Netflix Clustering Based System

Assume I'm a data analyst for Netflix and I want to investigate how people differ and agree in their movie preferences based on how they rate various movies. Can these ratings help Netflix to provide better user recommendations? Let's examine the facts and find out.

Objective

Let's start by examining how genre ratings connect to one another before moving on to individual movie ratings later in the notebook.

1

Dataset overview

The dataset has two files. We'll import them both into pandas dataframes:[1]

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import helper
import seaborn as sns

sns.set()
%matplotlib inline
```

Import movies Dataset [1]

```
movies = pd.read_csv('/content/movies.csv')
```

```
movies.head(7)
```

```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)
5        6                         Heat (1995)
6        7                      Sabrina (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                        Comedy|Drama|Romance
4                                       Comedy
```

```
5                         Action|Crime|Thriller
6                                Comedy|Romance
```

Import rating dataset[1]

```
ratings = pd.read_csv('/content/ratings.csv')
```

```
ratings.head(7)
```

```
   userId  movieId  rating   timestamp
0       1       31     2.5  1260759144
1       1     1029     3.0  1260759179
2       1     1061     3.0  1260759182
3       1     1129     2.0  1260759185
4       1     1172     4.0  1260759205
5       1     1263     2.0  1260759151
6       1     1287     2.0  1260759187
```

Let's count the number of entries in the tables for ratings and movies. [1]

```
movies.shape
```

```
(9125, 3)
```

```
ratings.shape
```

```
(100004, 4)
```

2

Romance vs. Scifi

Let's begin by examining the favourite genres of a small selection of users. The majority of the data preprocessing is placed in helper functions to keep main focus on clustering. After finishing this notebook, it would be helpful if you quickly review helper.py to see how these helpers are implemented.

Calculating the average rating for romance and science fiction films. [1]

```
genre_ratings = helper.get_genre_ratings(ratings, movies, ['Romance',
'Sci-Fi'], ['avg_romance_rating', 'avg_scifi_rating'])
genre_ratings.head()
```

```
        avg_romance_rating  avg_scifi_rating
userId
1                     3.50              2.40
2                     3.59              3.80
```

```
3                    3.65              3.14
4                    4.50              4.26
5                    4.08              4.00
```

Each user's average rating for all romantic movies and all sci-fi movies was determined using the `get_genre_ratings` function. Let's bias dataset by excluding individuals who prefer both sci-fi and romance so that our clusters try to categorize them as favoring one genre over the other. [1]

```
biased_dataset = helper.bias_genre_rating_dataset(genre_ratings, 3.2,
2.5)
print( "Number of records: ", len(biased_dataset))
biased_dataset.head()
```

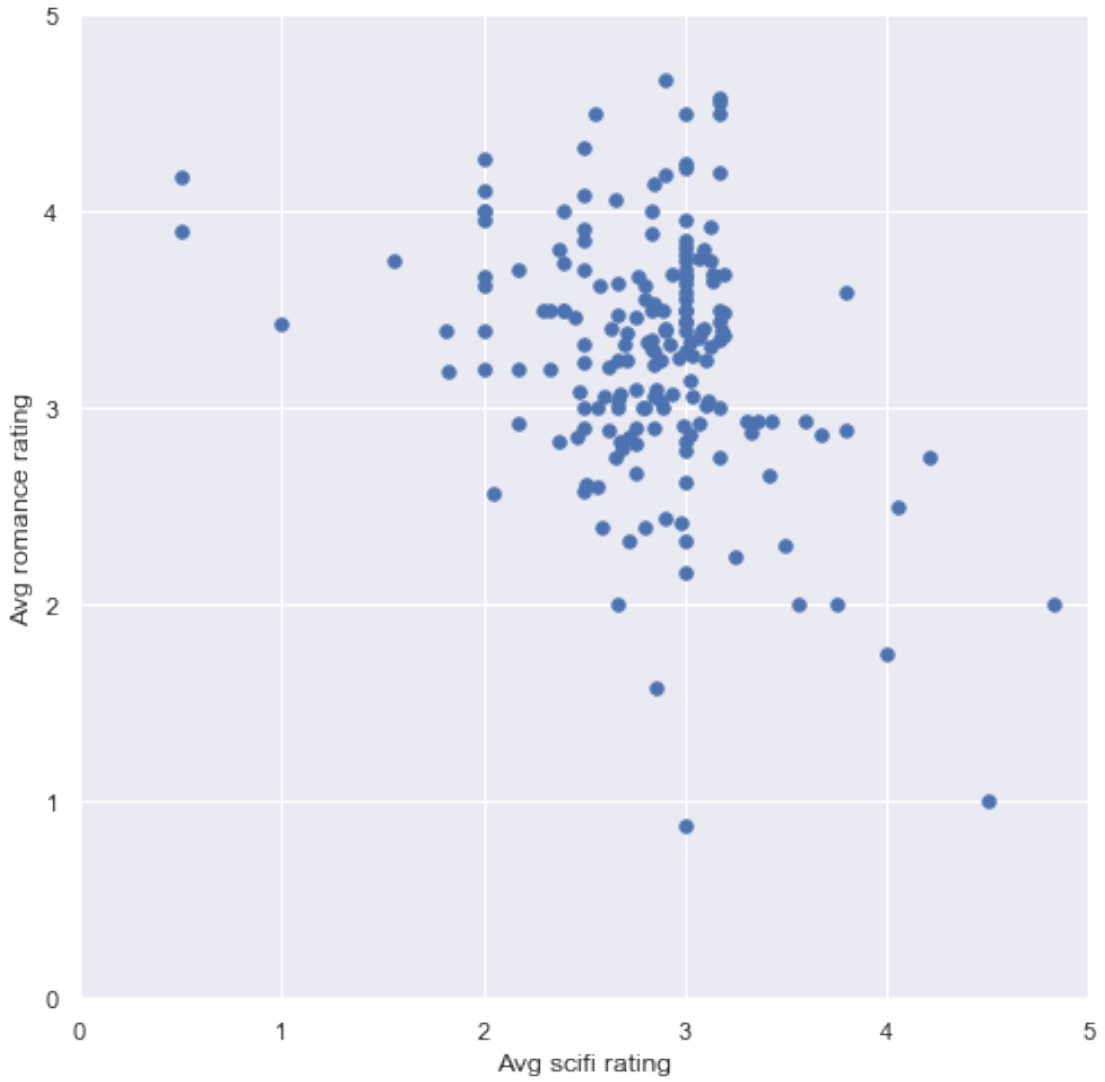
```
Number of records:  183
```

```
   userId  avg_romance_rating  avg_scifi_rating
0       1                3.50              2.40
1       3                3.65              3.14
2       6                2.90              2.75
3       7                2.93              3.36
4      12                2.89              2.62
```

We can see that there are 183 users and average rating of each user for the romance and sci-fi films they have seen.

Plotting this dataset will show:

```
%matplotlib inline
helper.draw_scatterplot(biased_dataset['avg_scifi_rating'],'Avg scifi
rating', biased_dataset['avg_romance_rating'], 'Avg romance rating')
```



This sample clearly has some bias (that we created on purpose). How would it look if we use k-means to divide the sample into two groups? [1]

```
# Let's convert our dataset to a list.
X = biased_dataset[['avg_scifi_rating','avg_romance_rating']].values
```

Import library **KMeans** [1]

```
from sklearn.cluster import KMeans
```

Create an instance of KMeans to find two clusters [1]

