          3.8        4.4                2.9
User_97          1.0        2.1                3.0
User_98          1.1        2.6                3.1
User_99          2.1        1.9                2.9
User_100         4.7        2.4                2.8

[100 rows x 18 columns]
```

-Here, we drop the "Workspace_Id" row from the category_averages_df DataFrame using the **drop()** method, which removes the specified row or column from the DataFrame. The **axis=0** parameter specifies that the row should be dropped, and the **inplace=True** parameter ensures

that the DataFrame is modified in place.

-The resulting DataFrame shows the user average rating for each workspace category based on the synthetic dataset generated in the previous code block, with the "Workspace_Id" row removed.

[312]:
```
# Copy clean_df in a dataframe called relevant_train_df
# relevant_train_df will be store only useful features to compare workspaces
relevant_train_df = clean_df.copy()

# Label encode categories
relevant_train_df['Category'] = relevant_train_df['Category'].astype('category')
relevant_train_df['Category'] = relevant_train_df['Category'].cat.codes
relevant_train_df
```

[312]:

	Workspace_Id	Name	Rating	\
0	1	Eugenio Trias Municipal Public Library	3.8	
1	2	Iván de Vargas Library	4.3	
2	3	Biblioteca Mario Vargas Llosa	3.8	
3	4	Pedro Salinas Library	4.0	
4	5	Acuna Public Library	2.9	
..	…	…	…	
113	261	Harina	3.9	
114	262	The Coffee Corner	4.3	
115	263	The Bear and the Madroño	4.4	
116	264	Cafés Pozo	4.6	
117	265	Cafés La Mexicana	4.6	

	Review_count	Price_range	Category	Address	\
0	800	0	13	P.º de Fernán Núñez, 24	
1	313	0	13	C. de San Justo, 5	
2	178	0	13	C. de Barceló, 4	
3	337	0	13	Gta. de la Prta de Toledo, 1	
4	118	0	13	C. de Quintana, 9	
..	…	…	…	…	
113	434	2	6	C. de Velázquez, 61	
114	314	1	6	Av. de Valladolid, 41	
115	590	1	11	C. del Doce de Octubre, 16	
116	52	0	7	C. de Miguel Arredondo, 4	
117	112	0	7	Calle de Fuencarral, 115	

	Latitude	Longitude	Next_status
0	40.416705	-3.679161	Opens 8:30 AM Mon
1	40.413991	-3.709750	Opens 8:30 AM Mon
2	40.426713	-3.699394	Opens 8:30 AM Mon
3	40.407074	-3.710894	Opens 9 AM Mon
4	40.427932	-3.716937	Opens 9 AM Mon
..	…	…	…
113	40.429262	-3.684050	Closes 9 PM

```
114  40.428630  -3.729667                Closes 9 PM
115  40.415687  -3.675956             Closes 10:30 PM
116  40.394994  -3.695993     Closes 2 PMReopens 5 PM
117  40.429825  -3.702889  Closes 1:40 PMReopens 5 PM

[264 rows x 10 columns]
```

[313]:
```
# Store irrelevant column names (those not needed for workspace to workspace comparison) in a list
irrelevant_cols = ["Name", "Review_count", "Address", "Next_status"]

# Drop irrelevant columns from relevant_train_df so that we are left with only
# Workspace Id, rating, price range, category, latitude and longitude
# Workspace Id will not be used in the similatity comparison but jus to identify workspaces
relevant_train_df.drop(irrelevant_cols, axis=1, inplace=True)
relevant_train_df
```

[313]:

	Workspace_Id	Rating	Price_range	Category	Latitude	Longitude
0	1	3.8	0	13	40.416705	-3.679161
1	2	4.3	0	13	40.413991	-3.709750
2	3	3.8	0	13	40.426713	-3.699394
3	4	4.0	0	13	40.407074	-3.710894
4	5	2.9	0	13	40.427932	-3.716937
..	…	…	…	…	…	…
113	261	3.9	2	6	40.429262	-3.684050
114	262	4.3	1	6	40.428630	-3.729667
115	263	4.4	1	11	40.415687	-3.675956
116	264	4.6	0	7	40.394994	-3.695993
117	265	4.6	0	7	40.429825	-3.702889

[264 rows x 6 columns]

- Here, we just create a dataframe called relevant_train_df which contains the workspace Id along with the rating, price range, category, latitude and longitude as we believe only these features are relevant for comparison between workspaces.
- It is important to note that the workspace Id will not be taken into account for the actual comparisons but just to identify the workspaces being compared at a single iteration.

[314]:
```
# Get list of Workspace Ids
workspace_ids = [i for i in weighted_clean_df["Workspace_Id"]]

