# # -\*- coding: utf-8 -\*-

# """popularityBasedRecommendation\_Movies.py

# Automatically generated by Colaboratory.

# Original file is located at

# https://colab.research.google.com/drive/1I1baWoS1W0zyrXOxHf9UP39BkA26MNEB

# """

# import numpy as np

# import pandas as pd

# import matplotlib.pyplot as plt

# from google.colab import drive

# drive.mount('/content/drive')

# movies = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/movies.csv')

# ratings = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/ratings.csv')

# userPreferences=['Adventure','Romance','Fantasy']

# def popularityBasedRecommendation(userPreferences):

# movies = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/movies.csv')

# ratings = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/ratings.csv')

# movieRatings=pd.merge(movies,ratings,on="movieId")

# movieRatings= movieRatings.drop(["genres"],axis=1)

# movie\_ratingCount = (movieRatings.groupby(by=['movieId'])['rating'].count().reset\_index().rename(columns = { 'rating':'totalRatingCount'})[['movieId','totalRatingCount']])

# rating\_with\_totalRatingCount = movieRatings.merge(movie\_ratingCount, on = 'movieId')

# popularity\_thresold = 50

# rating\_popular\_movie = rating\_with\_totalRatingCount.query('totalRatingCount >= @popularity\_thresold')

# movie\_ratingAvg = (movieRatings.groupby(by=['movieId'])['rating'].mean().reset\_index().rename(columns = { 'rating':'totalRatingAvg'})[['movieId','totalRatingAvg']])

# movie\_ratingAvg = (movieRatings.groupby(by=['movieId'])['rating'].mean().reset\_index().rename(columns = { 'rating':'totalRatingAvg'})[['movieId','totalRatingAvg']])

# movie\_ratingAvg=pd.merge(movies,movie\_ratingAvg,on='movieId')

# movie\_ratingAvg['z\_score'] = (movie\_ratingAvg['totalRatingAvg'] - movie\_ratingAvg['totalRatingAvg'].mean())/movie\_ratingAvg['totalRatingAvg'].std(ddof=0)

# movie\_ratingAvg['score']=movie\_ratingAvg['totalRatingAvg']\*movie\_ratingAvg['z\_score']

# # popularityBasedFilteredTable.sort\_values(by="score",ascending=False)

# popularityBasedFilteredTable = movie\_ratingAvg.sort\_values(by=["score"],ascending=False)

# k=20

# c=0

# i=0

# popularityBasedRecommendedMovies=[]

# userPreferencesSet=set(userPreferences)

# while(1):

# genresList=popularityBasedFilteredTable.iloc[i]['genres'].split('|')

# genresSet=set(genresList)

# if genresSet.intersection(userPreferencesSet):

# popularityBasedRecommendedMovies.append(popularityBasedFilteredTable.iloc[i]['title'])

# c+=1

# i+=1

# if c>k:

# break

# return popularityBasedRecommendedMovies

# movies.head()

# ratings.head()

# movieRatings=pd.merge(movies,ratings,on="movieId")

# movieRatings.head(10)

# # movieRatings= movieRatings.drop(["genres"],axis=1)

# # movieRatings

# movieRatings= movieRatings.dropna()

# movieRatings

# movie\_ratingCount = (movieRatings.groupby(by=['movieId'])['rating'].count().reset\_index().rename(columns = { 'rating':'totalRatingCount'})[['movieId','totalRatingCount']])

# movie\_ratingCount

# rating\_with\_totalRatingCount = movieRatings.merge(movie\_ratingCount, on = 'movieId')

# rating\_with\_totalRatingCount

# #giving some popularity thresold to discard less popular books

# popularity\_thresold = 50

# rating\_popular\_movie = rating\_with\_totalRatingCount.query('totalRatingCount >= @popularity\_thresold')

# rating\_popular\_movie

# # #combining rating dataset with book dataset on ISBN field

# # combine\_book\_rating = pd.merge(ratings, books, on='ISBN')

# # combine\_book\_rating = combine\_book\_rating.drop(['Book-Author','Year-Of-Publication','Publisher','Image-URL-S','Image-URL-M','Image-URL-L'],axis=1)

# # # print(combine\_book\_rating.shape)

# # combine\_book\_rating.head()

# # combine\_book\_rating = combine\_book\_rating.dropna(axis=0, subset = ['Book-Title'])

# # book\_ratingCount = (combine\_book\_rating.groupby(by=['Book-Title'])['Book-Rating'].count().reset\_index().rename(columns = { 'Book-Rating':'totalRatingCount'})[['Book-Title','totalRatingCount']])

# # book\_ratingCount.head()

# # rating\_with\_totalRatingCount = combine\_book\_rating.merge(book\_ratingCount, on = 'Book-Title')

# # rating\_with\_totalRatingCount.head()

# # #giving some popularity thresold to discard less popular books

# # popularity\_thresold = 50

# # rating\_popular\_book = rating\_with\_totalRatingCount.query('totalRatingCount >= @popularity\_thresold')

# # rating\_popular\_book = rating\_popular\_book.sort\_values(by="totalRatingCount",ascending=False)

# # rating\_popular\_book.head()

# # book\_ratingAvg = (combine\_book\_rating.groupby(by=['Book-Title'])['Book-Rating'].mean().reset\_index().rename(columns = { 'Book-Rating':'totalRatingAvg'})[['Book-Title','totalRatingAvg']])