```
kmeans_1 = KMeans(n_clusters=2, random_state=0)
kmeans_1.fit(X)
```

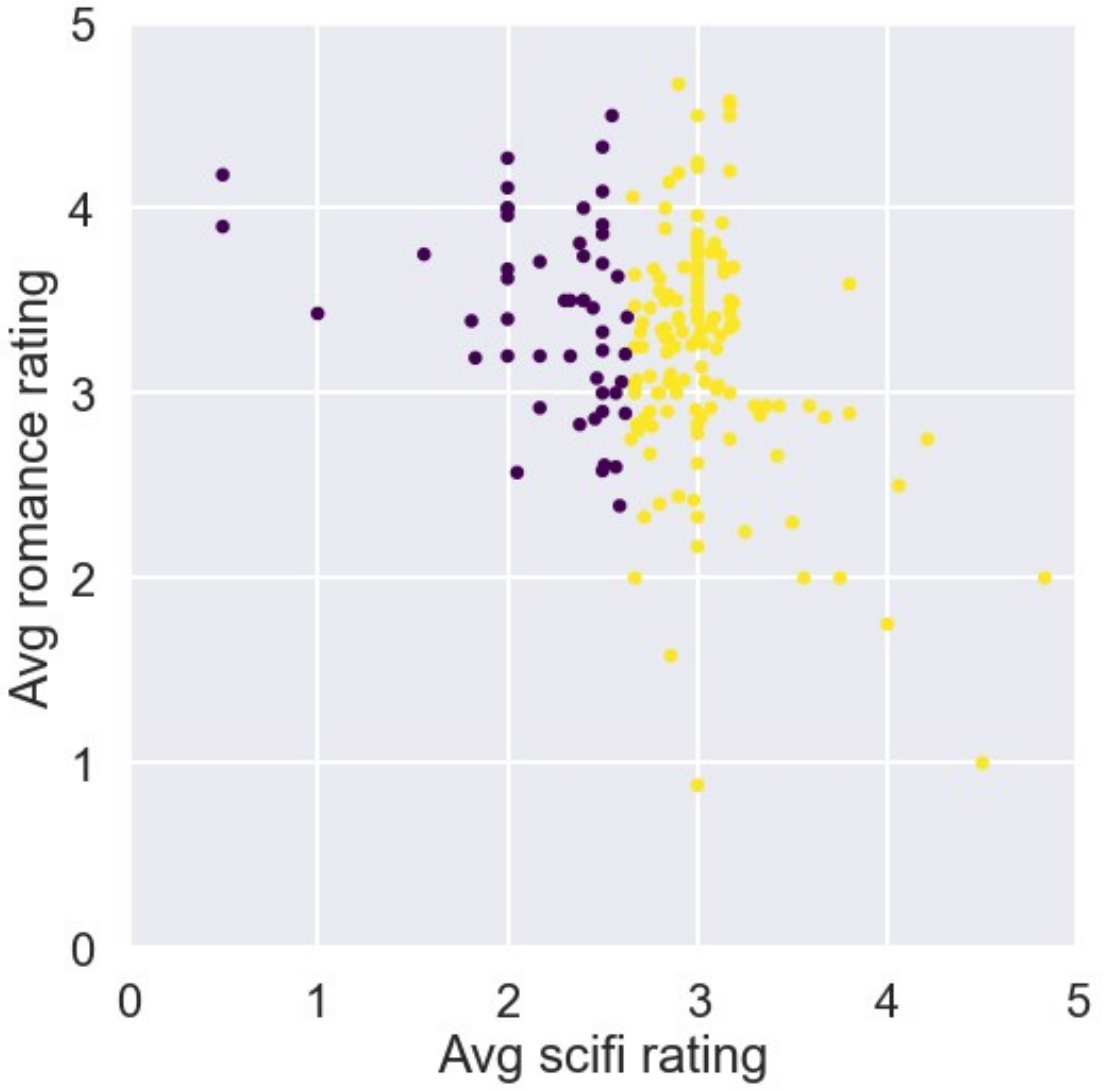
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

Cluster the dataset using fit_predict [1]

```
predictions = kmeans_1.fit_predict(X)
```

Display a plot [1]

```
helper.draw_clusters(biased_dataset, predictions)
```




```
predictions [1]
```

```
array([0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
0,
```

```
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
1,
       0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0,
       1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
1,
       0, 0, 0, 1, 0, 0, 0])
```

One cluster is from 2.0 to 2.5 while the other is from 2.5 to 3.2 in the sci-fi rating. The romance rating starts densing from 2.5 and goes to 4.2, with one cluster having fewer users and the other more.

It is clear that the groups mostly represent how each user rated romantic films. They fall into one category if their average rating for romantic films is more than three stars. They belong to the opposite group if not.

What would occur if we divide them into three groups?

Create an instance of KMeans to find three clusters [1]

```
kmeans_2 = KMeans(n_clusters=3, random_state=0)
kmeans_2.fit(X)
```

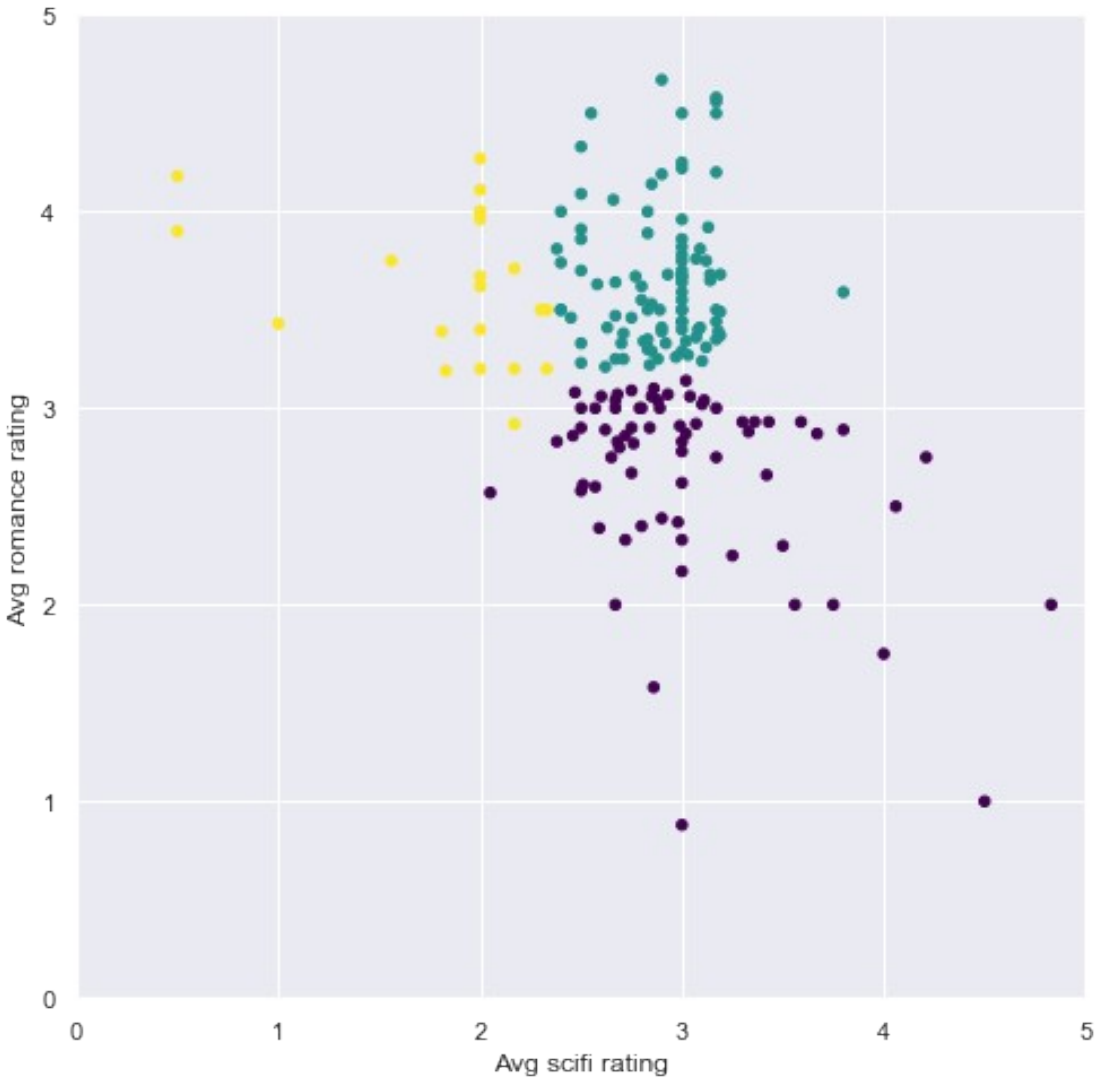
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

Use fit_predict to cluster the dataset [1]

```
predictions_2 = kmeans_2.fit_predict(X)
```

Display a plot [1]

```
helper.draw_clusters(biased_dataset, predictions_2)
```



- There are fewer users rated between 3.2 and 4.2 in the yellow cluster.
- In the sea-green cluster, several users had ratings between 3.3 and 4.3.
- As user density decreases, there are a lot of people rating close to 3 in the purple cluster.

Let's add one more group

Create an instance of KMeans to find four clusters [1]