# Create dataframe for workspace to workspace affinity matrix
# With indices and columns equal to the workspace Ids
workspace_workspace_df = pd.DataFrame(index=workspace_ids, columns = workspace_ids)
```

```
# Loop through each workspace Id
for i in workspace_ids:

    # Get the relevant data (rating, price range, category, latitude and␣
 ↪longitude) to store as workspace_j
    # Exclude column with index 0 as it contains the workspace Id which is only␣
 ↪useful for
    # Identifying the workspace but not for the actual similarity comparison
    workspace_i = np.array([relevant_train_df[relevant_train_df["Workspace_Id"]␣
 ↪== i].iloc[0, 1:]])

    # loop again through every workspace which we will call workspace_j to be␣
 ↪compared with workspace_i
    for j in workspace_ids:
        # For workspace_j, get the relevant data (rating, price range,␣
 ↪category, latitude and longitude)
        # Exclude column with index 0 as it contains the workspace Id which is␣
 ↪only useful for
        # Identifying the workspace but not for the actual similarity comparison
        workspace_j = np.
 ↪array([relevant_train_df[relevant_train_df["Workspace_Id"] == j].iloc[0, 1:
 ↪]])

        # Get the cosine similarity between workspace_i and workspace_j
        similarity = cosine_similarity(workspace_i, workspace_j)[0].round(5)

        # Add the cosine_similarity to the workspace to workspace affinity␣
 ↪matrix between two compared workspaces
        workspace_workspace_df.loc[i, j] = similarity[0]

print("\nWorkspace to Workspace Affinity Matrix\n")
workspace_workspace_df
```

```
Workspace to Workspace Affinity Matrix
```

[314]:

	1	2	3	4	5	6	7	8	\
1	1.0	0.99993	1.0	0.99999	0.99978	1.0	0.99987	0.99996	
2	0.99993	1.0	0.99993	0.99998	0.99947	0.99993	0.99961	1.0	
3	1.0	0.99993	1.0	0.99999	0.99978	1.0	0.99987	0.99996	
4	0.99999	0.99998	0.99999	1.0	0.99967	0.99999	0.99978	0.99999	
5	0.99978	0.99947	0.99978	0.99967	1.0	0.99978	0.99999	0.99954	
..	…	…	…	…	…	…	…	…	
261	0.98564	0.98565	0.98565	0.98565	0.98529	0.98565	0.98543	0.98564	
262	0.98641	0.98654	0.98643	0.98647	0.98587	0.98643	0.98605	0.98651	
263	0.99859	0.9987	0.99859	0.99865	0.99805	0.99859	0.99821	0.99868	

```
264  0.99019  0.99039   0.9902  0.99028   0.9895   0.9902  0.98971  0.99035
265  0.99017  0.99037  0.99018  0.99026  0.98948  0.99018  0.98969  0.99033

         9        10   …      256      257      258      259      260  \
1    0.99974  0.99999  …  0.99795  0.99878  0.97743  0.99023  0.98629
2    0.99968  0.99997  …  0.99779  0.99895  0.97775  0.99041   0.9865
3    0.99975  0.99999  …  0.99794  0.99878  0.97745  0.99024   0.9863
4    0.99973      1.0  …   0.9979  0.99886  0.97757  0.99031  0.98638
5    0.99951  0.99967  …  0.99789  0.99814  0.97656  0.98959  0.98558
..       …      …  …      …      …      …      …      …
261  0.98905  0.98574  …  0.97871  0.99186  0.99736  0.99844  0.99956
262  0.98983  0.98656  …  0.97829  0.99284  0.99843  0.99941  0.99997
263  0.99936  0.99867  …  0.99553  0.99971  0.98595  0.99536  0.99311
264  0.99306  0.99035  …  0.98193  0.99566  0.99736      1.0  0.99942
265  0.99305  0.99033  …  0.98191  0.99565  0.99737      1.0  0.99942

