Project 3: Movies Recommendations

Due Date: *Monday, November 17 (11.59PM)*

Submission: on Gradescope

Instructions:

The projects may be completed in groups (up to 4 students). If you do so, please make sure that you include everyone's full name, and that you *also select everyone's name when submitting the assignment on Gradescope*. This ensures that each group member will get a grade assigned and have access to the comments from the graders.

For this project, you only need to submit one file, e.g. your R-Markdown or Jupyter Notebook, **that includes both the code and the output** for the questions below.

Grading Rubric: *Part A*= 30 points, *Part B*= 70 points.

Project Description:

The goal of this project is to make **movies recommendations**. Specifically, we are going to work with the **1M MovieLens** data set found here. The data set contains about $1,000,209$ ratings from $6,040$ users for $3,900$ movies on an $1-5$ stars scale.

After loading and pre-processing the data as needed, consider the following tasks.

(A) Recommend the **top 10 "*most popular*" movies**. The term "most popular" is open to interpretation from each group. So, it is important to clearly state how you define the popularity of a movie. The recommendation you make should display the top 10 movie titles, and information related to the criteria you used - e.g. if you recommend the top 10 movies based on ratings for movies with at least 1,000 ratings, report these numbers.

(B) Make a recommendation based on **Item-Based Collaborative Filtering** (IBCF) by writing your own **IBCF** function following the steps below:

1. Create the rating matrix by R, where the rows correspond to users and the columns to movies. Normalize R by centering each row, i.e. subtract the row means from each row of the matrix. The matrix is *sparse*, so the row means should be computed based on the non-NA entries.

2. Compute the **cosine similarity** for all the movies. You can ignore similarities computed based on less than 3 user ratings. You can also use the $(1 + cos)/2$ transformation to ensure than the similarity will be between 0 and 1. Keep in mind that you may have NAs when a pair off movies (for example) has been rated by 0, 1, or 2 users, or when the denominator in the similarity metric is zero.

3. For the similarity matrix in (2), sort the non-NA similarity measures and keep the *top 30*, setting the rest to NA. Display the pairwise similarity values for this new matrix for the following movies:
   - "Toy Story (1995)"
   - "GoldenEye (1995)"
   - "Liar Liar (1997)"
   - "Lost World: Jurassic Park (1997)"
   - "Sixth Sense, The (1999)"

4. Write your own *IBCF function*:
   - *Input*: `newuser`, i.e. a vector containing ratings for all the movies.
     Many of the entries will be NA, while the non-NA values should be ratings 1, 2, 3, 4, 5 - the star ratings.

   - *Output*: The top 10 movies recommendation for this new user.
     If fewer than 10 predictions are non-NA, select the remaining movies based on the popularity defined in part (A), prioritizing the most popular ones and excluding those already rated by the user. Here, you may want to save the ranking of all movies (based on popularity) in a separate file to avoid recomputing the ranking each time.

   - The function you write should load the similarity matrix and use it to compute predictions for movies that have not been rated by the new user yet, using the formula from the notes/slides.

5. Test your function by displaying the top 10 recommendations for the following two users:
   - User in row 1500 from the rating matrix.
   - A hypothetical user who rates movie "Star Wars: Episode IV - A New Hope (1977)" with *5 stars* and movie "Independence Day (ID4) (1996)" with *4 stars*.