```
kmeans_3 = KMeans(n_clusters=4, random_state=0)
kmeans_3.fit(X)
```

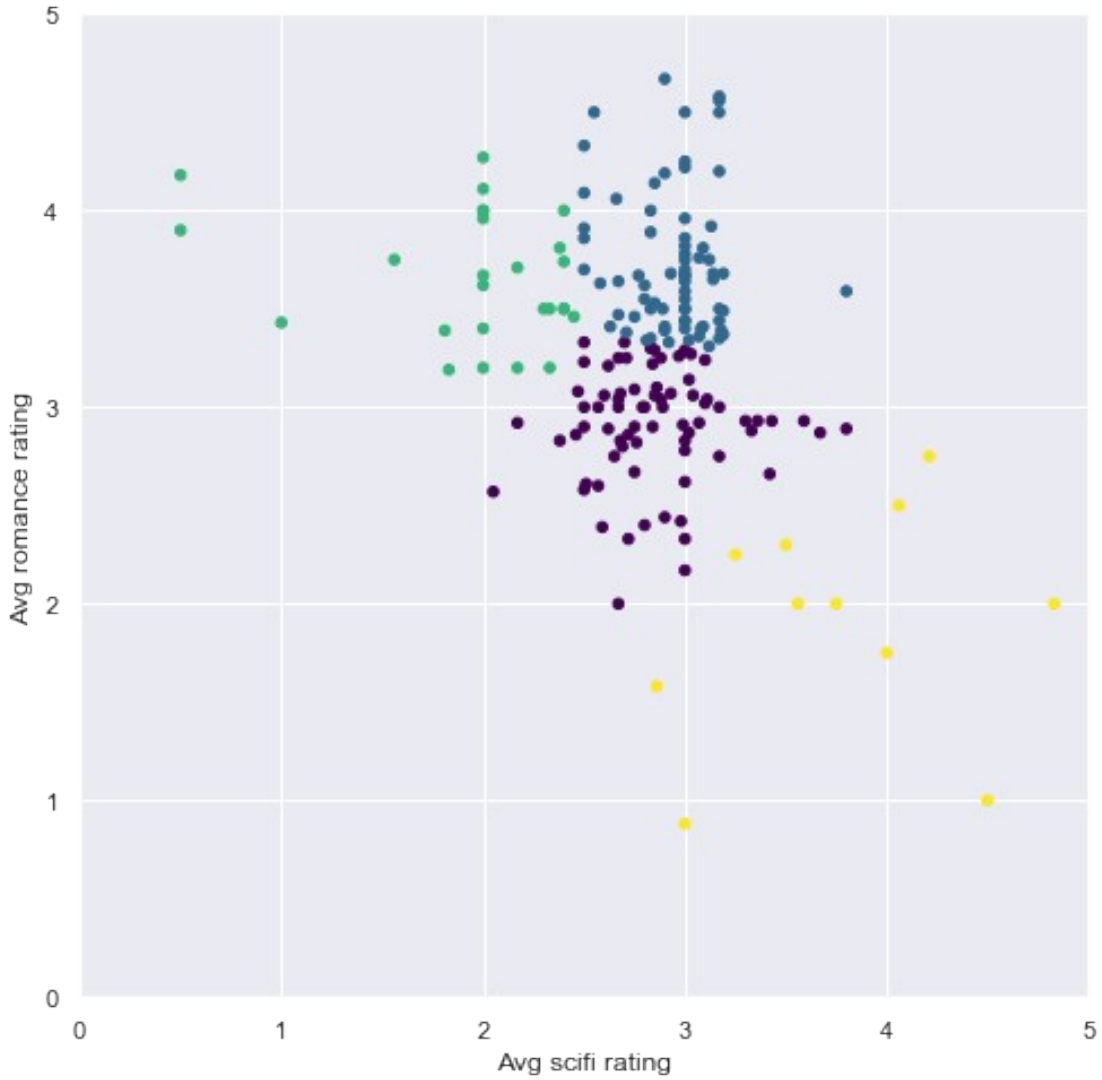
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

Use fit_predict to cluster the dataset [1]

```
predictions_3 = kmeans_3.fit_predict(X)
```

Display a plot [1]

```
helper.draw_clusters(biased_dataset, predictions_3)
```



- in green cluster they are smaller in number but highest rated
- in blue they are rated highest and large in number
- in purple they are rated medium and large in number
- in yellow they are less in number and also low rated

3

Silhouette Score

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

Import library **Silhouette_score**

```
from sklearn.metrics import silhouette_score

#task
s_score = []
for k in range(2, 183, 5):
    kmeans = KMeans(n_clusters=k, random_state=0).fit(X) #Create an
instance of KMeans to find k clusters with random_state 0 then fit
    s_score.append([k, silhouette_score(X, kmeans.labels_)])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\
k_means_.py:971: ConvergenceWarning: Number of distinct clusters (178)
found smaller than n_clusters (182). Possibly due to duplicate points
in X.
  return_n_iter=True)
```

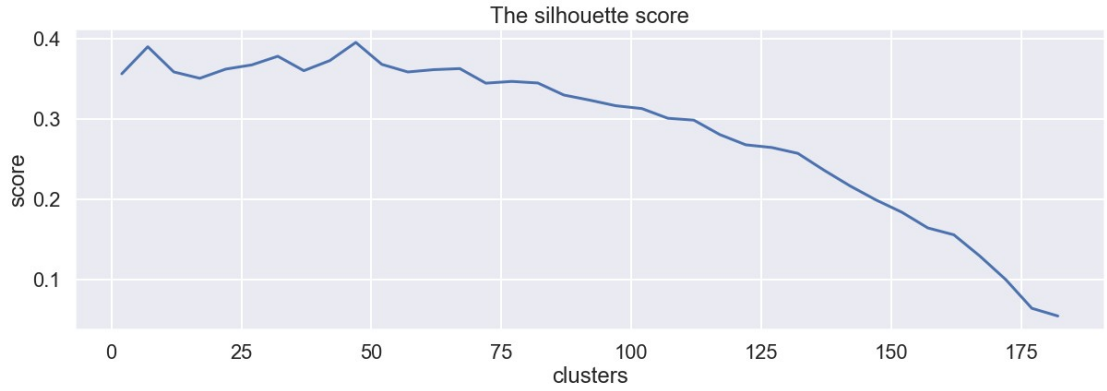
```
s_score
```

```
[[2, 0.3558817876472825],
 [7, 0.38953342844609656],
 [12, 0.3581974687657583],
 [17, 0.3502856136343623],
 [22, 0.3617421566757448],
 [27, 0.36693813543174936],
 [32, 0.3776980303487645],
 [37, 0.35975988674633536],
 [42, 0.372279308070513],
 [47, 0.39490936357676387],
 [52, 0.36759244077425984],
 [57, 0.3581625729665141],
 [62, 0.36103500283024587],
 [67, 0.3623546060941169],
 [72, 0.34422649399045513],
 [77, 0.34640246785935286],
 [82, 0.34444423754545006],
 [87, 0.32948512980500305],
 [92, 0.3230791029760401],
 [97, 0.31614756830696206],
 [102, 0.3126433371386654],
```

```
 [107, 0.3005833276430467],
 [112, 0.2982495255455439],
 [117, 0.2802650173585646],
 [122, 0.2675303015141549],
 [127, 0.26416818251997415],
 [132, 0.2569510743702076],
 [137, 0.2361166020024426],
 [142, 0.21659990220950903],
 [147, 0.19908335270651936],
 [152, 0.18372286145212785],
 [157, 0.1641628841708429],
 [162, 0.1556499460286296],
 [167, 0.12920960729951275],
 [172, 0.09984309419914657],
 [177, 0.0642301201631745],
 [182, 0.05464480874316941]]
```

```
import seaborn as sns
sns.set()
```

```
plt.figure(figsize=(20,6))
sns.set_context('poster')
plt.plot( pd.DataFrame(s_score)[0], pd.DataFrame(s_score)[1])
plt.xlabel('clusters')
plt.ylabel('score')
plt.title('The silhouette score')
plt.show()
```



Looking at this graph, So you should select the number of clusters that maximizes the silhouette coefficient good choices include 7, 47 amongst other values (with a slight variation between different runs).Our pick would be k=7 because it's easier to visualize: What would you prefer 7 or 47??

Create an instance of KMeans to find seven clusters

```
kmeans_4 = KMeans(n_clusters=7, random_state=0)
kmeans_4.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=7, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

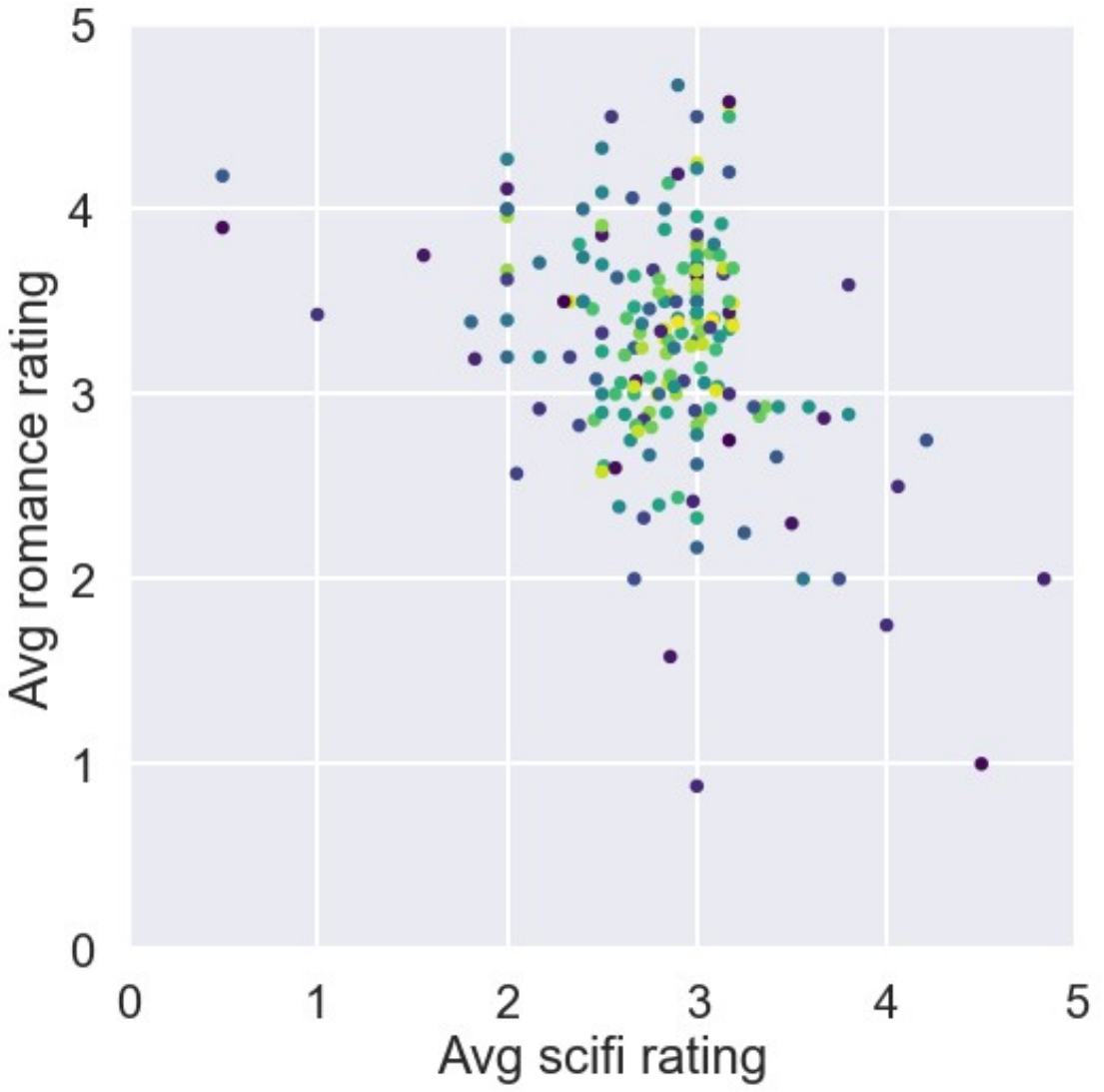
Use fit_predict to cluster the dataset

```
predictions_4 = kmeans.fit_predict(X)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\
k_means_.py:971: ConvergenceWarning: Number of distinct clusters (178)
found smaller than n_clusters (182). Possibly due to duplicate points
in X.
  return_n_iter=True)
```