         261      262      263      264      265
1    0.98564  0.98641  0.99859  0.99019  0.99017
2    0.98565  0.98654   0.9987  0.99039  0.99037
3    0.98565  0.98643  0.99859   0.9902  0.99018
4    0.98565  0.98647  0.99865  0.99028  0.99026
5    0.98529  0.98587  0.99805   0.9895  0.98948
..       …      …      …      …      …
261      1.0  0.99966  0.99278   0.9984  0.99841
262  0.99966      1.0  0.99313  0.99939   0.9994
263  0.99278  0.99313      1.0  0.99535  0.99534
264   0.9984  0.99939  0.99535      1.0      1.0
265  0.99841   0.9994  0.99534      1.0      1.0

[264 rows x 264 columns]
```

-Here, we create **a workspace to workspace affinity matrix** based on the synthetic dataset generated in the previous code block:

1. First, a list of workspace IDs is created by selecting the "Workspace_Id" column from the weighted_clean_df DataFrame.

2. Next, a new DataFrame called **workspace_workspace_df** is created, with the same index and columns as the workspace IDs list. This DataFrame will serve as the workspace to workspace affinity matrix.

3. The code then loops through each row index of the workspace_workspace_df DataFrame, **representing the current workspace being analyzed**. For each workspace, it retrieves the rating, price range, category, latitude and longitude for that workspace from the relevant_train_df DataFrame.

4. It then loops again through each row index of the workspace_workspace_df DataFrame, **representing the workspace to compare**. For each comparison workspace, it retrieves the rating, price range, category, latitude and longitude for that workspace from the rele-

vant_train_df DataFrame.

5. The code then calculates **the cosine similarity between the current workspace and the workspace to compare**, and fills in the corresponding cell in the workspace_workspace_df DataFrame with the calculated cosine similarity.

6. The resulting workspace_workspace_df DataFrame shows the **affinity scores between all pairs of workspaces** based on the synthetic dataset generated in the previous code block. It has the same index and columns as the workspace IDs list, and the cells represent the cosine similarity between each pair of workspaces.

[315]:
```
# Recommendation scores are obtained by multiplying the workspace-to-workspace affinity matrix
# by the User_1 affinity vector
rec_scores = workspace_workspace_df.values.dot(weighted_clean_df["User_1"].values)

# Get data_frame for User_1 Workspace recommendation scores (descending order)
# Index equates to the workspace Id for each workspace
data = {"User_1_Recommendations": rec_scores}
user_1_rec = pd.DataFrame(data=data, index=workspace_workspace_df.index)
user_1_rec.sort_values("User_1_Recommendations", ascending=False, inplace=True)
user_1_rec
```

[315]:

	User_1_Recommendations
173	786.468805
146	786.410329
139	786.409325
164	786.40885
129	786.408236
..	…
68	772.978108
57	772.841881
30	772.485873
236	772.404855
245	769.187075

[264 rows x 1 columns]

Here, we calculate the recommendation scores for each workspace for the hypothetical user "User_1". We use the workspace to workspace affinity matrix (workspace_workspace_df) and the affinity vector for "User_1" **(weighted_clean_df["User_1"])** generated in the code blocks prior:

1. The first line of the code calculates the recommendation scores by performing a **dot product between the workspace-to-workspace affinity matrix** (workspace_workspace_df.values) and the User_1 affinity vector (weighted_clean_df["User_1"].values).

2. The resulting recommendation scores are then added to a new DataFrame called **user_1_rec**

with the column name **"User_1_Recommendations"**. The index of the DataFrame corresponds to the workspace IDs, and the cells represent the recommendation scores for each workspace.

3. Finally, the user_1_rec DataFrame is sorted **in descending order** based on the recommendation scores.

The resulting DataFrame shows the recommended workspaces for User_1, with the top recommended workspace at the top of the DataFrame.

Print Recommendations

[316]:
```
# Get sub-dataframe with top 5 scored workspaces
top_5_workspaces = user_1_rec.head(5)
print("\nTop 5 Workspaces for User 1")
top_5_workspaces
```

