Recommender Exercise

April 7, 2019

Start:

movies.csv

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
...	...	...

ratings.csv

userId	movieId	rating	timestamp
1	1	4.0	964982703
1	3	4.0	965981247
1	6	4.0	964982224
1	47	5.0	964983815
1	50	5.0	964982931
1	70	3.0	964982400
1	101	5.0	964980868
1	110	4.0	964982176
...	...	...	...

Goal 1

Description: a merge table containing all our data at once from which we can drop various columns later

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
5	18	1	3.5	1455209816	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
6	19	1	4.0	965705637	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
7	21	1	3.5	1407618878	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
...	...	...	...	...	...	...

Differences:

Process:

Goal 2A

Description: A title/genre table to show the user alongside their results (this will have a lot of dupliates so remember to drop.na)

	title	genres
0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
5	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
6	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
7	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
...	...	...

Differences:

Process:

Goal 2B

Description: We don't need duplicates just to show users the genres of the movie result list so let's drop the duplicates.We also need to force reindex. The code for forced reindexing is provided below.

	title	genres
0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
215	Grumpier Old Men (1995)	Comedy|Romance
267	Heat (1995)	Action|Crime|Thriller
3369	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4572	Usual Suspects, The (1995)	Crime|Mystery|Thriller
5776	From Dusk Till Dawn (1995)	Action|Comedy|Horror|Thriller
6831	Bottle Rocket (1995)	Adventure|Comedy|Crime|Romance
7854	Braveheart (1995)	Action|Drama|War
...	...	...

Differences:

Process:

```
movie_genres = pandas.DataFrame.copy(movie_data2)
movie_genres = movie_genres.reset_index()
# fix the column names
movie_genres.columns = ['id','title','genres']
# Make the movie title the index
movie_genres = movie_genres.set_index('title')
movie_genres = movie_genres.drop(['id'], axis=1)
print(movie_genres.head(10))
```

Goal 3A, B, & C

Description: Calculate an average rating for each title, Count how many ratings each movie has (a higher rating count indicates stronger confidence in the rating), Create a table showing title, average rating, and rating count

title	rating	rating_counts
September 11th 2001 (2002)	4.0	2
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2
'Tis the Season for Love (2015)	1.5	1
'burbs, The (1989)	3.176	17
...	...	...

Differences:

Process:

Goal 4

Description: Create a pivot table with a row for each user, a column for each title, and values wherever a given user rated a given movie (there will be a lot of null values - this is OK)

title	September 11th, 2001 (2002)	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	...
userId	NaN	NaN	NaN	...
1	NaN	NaN	NaN	...
2	NaN	NaN	NaN	...
3	NaN	NaN	NaN	...
4	NaN	NaN	NaN	...
...	...	...	...	...

Differences:

Process:

Goal 5

From here, on, we enter the user interface stage and shift primary focus from data manipulation to user experience. Enter those answers at the end.

Description: Create a correlation matrix for all movies in the dataset with 5 or more ratings. The code to do this in Python is simple:

```
matrix_corr = user_movie_rating.corr(method='pearson', min_periods=5)
```

The result should look somthing like this (many Null values are likely but the diagonal where same movie row and column meet should always be 1):

	September 11th, 2001 (2002)	'71 (2014)	'Hellboy': The Seeds of Creation (2004)
September 11th, 2001 (2002)	1.00	NaN	NaN
'71 (2014)	NaN	1.00	NaN
'Hellboy': The Seeds of Creation (2004)	NaN	NaN	1.00
...	...	...	...

Prompt the User for Input

This can be done in several ways.For command line use, a simple input asking for a title and storing it in the MyMovie variable can be used. Note: users must be careful to input the title they want exactly has it appears in the movie csv le, including all formatting and dates associated. Alternatively, a databricks dropdown utility can be used to oer a subset of titles (around 1000) to the user to choose from, avoiding the potential data mismatch error from incorrect text entry. If you choose to use databricks, the code for a dropdown is as follows:

```
movies = movie_names.title.unique()
movies = random.sample(list(movies), 1000)
movie = movies[0]
print("Select a movie title from the dropdown above.")
dbutils.widgets.dropdown("movies", movie, [str(x) for x in movies], "Select a movie title")
MyMovie = dbutils.widgets.get("movies")
```

Goal 6

Description: Query the correlation matrix for the user's chosen movie as follows:

```
MyMovie_corr = matrix_corr[MyMovie]
MyMovie_corr.dropna(inplace=True)
```

Goal 7

Description: Reindex

```
MyMovie_corr = MyMovie_corr.reset_index()
MyMovie_corr.columns = ['title','correlation']
MyMovie_corr = MyMovie_corr.set_index('title')
```

Goal 8A & B

Description: Using joins, combine the MyMovie_corr table with rst the average rating and rating_counts table and then the movie_genres table. *Remember, the contents of your table should be similar in content but the particular titles and values will depend on what movie you input in the query!

title	correlation	rating	rating_counts	genres
(500) Days of Summer (2009)	-0.944911	3.6667	42	Comedy|Drama|Romance
10 Things I Hate About You (1999)	-0.960769	3.52778	54	Comedy|Romance
101 Dalmatians (One Hundred and One Dalmatians)...	0.277350	3.4318	44	Adventure|Animation|Childre
12 Years a Slave (2013)	1.0000	3.625	16	Drama
13 Going on 30 (2004)	-0.866025	3.190476	21	Comedy|Fantasy|Romance
15 Minutes (2001)	-1.0000	3.000	10	Thriller
16 Blocks (2006)	1.0000	3.000	9	Crime|Thriller
2 Fast 2 Furious (Fast and the Furious 2, The) ...	.960769	2.605263	19	Action|Crime|Thriller
2001: A Space Odyssey (1968)	-0.174078	3.894495	109	Adventure|Drama|Sci-Fi
...	...	...	...	...

Results

Description: Sort the resulting table by correlation, and within that, by rating, then rating_counts. Print at least 10 rows for the user.

title	correlation	rating	rating_counts	genres
Stand by Me(1986)	1.0	4.005	91	Adventure|Drama
American President, The (1995)	1.0	3.671	70	Comedy|Drama|Romance
Citizen Kane (1941)	1.0	4.04	69	Drama|Mystery
Legends of the Fall (1994)	1.0	3.397	68	Drama|Romance|War|Western
District 5 (2009)	1.0	3.77	65	Mystery|Sci-Fi|Thriller
Vertigo (1958)	1.0	4.025	60	Drama|Mystery|Romance|Thriller
...	...	...	...	...

Final: User Evaluation – Test the system by entering a few movies you know well. Do the suggestions given match your expectations? Why or why not? Give at least one suggestion based on the readings about other types of recommender systems that could be used to improve this system's performance.