Display a plot

```
helper.draw_clusters(biased_dataset, predictions_4)
```



- yellow is less despersed and rating between 2.8 to 3.8 and average in numbers of users
- green is less despersed and rating between 2.8 to 4.0 and many in numbers
- purple is despersed and low in numbers rating between 1.0 to 4.5
- sea-green is large in number and also despersed between 2.0 to 4.2

4

Throwing some Action into the mix

Until now we've only been focusing at how users rated romance and scifi movies. Let's include a different genre in the mix. Add the Action category.

Now, our dataset appears as follows:

```
biased_dataset_3_genres = helper.get_genre_ratings(ratings, movies,
                                                   ['Romance', 'Sci-
```

```
Fi', 'Action'],

['avg_romance_rating', 'avg_scifi_rating', 'avg_action_rating'])
biased_dataset_3_genres =
helper.bias_genre_rating_dataset(biased_dataset_3_genres, 3.2,
2.5).dropna()

print( "Number of records: ", len(biased_dataset_3_genres))
biased_dataset_3_genres.head()
```

```
Number of records:  183
```

	userId	avg_romance_rating	avg_scifi_rating	avg_action_rating
0	1	3.50	2.40	2.80
1	3	3.65	3.14	3.47
2	6	2.90	2.75	3.27
3	7	2.93	3.36	3.29
4	12	2.89	2.62	3.21

```
X_with_action = biased_dataset_3_genres[['avg_scifi_rating',

'avg_romance_rating',

'avg_action_rating']].values
```

Create an instance of KMeans to find seven clusters

```
kmeans_5 = KMeans(n_clusters=7, random_state=0)
kmeans_5
```

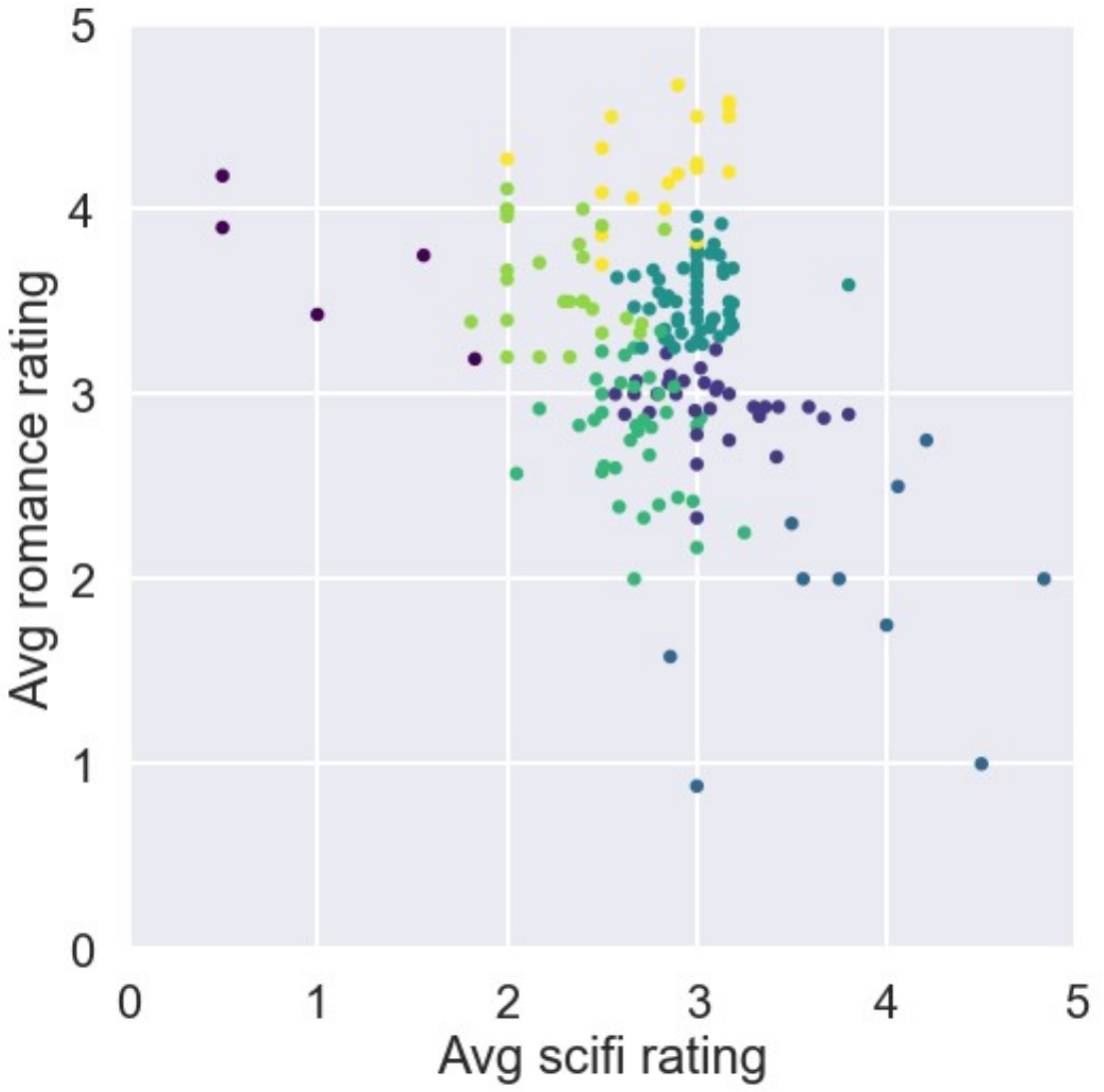
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=7, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

Use fit_predict to cluster the dataset 'X_with_action'

```
predictions_5 = kmeans_5.fit_predict(X_with_action)
```

Display a plot

```
helper.draw_clusters(biased_dataset_3_genres, predictions_5)
```



We're still using the x and y axes for scifi and romance respectively. We are using the size of the dot to roughly code the 'action' rating (large dot for avg ratings over than 3, small dot otherwise).

5

Multi Level Clustering

Let's take a bigger bite and look at how users rated certain movies now that we have some confidence in how k-means clusters users depending on their genre preferences. To do this, we will structure the dataset so that it relates to each movie's userId and user rating. Let's study a subset of the dataset, for illustration:

```
# Merge the two tables then pivot so we have Users X Movies dataframe
ratings_title = pd.merge(ratings, movies[['movieId', 'title']],
on='movieId' )
user_movie_ratings = pd.pivot_table(ratings_title, index='userId',
```

```
columns= 'title', values='rating')

print('dataset dimensions: ', user_movie_ratings.shape, '\n\nSubset
example:')
user_movie_ratings.iloc[:6, :10]
```

```
dataset dimensions:  (671, 9064)

Subset example:
```

```
title   "Great Performances" Cats (1998)  $9.99 (2008)  \
userId
1                                      NaN           NaN
2                                      NaN           NaN
3                                      NaN           NaN
4                                      NaN           NaN
5                                      NaN           NaN
6                                      NaN           NaN

title   'Hellboy': The Seeds of Creation (2004)  \
userId
1                                           NaN
2                                           NaN
3                                           NaN
4                                           NaN
5                                           NaN
6                                           NaN

title   'Neath the Arizona Skies (1934)  'Round Midnight (1986)  \
userId
1                                    NaN                     NaN
2                                    NaN                     NaN
3                                    NaN                     NaN
4                                    NaN                     NaN
5                                    NaN                     NaN
6                                    NaN                     NaN

title   'Salem's Lot (2004)  'Til There Was You (1997)  'burbs, The
(1989)  \
userId

1                       NaN                       NaN
NaN
2                       NaN                       NaN
NaN
3                       NaN                       NaN
NaN
4                       NaN                       NaN
NaN
5                       NaN                       NaN
```

```
NaN
6                        NaN                        NaN
4.0

title   'night Mother (1986)  (500) Days of Summer (2009)
userId
1                         NaN                           NaN
2                         NaN                           NaN
3                         NaN                           NaN
4                         NaN                           NaN
5                         NaN                           NaN
6                         NaN                           NaN
```

The dominance of NaN values presents the first issue. Most users have not rated and watched most movies. Datasets like this are called "sparse" because only a small number of cells have values.

