DSC630

Xin Tang

Week 9 Movie recommender

---

create a recommender system that allows users to input a movie they like (in the data set) and recommends ten other movies for them to watch

I followed the method used in article (Nair, 2019) 'How To Build Your First Recommender System Using Python & MovieLens Dataset' but added my own additions

In [22]:
```
#load package and import dataset
import numpy as np
import pandas as pd

#Suspend the warning
import warnings
warnings.filterwarnings('ignore')

# I used small version of movieLens data
# import rating file
rating = pd.read_csv('ratings.csv')
rating.head(3)
```

Out[22]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224

In [24]:
```
# import movie file
movie_info = pd.read_csv("movies.csv")
movie_info.head(3)
```

Out[24]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance

In [25]:
```
#merge the 2 files together
movie_merged = rating.merge(movie_info,on='movieId', how='left')
movie_merged.head(2)
```

Out[25]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance

In [26]:
```
# Find average rating for each movie
rating_stat =pd.DataFrame(movie_merged.groupby('title')['rating'].mean())

# Find rating count per movie
rating_stat['rating_count'] = pd.DataFrame(movie_merged.groupby('title')['rating'].count())

rating_stat.head(3)
```

Out[26]:

	rating	rating_count
title		
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2

I am thinking of recommend the movie based on correlation and Genres

To avoid 'rich-get-richer' effect, I will recommend one movie with highest correlation from same genres, and another top correlated movie with different genres, I will also add one movie with latest release date as the third choice.

In [27]:
```
#Build a user to movie title correlation table
movie_user = movie_merged.pivot_table(index='userId',columns='title',values='rating')
```

```
movie_user.head(6)
```

Out[27]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]^2 (2009)	[REC]^3 3 Génesis (2012)	an
userId																
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	

6 rows × 9719 columns

In [52]:
```
# merge in the movie genres, show the list
#recc = recc.merge(movie_info,on='title', how='left')
#recc.head(10)
```

In [28]:
```
# build recommendation list based on user input of movie ID
def movie_recc_list(name):
    #build correlation list based on the movie name given
    correlations = movie_user.corrwith(movie_user[name])

    rec_list = pd.DataFrame(correlations,columns=['Correlation'])

    #drop NA
    rec_list.dropna(inplace=True)
    # add in count of rating
    rec_list = rec_list.join(rating_stat['rating_count'])

    #pull list of the correlated movies which being rated >100 times, sort it
    recc = rec_list[rec_list['rating_count']>100].sort_values('Correlation',ascending=False).reset_index()

    #now add in movie id and genres to form a complete list
    recc = recc.merge(movie_info,on='title', how='left')
    #print(recc.head(5))
    #return list for further process
    return(recc)
```

In [29]:
```
# Ideally need to ask user to input the choice, since I am in PN, so I just test it, using movieid
# pick movie with id= 4
M_name = movie_info[movie_info['movieId'] == 4].title

print(M_name)
```

```
3    Waiting to Exhale (1995)
Name: title, dtype: object
```

In [30]:
```
# create the recommendation list
movie_recc_list(M_name)
```

Out[30]:

Correlation	rating_count	movieId	title	genres

In [31]:
```
# validate the function runs okay
recc.head(4)
```

Out[31]:

	title	Correlation	rating_count	movieId	genres	year
0	Goodfellas (1990)	1.0	126	1213	Crime|Drama	1990
1	E.T. the Extra-Terrestrial (1982)	1.0	122	1097	Children|Drama|Sci-Fi	1982
2	Alien (1979)	1.0	146	1214	Horror|Sci-Fi	1979
3	Aliens (1986)	1.0	126	1200	Action|Adventure|Horror|Sci-Fi	1986

In [33]:
```
# normally the first one is user picked movie. so pick the top one with the same genres.
same = recc[recc['genres']== recc.iloc[0].genres]

first_name = same.iloc[1].title
```

In [78]:
```
# now find one with different genres
rest = recc[recc['genres']!= recc.iloc[0].genres]
```

```
second_name = rest.iloc[0].title
```

In [35]:
```
#third one is the one with latest release year
# start to extract year info from the returned list
import re
# define a function to get number out
def find_number(text):
    num = re.search(r'\d{4}',text)
    return num[0]

#create year column
recc['year']=recc['title'].apply(lambda x: find_number(x))
```

In [48]:
```
recc.head(5)
```

Out[48]:

	title	Correlation	rating_count	movieId	genres	year
0	Goodfellas (1990)	1.0	126	1213	Crime|Drama	1990
1	E.T. the Extra-Terrestrial (1982)	1.0	122	1097	Children|Drama|Sci-Fi	1982
2	Alien (1979)	1.0	146	1214	Horror|Sci-Fi	1979
3	Aliens (1986)	1.0	126	1200	Action|Adventure|Horror|Sci-Fi	1986
4	Shining, The (1980)	1.0	109	1258	Horror	1980

In [39]:
```
# the year have extract digits inside, so extract the last 4 digites.
recc['year'] = recc['year'][-4:]
```

In [40]:
```
# drop NA from year
recc.dropna()
```

Out[40]:

	title	Correlation	rating_count	movieId	genres	year
55	Home Alone (1990)	-1.0	116	586	Children|Comedy	1990
56	Groundhog Day (1993)	-1.0	143	1265	Comedy|Fantasy|Romance	1993
57	Back to the Future (1985)	-1.0	171	1270	Adventure|Comedy|Sci-Fi	1985
58	Indiana Jones and the Last Crusade (1989)	-1.0	140	1291	Action|Adventure	1989

In [55]:
```
# sort to get latest year to the top
recc_sorted = recc.sort_values(by=['year'], ascending=False)
recc_sorted.head()
```

Out[55]:

	title	Correlation	rating_count	movieId	genres	year
56	Groundhog Day (1993)	-1.0	143	1265	Comedy|Fantasy|Romance	1993
55	Home Alone (1990)	-1.0	116	586	Children|Comedy	1990
58	Indiana Jones and the Last Crusade (1989)	-1.0	140	1291	Action|Adventure	1989
57	Back to the Future (1985)	-1.0	171	1270	Adventure|Comedy|Sci-Fi	1985
0	Goodfellas (1990)	1.0	126	1213	Crime|Drama	NaN

In [73]:
```
# now the third choice is the one with latest year, which is at the first row
third_name = recc_sorted.iloc[0].title
```

Out[73]:
```
'Groundhog Day (1993)'
```

In [79]:
```
print('Based on your input of movie\n')
print('my recommendation of 3 movies are:', first_name, 'and', second_name, 'and', third_name)
```

```
Based on your input of movie

my recommendation of 3 movies are: Shawshank Redemption, The (1994) and E.T. the Extra-Terrestrial (1982) and Groundhog
Day (1993)
```

In [ ]:
```
Reference:

(Nair, 2019) 'How To Build Your First Recommender System Using Python & MovieLens Dataset

data resource: dataset is provided by Bellevue University for using in DSC630 course.
```