# # # book\_ratingAvg.index='Book-Title'

# # book\_ratingAvg.head()

# # book\_ratingAvg.describe()

# movieRatings

# movie\_ratingAvg = (movieRatings.groupby(by=['movieId'])['rating'].mean().reset\_index().rename(columns = { 'rating':'totalRatingAvg'})[['movieId','totalRatingAvg']])

# movie\_ratingAvg

# movie\_ratingSum = (movieRatings.groupby(by=['movieId'])['rating'].sum().reset\_index().rename(columns = { 'rating':'totalRatingSum'})[['movieId','totalRatingSum']])

# movie\_ratingSum

# movie\_ratingAvg=pd.merge(movieRatings,movie\_ratingAvg,on='movieId')

# movie\_ratingAvg

# movies

# # movie\_ratingAvg=pd.merge(movies,movie\_ratingAvg,on='movieId')

# # movie\_ratingAvg

# # zscore = pd.DataFrame([])

# # book\_ratingAvg=book\_ratingAvg.drop(['Z\_Score'],axis=1)

# movie\_ratingAvg['z\_score'] = (movie\_ratingAvg['totalRatingAvg'] - movie\_ratingAvg['totalRatingAvg'].mean())/movie\_ratingAvg['totalRatingAvg'].std(ddof=0)

# movie\_ratingAvg.head(10)

# # movie\_ratingAvg['score']=movie\_ratingAvg['totalRatingAvg']\*movie\_ratingAvg['z\_score']

# popularityBasedFilteredTable = movie\_ratingAvg.sort\_values(by=["totalRatingAvg","z\_score"],ascending=(False,False))

# popularityBasedFilteredTable.head(10)

# movie\_ratingAvg['score']=movie\_ratingAvg['timestamp']\*movie\_ratingAvg['z\_score']

# movie\_ratingAvg

# # popularityBasedFilteredTable.sort\_values(by="score",ascending=False).head(10)

# movie\_ratingAvg.sort\_values(by="z\_score",ascending=False).head(10)

# popularityBasedFilteredTable = movie\_ratingAvg.sort\_values(by="score",ascending=False)

# popularityBasedFilteredTable

# movie\_ratingAvg.sort\_values(by="totalRatingAvg",ascending=False).head(10)

# l=popularityBasedFilteredTable.iloc[0]['genres'].split('|')

# print(l)

# if l in userPreferences:

# print("h")

# userPreferencesSet=set(userPreferences)

# userPreferencesSet

# k=20

# c=0

# i=0

# popularityBasedRecommendedMovies=[]

# userPreferencesSet=set(userPreferences)

# while(1):

# genresList=popularityBasedFilteredTable.iloc[i]['genres'].split('|')

# genresSet=set(genresList)

# if genresSet.intersection(userPreferencesSet):

# popularityBasedRecommendedMovies.append(popularityBasedFilteredTable.iloc[i]['title'])

# c+=1

# i+=1

# if c>k:

# break

# # print(c)

# popularityBasedRecommendedMovies

def popularityBasedRecommendation(userPreferences,coldUserRatedMovieIds):

movies = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/movies.csv')

ratings = pd.read\_csv('/content/drive/My Drive/FinalYearProj-2020/DataSets/Movielens/ratings.csv')

movieRatings=pd.merge(movies,ratings,on="movieId")

movieRatings= movieRatings.drop(["genres"],axis=1)

movie\_ratingCount = (movieRatings.groupby(by=['movieId'])['rating'].count().reset\_index().rename(columns = { 'rating':'totalRatingCount'})[['movieId','totalRatingCount']])

rating\_with\_totalRatingCount = movieRatings.merge(movie\_ratingCount, on = 'movieId')

popularity\_thresold = 50

rating\_popular\_movie = rating\_with\_totalRatingCount.query('totalRatingCount >= @popularity\_thresold')

movie\_ratingAvg = (movieRatings.groupby(by=['movieId'])['rating'].mean().reset\_index().rename(columns = { 'rating':'totalRatingAvg'})[['movieId','totalRatingAvg']])

movie\_ratingAvg = (movieRatings.groupby(by=['movieId'])['rating'].mean().reset\_index().rename(columns = { 'rating':'totalRatingAvg'})[['movieId','totalRatingAvg']])

movie\_ratingAvg=pd.merge(movies,movie\_ratingAvg,on='movieId')

movie\_ratingAvg['z\_score'] = (movie\_ratingAvg['totalRatingAvg'] - movie\_ratingAvg['totalRatingAvg'].mean())/movie\_ratingAvg['totalRatingAvg'].std(ddof=0)

movie\_ratingAvg['score']=(movie\_ratingAvg['timestamp']\*0.00000001)\*(movie\_ratingAvg['z\_score']\*0.99999999)

# popularityBasedFilteredTable.sort\_values(by="score",ascending=False)

popularityBasedFilteredTable = movie\_ratingAvg.sort\_values(by=["score"],ascending=False)

k=20

c=0

i=0

popularityBasedRecommendedMovies=[]

userPreferencesSet=set(userPreferences)

while(1):

genresList=popularityBasedFilteredTable.iloc[i]['genres'].split('|')

genresSet=set(genresList)

if genresSet.intersection(userPreferencesSet):

if popularityBasedFilteredTable.iloc[i]['movieId'] not in coldUserRatedMovieIds:

popularityBasedRecommendedMovies.append(popularityBasedFilteredTable.iloc[i]['title'])

c+=1

i+=1

if c>k:

break

return popularityBasedRecommendedMovies