To get around this, let's sort by the most rated movies, and the users who have rated the most number of movies. That will present a more 'dense' region when we peak at the top of the dataset.

If we're to choose the most-rated movies vs users with the most ratings, it would look like this:

```
n_movies = 30
n_users = 18
most_rated_movies_users_selection =
helper.sort_by_rating_density(user_movie_ratings, n_movies, n_users)

print('dataset dimensions: ', most_rated_movies_users_selection.shape)
most_rated_movies_users_selection.head()
```

```
dataset dimensions:  (18, 30)

title  Forrest Gump (1994)  Pulp Fiction (1994)  \
29                     5.0                  5.0
508                    4.0                  5.0
14                     1.0                  5.0
72                     5.0                  5.0
653                    4.0                  5.0

title  Shawshank Redemption, The (1994)  Silence of the Lambs, The
(1991)  \
29                                   5.0
4.0
508                                  4.0
4.0
14                                   2.0
5.0
72                                   5.0
4.5
```

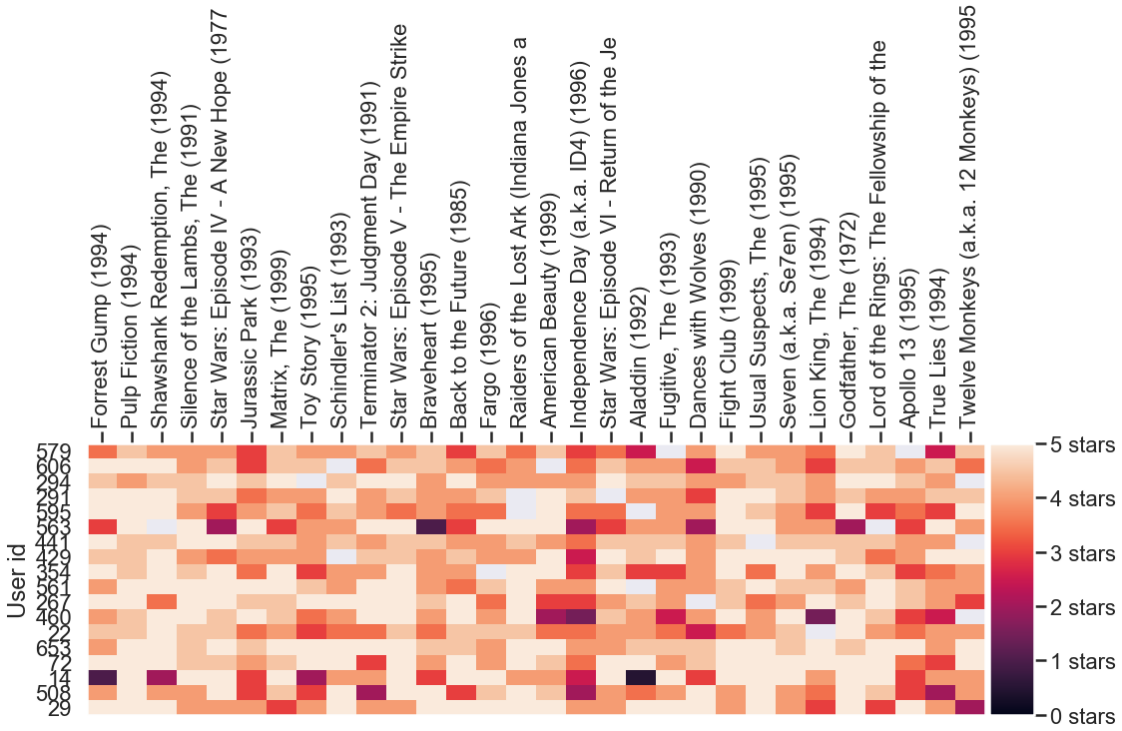
```
653                                 5.0
4.5

title  Star Wars: Episode IV - A New Hope (1977)  Jurassic Park (1993)
\
29                                          4.0                   4.0

508                                         5.0                   3.0

14                                          5.0                   3.0

72                                          4.5                   4.0

653                                         5.0                   4.5


title  Matrix, The (1999)  Toy Story (1995)  Schindler's List
(1993)  \
29                    3.0               4.0                        5.0

508                   4.5               3.0                        5.0

14                    5.0               2.0                        4.0

72                    4.5               5.0                        5.0

653                   5.0               5.0                        5.0


title  Terminator 2: Judgment Day (1991)  \
29                                    4.0
508                                   2.0
14                                    4.0
72                                    3.0
653                                   5.0

title                     ...                      Dances with Wolves
(1990)  \
29                        ...
5.0
508                       ...
5.0
14                        ...
3.0
72                        ...
4.5
653                       ...
4.5
```

```
title  Fight Club (1999)  Usual Suspects, The (1995)  \
29                   4.0                         5.0
508                  4.0                         5.0
14                   5.0                         5.0
72                   5.0                         5.0
653                  5.0                         5.0

title  Seven (a.k.a. Se7en) (1995)  Lion King, The (1994)  \
29                              4.0                    3.0
508                             4.0                    3.5
14                              5.0                    4.0
72                              5.0                    5.0
653                             4.5                    5.0

title  Godfather, The (1972)  \
29                        5.0
508                       5.0
14                        5.0
72                        5.0
653                       4.5

title  Lord of the Rings: The Fellowship of the Ring, The (2001)  \
29                                                           3.0
508                                                          4.5
14                                                           5.0
72                                                           5.0
653                                                          5.0

title  Apollo 13 (1995)  True Lies (1994)  \
29                   5.0                4.0
508                  3.0                2.0
14                   3.0                4.0
72                   3.5                3.0
653                  5.0                4.0

title  Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
29                                           2.0
508                                          4.0
14                                           4.0
72                                           5.0
653                                          5.0

[5 rows x 30 columns]
```

That's more like it. Let's also establish a good way for visualizing these ratings so we can attempt to visually recognize the ratings (and later, clusters) when we look at bigger subsets.

Let's use colors instead of the number ratings:

```
helper.draw_movies_heatmap(most_rated_movies_users_selection);
```



Each column is a movie. Each row is a user. The color of the cell is how the user rated that movie based on the scale on the right of the graph.

Notice how some cells are white? This means the respective user did not rate that movie. This is an issue you'll come across when clustering in real life. Unlike the clean example we started with, real-world datasets can often be sparse and not have a value in each cell of the dataset. This makes it less straightforward to cluster users directly by their movie ratings as k-means generally does not like missing values.

For performance reasons, we'll only use ratings for 1000 movies (out of the 9000+ available in the dataset).

```
user_movie_ratings =  pd.pivot_table(ratings_title, index='userId',
columns= 'title', values='rating')
most_rated_movies_1k =
helper.get_most_rated_movies(user_movie_ratings, 1000)
```

To have sklearn run k-means clustering to a dataset with missing values like this, we will first cast it to the sparse csr matrix type defined in the SciPi library.

To convert from a pandas dataframe to a sparse matrix, we'll have to convert to SparseDataFrame, then use pandas' `to_coo()` method for the conversion.

Note: `to_coo()` was only added in later versions of pandas. If you run into an error with the next cell, make sure pandas is up to date.

```
#import csr_matrix from scipy.sparse
from scipy.sparse import csr_matrix

sparse_ratings =
csr_matrix(pd.SparseDataFrame(most_rated_movies_1k).to_coo())
```

6

Let's cluster!

With k-means, we have to provide specify k, the number of clusters. Let's try k=10

Task Make 10 clusters of the Sparsed Data Set and apply fit_predict() function on it.

```
sparse_ratings
```