```
Top 5 Workspaces for User 1
```

[316]:

	User_1_Recommendations
173	786.468805
146	786.410329
139	786.409325
164	786.40885
129	786.408236

Here, we extract the top 5 recommended workspaces for "User_1" from the user_1_rec DataFrame generated in the previous code block. We create a new DataFrame called **top_5_workspaces** that contains the top 5 recommended workspaces with their corresponding recommendation scores.

The **head(5)** method is used to extract the first 5 rows (i.e., the top 5 recommended workspaces) of the user_1_rec DataFrame.

The resulting top_5_workspaces DataFrame is printed.

[317]:
```
def print_workspace(workspace_id):
    # Get workspace row based on id
    workspace = clean_df[clean_df["Workspace_Id"] == workspace_id]

    # Get price range string based on category codes
    price_range_cat = workspace["Price_range"].values[0]
    if(price_range_cat == 0):
        price_range = None
    elif(price_range_cat == 1):
        price_range = "€"
    elif(price_range_cat == 2):
        price_range = "€€"
    elif(price_range_cat == 3):
        price_range = "€€€"
```

```
    # Print workspace details
    print(workspace["Name"].values[0])
    print(workspace["Address"].values[0])
    print(workspace["Category"].values[0])
    if price_range is not None:
      print(f"Price range: {price_range}")
    print(f"Overall Rating: {workspace['Rating'].values[0]}")
```

Here, we define a function called **print_workspace** that takes a **workspace_id** as an input parameter.

The function first retrieves the row in the clean_df dataframe that corresponds to the given workspace id. It then extracts the details about the workspace such as **its name, address, category, price range and overall rating**, and prints them out to the console in a structured format.

The function is meant to be used to display details of a workspace to a user, given a workspace id.

[318]:
```
# Print top 5 recommended workspaces for User_1

print("\nWorkspace Recommendations for User 1\n")

# Initialise to make top 5 count
top = 0

# Get index/workspace Id of each top 5 workspace
for i in top_5_workspaces.index:
    top += 1
    # Print details for each top choice
    print(f"Top {top} Choice\n")
    print_workspace(i)
    print("\n\n")
```

```
Workspace Recommendations for User 1

Top 1 Choice

1000 Cups Specialty Coffee & Food
Cmo. de Ganapanes, 1
Dog cafe
Overall Rating: 4.2



Top 2 Choice

WeWork - Espacio de oficinas y coworking
P.º de la Castellana, 43
```

```
Coworking space
Overall Rating: 4.4



Top 3 Choice

WeWork - Espacio de oficinas y coworking
P.º de la Castellana, 77
Coworking space
Overall Rating: 4.4



Top 4 Choice

Talent Garden Madrid
C. de Juan de Mariana, 15
Coworking space
Overall Rating: 4.5



Top 5 Choice

la raum de chamberi coworking
Calle de Modesto Lafuente, 7
Coworking space
Overall Rating: 4.3
```

-Here, this code prints the top 5 recommended workspaces for User 1 by looping through each workspace index in the top_5_workspaces dataframe, printing the details of each workspace using the print_workspace() function, and **incrementing a counter** to keep track of the current top recommendation number.

-At each iteration of the loop, the code prints a heading indicating the current top choice number ("Top 1 Choice", "Top 2 Choice", etc.), followed by the workspace details printed by the print_workspace() function, and two newlines for separation between recommendations.

-This code enables the user to quickly view and compare the recommended workspaces for User 1, ranked by their recommendation score from highest to lowest, and allows them to easily identify the top choices for further investigation.