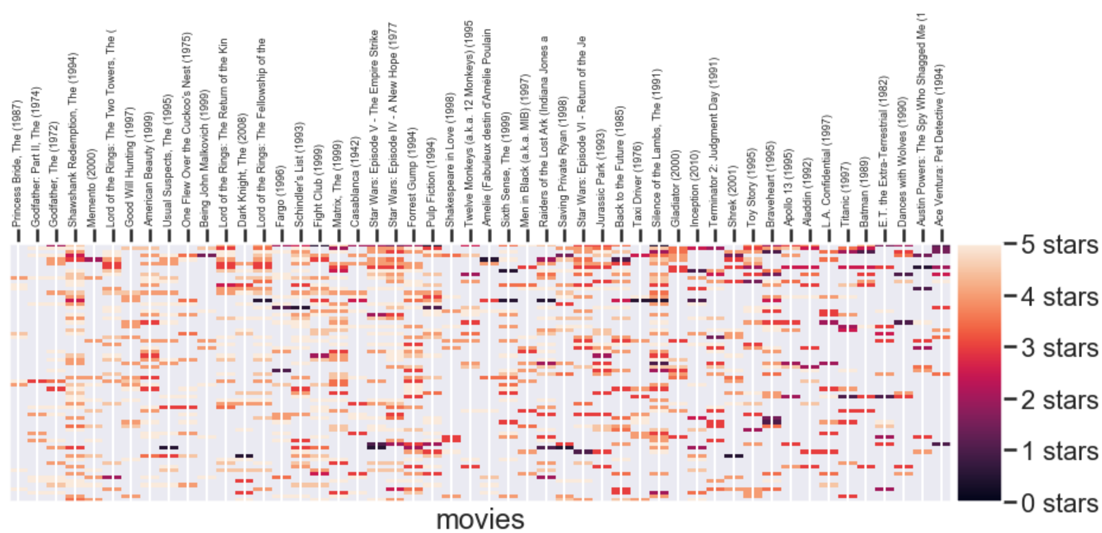
```
<671x1000 sparse matrix of type '<class 'numpy.float64'>'
     with 62397 stored elements in Compressed Sparse Row format>
```

```
#todo, write code here
predictions = KMeans(n_clusters=10,
random_state=0).fit_predict(sparse_ratings)
predictions
```

```
array([7, 0, 7, 1, 4, 7, 1, 4, 7, 7, 7, 7, 7, 7, 8, 7, 3, 2, 5, 7, 1,
4,
       5, 7, 2, 4, 7, 7, 7, 5, 4, 0, 7, 1, 7, 0, 7, 4, 0, 7, 1, 4, 7,
2,
       7, 7, 0, 3, 7, 0, 7, 7, 7, 7, 2, 3, 9, 7, 7, 7, 4, 7, 1, 0, 7,
7,
       0, 4, 4, 2, 7, 4, 8, 7, 1, 7, 6, 3, 7, 2, 7, 0, 1, 4, 0, 0, 2,
6,
       4, 2, 4, 0, 4, 4, 1, 7, 1, 7, 1, 2, 7, 5, 7, 4, 6, 7, 7, 0, 7,
0,
       6, 2, 7, 0, 7, 7, 7, 1, 5, 1, 0, 7, 7, 4, 3, 0, 7, 4, 7, 6, 7,
7,
       7, 6, 7, 7, 1, 4, 7, 7, 7, 7, 7, 0, 0, 4, 7, 1, 4, 6, 0, 3, 7,
2,
       1, 7, 3, 7, 1, 7, 0, 2, 7, 1, 6, 7, 7, 0, 4, 7, 7, 7, 2, 7, 4,
4,
       1, 1, 7, 7, 7, 0, 7, 0, 1, 7, 4, 0, 7, 7, 0, 0, 1, 0, 5, 7, 7,
7,
       3, 4, 4, 7, 7, 7, 4, 7, 7, 7, 7, 7, 2, 8, 4, 1, 7, 4, 1, 7, 1,
1,
       7, 7, 2, 0, 0, 7, 7, 7, 7, 7, 7, 5, 2, 4, 1, 1, 7, 7, 1, 4, 0,
5,
       6, 7, 1, 7, 6, 1, 7, 4, 4, 2, 6, 0, 7, 2, 7, 7, 7, 7, 7, 3, 7,
7,
       1, 0, 7, 6, 7, 4, 1, 7, 4, 7, 4, 7, 7, 7, 0, 7, 1, 4, 4, 2, 6,
7,
       4, 0, 7, 7, 1, 6, 7, 6, 6, 7, 1, 7, 1, 7, 7, 0, 6, 1, 7, 9, 7,
7,
```

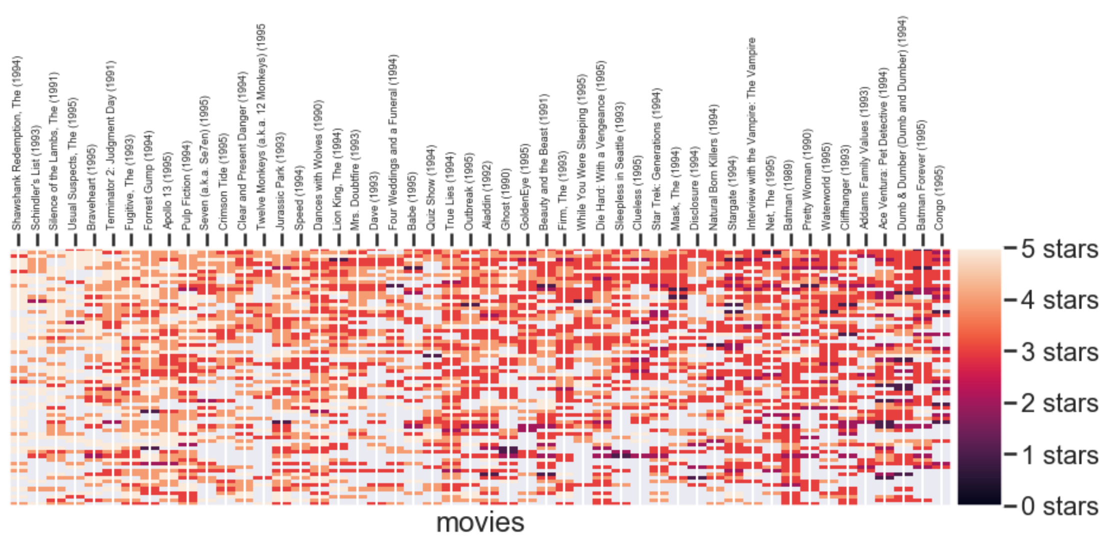
```
       1, 7, 5, 1, 4, 7, 7, 4, 0, 2, 7, 7, 7, 7, 7, 4, 7, 7, 7, 1, 2,
7,
       7, 7, 4, 7, 7, 7, 7, 0, 4, 7, 7, 1, 2, 0, 1, 1, 7, 7, 7, 1, 7,
7,
       3, 0, 3, 7, 7, 5, 7, 7, 2, 4, 1, 7, 7, 7, 1, 7, 0, 1, 4, 7, 1,
7,
       0, 7, 7, 7, 4, 8, 7, 4, 2, 3, 0, 7, 1, 5, 0, 2, 7, 7, 7, 4, 7,
0,
       7, 7, 7, 0, 1, 4, 7, 7, 1, 7, 1, 1, 0, 7, 7, 1, 2, 7, 0, 0, 4,
4,
       7, 1, 0, 1, 3, 7, 7, 4, 7, 4, 7, 6, 6, 4, 3, 1, 7, 7, 7, 7, 1,
0,
       1, 6, 7, 7, 7, 7, 0, 7, 0, 4, 7, 5, 7, 7, 7, 4, 3, 4, 2, 1, 6,
7,
       6, 7, 7, 1, 7, 5, 7, 7, 4, 5, 4, 7, 3, 7, 7, 4, 4, 3, 3, 7, 4,
2,
       7, 1, 0, 7, 7, 2, 7, 7, 7, 1, 0, 0, 1, 7, 7, 4, 4, 1, 7, 7, 5,
7,
       2, 0, 5, 1, 7, 7, 0, 5, 7, 0, 0, 9, 1, 4, 7, 4, 4, 7, 1, 7, 1,
1,
       5, 2, 4, 7, 1, 0, 0, 0, 5, 7, 7, 7, 7, 4, 2, 3, 7, 7, 5, 0, 7,
1,
       7, 7, 4, 7, 0, 0, 7, 1, 1, 1, 6, 6, 4, 5, 7, 7, 7, 0, 1, 4, 7,
7,
       7, 3, 5, 7, 6, 7, 7, 8, 7, 7, 7, 4, 5, 7, 5, 0, 0, 0, 7, 1, 0,
1,
       1, 3, 1, 6, 4, 7, 7, 0, 1, 7, 1, 7, 6, 1, 7, 7, 7, 7, 7, 7, 3,
2,
       7, 7, 0, 4, 7, 7, 4, 8, 7, 1, 4, 7, 0, 7, 7, 7, 0, 7, 7, 2, 7,
7,
       0, 2, 0, 7, 7, 7, 7, 1, 1, 4, 0, 2, 7, 7, 7, 8, 4, 1, 7, 7, 0,
4,
       7, 0, 7, 3, 6, 0, 0, 7, 7, 7, 1])

max_users = 70
max_movies = 50

clustered = pd.concat([most_rated_movies_1k.reset_index(),
pd.DataFrame({'group':predictions})], axis=1)
helper.draw_movie_clusters(clustered, max_users, max_movies)

cluster # 7
# of users in cluster: 296. # of users in plot: 70

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



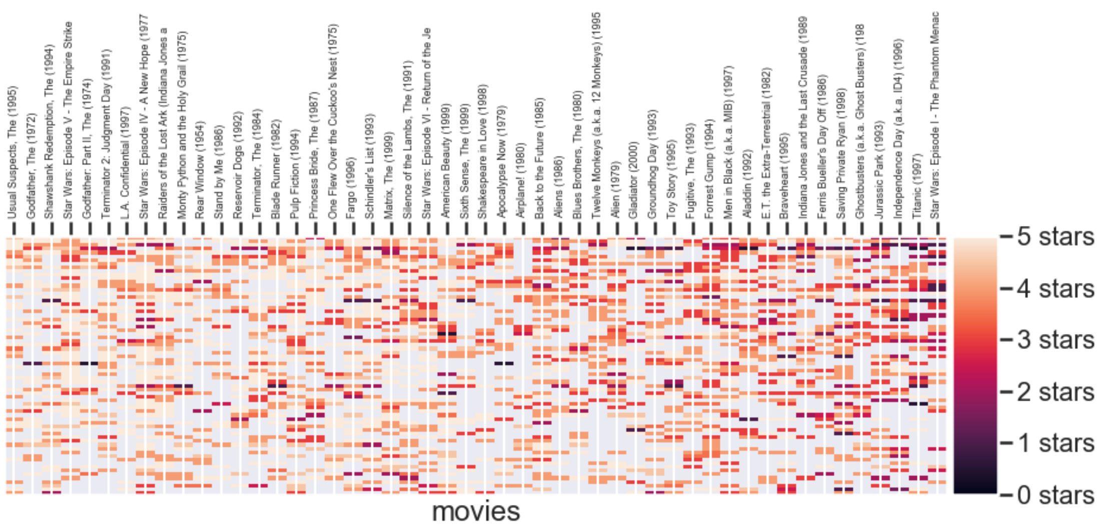

```
cluster # 0
# of users in cluster: 82. # of users in plot: 70

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



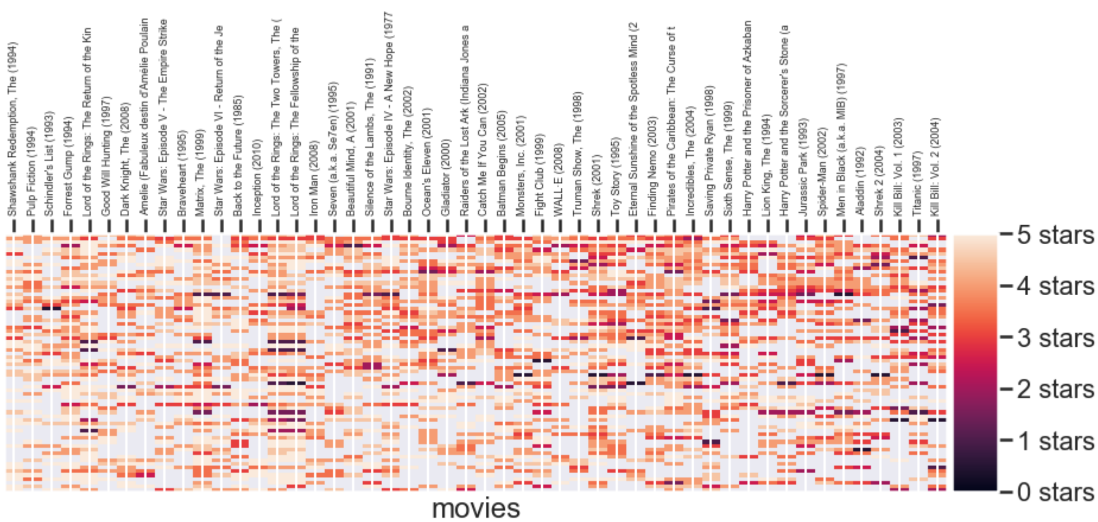

```
cluster # 1
# of users in cluster: 89. # of users in plot: 70

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



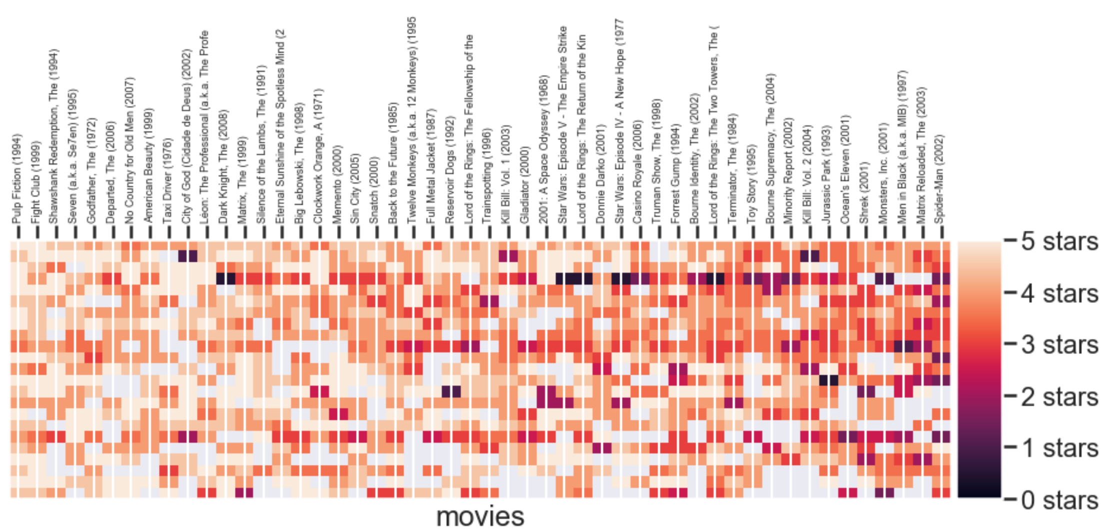

```
cluster # 4
# of users in cluster: 83. # of users in plot: 70

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



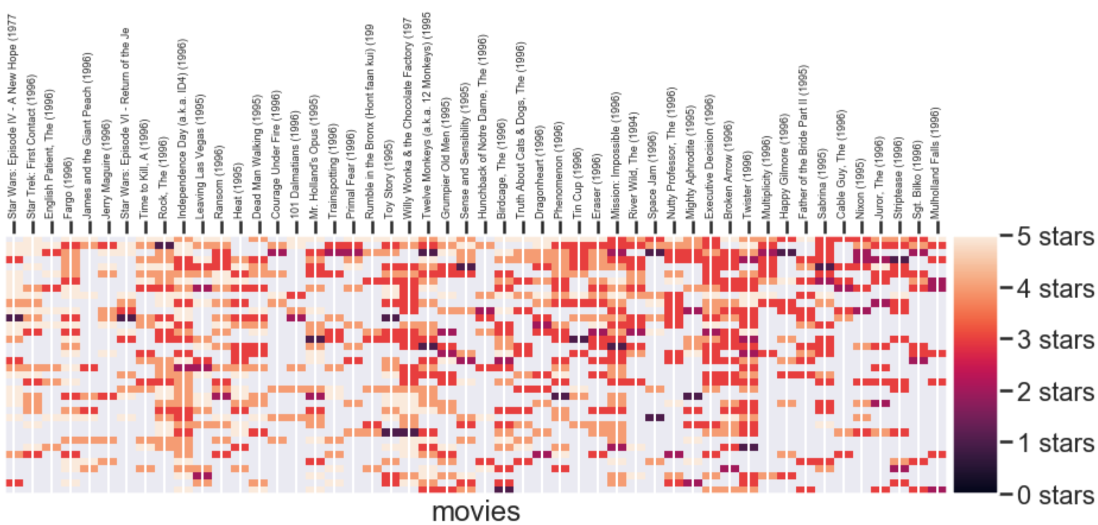

```
cluster # 3
# of users in cluster: 23. # of users in plot: 23

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



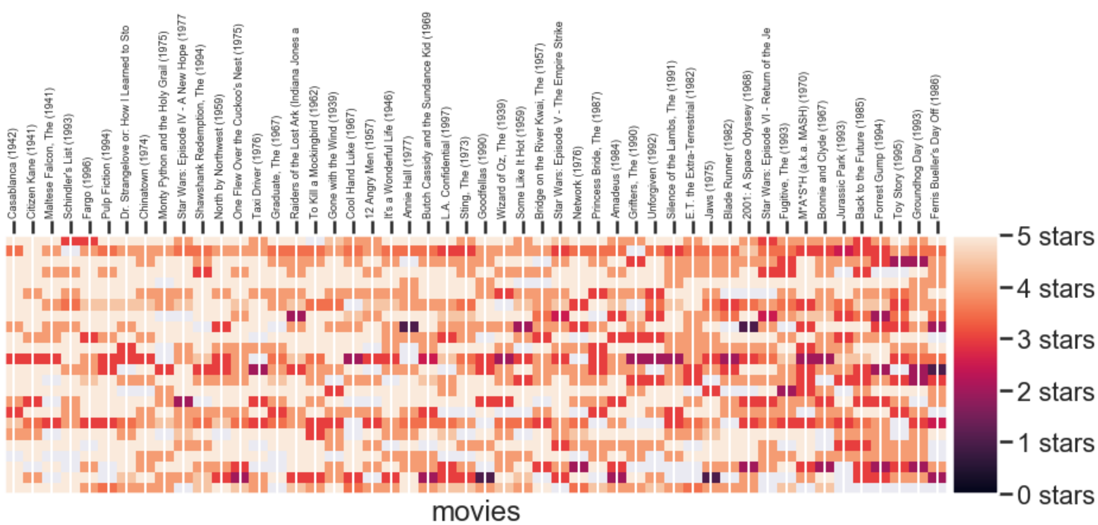

```
cluster # 2
# of users in cluster: 36. # of users in plot: 36

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



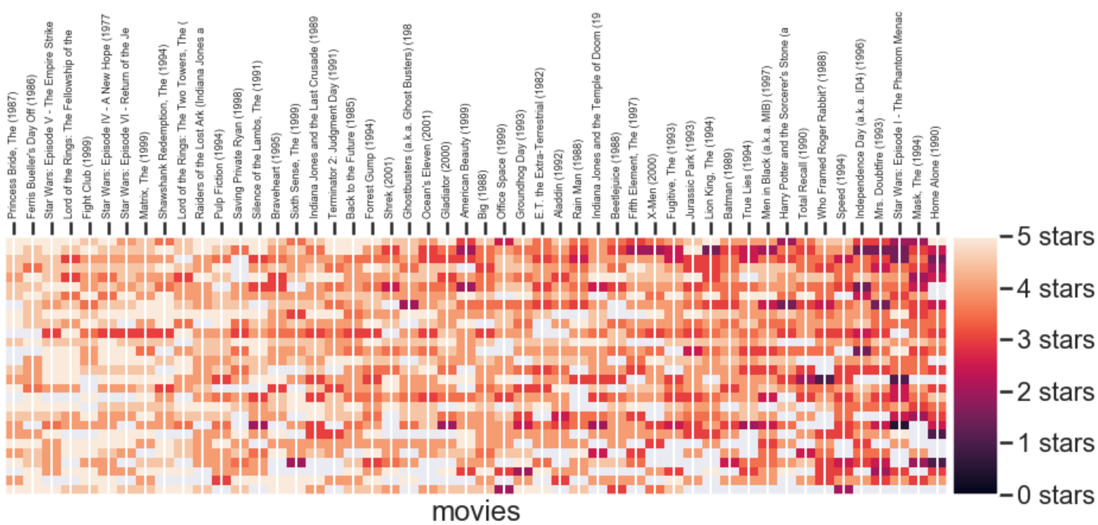

```
cluster # 5
# of users in cluster: 24. # of users in plot: 24

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```




```
cluster # 6
# of users in cluster: 28. # of users in plot: 28

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
```

```
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\
__init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is
deprecated; use an actual boolean (True/False) instead.
  warn_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false'
as a "
```



Here are a few things to take note of: *The more vertical lines of similar colors you can trace in a cluster, the more similar the ratings in that cluster are.

- Finding patterns in clusters is quite interesting:
- There are certain clusters that are sparser than others, suggesting that the people in such clusters presumably see and rate fewer movies overall.

- Some clusters are mostly peach and assemble fans of a certain genre of films. Other clusters, which are primarily orange or pink, involve users who feel that a certain collection of movies is deserving of 2-3 stars.
- Take note of how the movies differ in each cluster. The graph sorts the movies by average rating after filtering the data to only display the highest rated titles.
- Can you identify which clusters contain the Lord of the Rings movies? How about the Star Wars films?
- **Horizontal** lines with similar hues are simple to identify as belonging to users whose ratings are not very diverse. This is probably one of the factors behind Netflix's decision to switch from star ratings to thumbs-up/thumbs-down ratings. Four stars can symbolize many things to various individuals.
- We used many techniques (filtering, sorting, and slicing) to make the clusters visible. Due to the "sparse" nature of these datasets and the fact that the majority of the cells lack a value,

7

Prediction

Let's choose a cluster and a single user to discover what valuable things this clustering enables us to perform.

Let's begin with a cluster:

```
# TODO: Pick a cluster ID from the clusters above
cluster_number = 2

# Let's filter to only see the region of the dataset with the most
number of values
n_users = 75
n_movies = 300
cluster = clustered[clustered.group == cluster_number].drop(['index',
'group'], axis=1)

cluster = helper.sort_by_rating_density(cluster, n_movies, n_users)
helper.draw_movies_heatmap(cluster, axis_labels=False)
```

```
cluster.fillna('').head()
```

```
    Independence Day (a.k.a. ID4) (1996) Mission: Impossible (1996)  \
24                                     5                           2
30                                     3
4                                      4                           3
22                                     5                           3
23                                     4                           4

    Twister (1996) Twelve Monkeys (a.k.a. 12 Monkeys) (1995)  \
24              4                                         1
30              4
4               4                                         4
22              5                                         3
23              3                                         5

    Willy Wonka & the Chocolate Factory (1971) Rock, The (1996)  \
24                                                             4
30                                         4                   4
4                                          5                   5
22                                         4                   1
23                                         3                   3

    Executive Decision (1996) Fargo (1996) Toy Story (1995)  \
24                         4            4                 4
30                                      5                 5
4                          3            5                 5
22                         3            3                 3
23                         4            5                 5

    Birdcage, The (1996)           ...           Midnight Cowboy (1969)
\
24                     5           ...

30                                 ...

4                      5           ...

22                                 ...

23                     3           ...


    Thomas Crown Affair, The (1999) Fried Green Tomatoes (1991)  \
24
30
4
22
23
```

```
   Lethal Weapon 3 (1992) Remember the Titans (2000) Flintstones, The
(1994)  \
24

30

4

22

23


   What Lies Beneath (2000) Donnie Brasco (1997) Insomnia (2002)  \
24
30
4
22
23

   The Hunger Games (2012)
24
30
4
22
23

[5 rows x 300 columns]
```

Pick a blank cell from the table. It's blank because that user did not rate that movie. Can we predict whether she would like it or not? Since the user is in a cluster of users that seem to have similar taste, we can take the average of the votes for that movie in this cluster, and that would be a reasonable predicition for much she would enjoy the film.

```
# TODO: Fill in the name of the column/movie. e.g. 'Forrest Gump
(1994)'
movie_name = 'Executive Decision (1996)'

cluster[movie_name].mean()
```

```
3.4615384615384617
```

And this is our prediction how she will rate the film.

8

Recommendation

Let's go through everything we performed in the preceding phase once again. K-means has been used to cluster users based on their ratings. This helped us identify groups of people who had comparable ratings and, as a result, typically shared movie preferences. Based on this, we estimated how one user would feel about a movie when they had not yet given it a rating by averaging the ratings of all the other users in the cluster.

By applying this logic, we might determine the average score in this cluster for each movie and gain insight into how this "taste cluster" perceives each film in the dataset.

```
# The average rating of 20 movies as rated by the users in the cluster
cluster.mean().head(20)
```

```
Independence Day (a.k.a. ID4) (1996)          3.970588
Mission: Impossible (1996)                    3.533333
Twister (1996)                                3.366667
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)     3.714286
Willy Wonka & the Chocolate Factory (1971)    3.750000
Rock, The (1996)                              4.000000
Executive Decision (1996)                     3.461538
Fargo (1996)                                  4.269231
Toy Story (1995)                              3.760000
Birdcage, The (1996)                          3.666667
Broken Arrow (1996)                           3.391304
Phenomenon (1996)                             3.608696
Star Wars: Episode IV - A New Hope (1977)     4.363636
Mr. Holland's Opus (1995)                     3.863636
Ransom (1996)                                 3.904762
Heat (1995)                                   3.904762
Eraser (1996)                                 3.550000
Sabrina (1995)                                3.200000
Star Trek: First Contact (1996)               4.350000
Grumpier Old Men (1995)                       3.700000
dtype: float64
```

This becomes really useful for us because we can now use it as a recommendation engine that enables our users to discover movies they're likely to enjoy.

When a user logs in to our app, we can now show them recommendations that are appropriate to their taste. The formula for these recommendations is to select the cluster's highest-rated movies that the user did not rate yet.

```
# TODO: Pick a user ID from the dataset
# Look at the table above outputted by the command
"cluster.fillna('').head()"
# and pick one of the user ids (the first column in the table)
user_id = 3
```

```
# Get all this user's ratings
user_2_ratings  = cluster.loc[user_id, :]

# Which movies did they not rate? (We don't want to recommend movies
they've already rated)
user_2_unrated_movies =  user_2_ratings[user_2_ratings.isnull()]

# What are the ratings of these movies the user did not rate?
avg_ratings = pd.concat([user_2_unrated_movies, cluster.mean()],
axis=1, join='inner').iloc[:,-1]

# Let's sort by rating so the highest rated movies are presented first
avg_ratings.sort_values(ascending=False)[:20]
```

```
Snow White and the Seven Dwarfs (1937)                    5.0
Schindler's List (1993)                                   5.0
Fantasia (1940)                                           5.0
Speed (1994)                                              5.0
Grosse Pointe Blank (1997)                                5.0
Seven (a.k.a. Se7en) (1995)                               5.0
Good Will Hunting (1997)                                  5.0
African Queen, The (1951)                                 5.0
Contact (1997)                                            5.0
Casino (1995)                                             5.0
Star Trek II: The Wrath of Khan (1982)                    5.0
Titanic (1997)                                            5.0
Jurassic Park (1993)                                      5.0
Swingers (1996)                                           5.0
Gandhi (1982)                                             5.0
Wallace & Gromit: The Best of Aardman Animation (1996)    5.0
Hercules (1997)                                           5.0
Sling Blade (1996)                                        5.0
Star Trek IV: The Voyage Home (1986)                      5.0
Butch Cassidy and the Sundance Kid (1969)                 5.0
Name: 0, dtype: float64
```

```
user_2_ratings.isnull().sum()
```

```
269
```

```
ratings[ratings['movieId']== 987]
```

```
       userId  movieId  rating  timestamp
72474     509      987     3.0  940013699
```

```
### 9
### References
1. https://programming.rhysshea.com/K-means_movie_ratings/
2. https://notebooks.githubusercontent.com/view/ipynb?
3. https://towardsdatascience.com/introduction-to-recommender-systems-
6c66cf15ada
```