Zee Recommender Systems

Create a Recommender System to show personalized movie recommendations based on ratings given by a user and other users similar to them in order to improve user experience.

Dataset: **https://drive.google.com/drive/folders/1RY4RG7rVfY8-0uGeOPWqWzNIuf-iosuv**

Data Dictionary:

1. **RATINGS FILE DESCRIPTION**

All ratings are contained in the file "ratings.dat" and are in the following format:

`UserID::MovieID::Rating::Timestamp`

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds
- Each user has at least 20 ratings

2. USERS FILE DESCRIPTION

User information is in the file "users.dat" and is in the following format:

`UserID::Gender::Age::Occupation::Zip-code`

All demographic information is provided voluntarily by the users and is not checked for accuracy.

Only users who have provided some demographic information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:
    - 1: "Under 18"
    - 18: "18-24"
    - 25: "25-34"
    - 35: "35-44"
    - 45: "45-49"
    - 50: "50-55"
    - 56: "56+"
- Occupation is chosen from the following choices:
    - 0: "other" or not specified
    - 1: "academic/educator"
    - 2: "artist"
    - 3: "clerical/admin"
    - 4: "college/grad student"
    - 5: "customer service"
    - 6: "doctor/health care"
    - 7: "executive/managerial"
    - 8: "farmer"
    - 9: "homemaker"
    - 10: "K-12 student"
    - 11: "lawyer"

- 12: "programmer"
- 13: "retired"
- 14: "sales/marketing"
- 15: "scientist"
- 16: "self-employed"
- 17: "technician/engineer"
- 18: "tradesman/craftsman"
- 19: "unemployed"
- 20: "writer"

3. MOVIES FILE DESCRIPTION

Movie information is in the file "movies.dat" and is in the following format:

`MovieID::Title::Genres`

- Titles are identical to titles provided by the IMDB (including year of release)
- Genres are pipe-separated and are selected from the following genres:
  - Action
  - Adventure
  - Animation
  - Children's
  - Comedy
  - Crime
  - Documentary
  - Drama
  - Fantasy
  - Film-Noir
  - Horror

    - Musical
    - Mystery
    - Romance
    - Sci-Fi
    - Thriller
    - War
    - Western

Concepts Tested:

- Recommender Engine
- Collaborative Filtering (Item-based & User-based Approach)
- Pearson Correlation
- Nearest Neighbors using Cosine Similarity
- Matrix Factorization

What does "good" look like?

- Reading the data files, formatting them into a proper workable format and merging the data files into one single dataframe

Eg: pd.read_fwf('../input/zeemovie/movies.dat', encoding='ISO-8859-1'

- Performing exploratory data analysis like checking the structure & characteristics of the dataset and cleaning the data
- Performing feature engineering steps type conversions and deriving new features like 'Release Year'
- Visualizing the data with respect to different categories to get a better understanding of the underlying distribution
- Grouping the data in terms of Average Rating and No. of Ratings given
- Creating a pivot table of movie titles & user id and imputing the NaN values with a suitable value
- Follow the Item-based approach and
    - **Pearson Correlation**
        - Take a movie name as input from the user
        - Recommend 5 similar movies based on Pearson Correlation
    - **Cosine Similarity**
        - Print the item similarity matrix and user similarity matrix
            - Example: An user-user similarity matrix just for demonstration.

	U1	U2	U3	U4
U1	1	0.16	0.82	0.27
U2	0.16	1	0.57	0.85
U3	0.82	0.57	1	0.52
U4	0.27	0.85	0.52	1

-
    - Create a CSR matrix using the pivot table.[**Optional**, This is an extended approach, **link** to example implementation].
    - Write a function to return top 5 recommendations for a given item

    - **[sklearn optional]** Take a movie name as user input and use KNN algorithm to recommend 5 similar movies based on Cosine Similarity. [**link** to sklearn Nearest Neighbor documentation]

- **Matrix Factorization**
    - Use **cmfrec/Surprise** library to run matrix factorization. (Show results with d=4).
    - Evaluate the model's performance using RMSE and MAPE.
    - **Bonus** - how can you do a train test split for MF?
- **Embeddings for item-item and user-user similarity**
    - Re-design the item-item similarity function to use MF embeddings (d=4) instead of raw features
    - Similarly, do this for user-user similarity
    - **Bonus**: Get d=2 embeddings, and plot the results. Write down your analysis from this visualisation. (Compare with other visualization techniques)
- **Follow the User-based approach (Optional)**
    - Ask the user to rate a few movies and create a dataframe of the user's choices.
    - Find other users who've watched the same movies as the new user.
    - Sort the old users by the count of most movies in common with the new user.
    - Take the top 100 users and calculate a Similarity Score for each user using the Pearson Correlation function.
    - Get the top 10 users with the highest similarity indices, all the movies for these users, and add Weighted movie Ratings by Multiplying the Rating to the Similarity Index.
    - Calculate the average recommendation score by dividing the Weighted Rating by the Similarity Index and select movies with the highest score i.e., 5.
    - Now, recommend 10 movies based on the ratings given by old users who are similar to the new user.

Evaluation Criteria (100 points)

1. **Define Problem Statement and Formatting the Data (20 points)**
    1. Definition of the problem (as per the given problem statement with additional views)
    2. Formatting the data files to bring them into a workable format
    3. Merging the data files and creating a single consolidated dataframe
2. **Performing EDA, Data Cleaning, and Feature Engineering (20 Points)**
    1. Reviewing the shape and structure of the dataset
    2. Performing necessary type conversion and deriving new features
    3. Investigating the data for any inconsistency
    4. Group the data according to the average rating and no. of ratings
3. **Build a Recommender System based on Pearson Correlation (10 Points)**
    1. Creating a pivot table of movie titles & user id and imputing the NaN values
    2. Use the Item-based approach to create a simple recommender system that uses Pearson Correlation
4. **Build a Recommender System based on Cosine Similarity. (20 Points)**
    1. Print the user similarity matrix and item similarity matrix
    2. Use the Item-based approach to create a recommender system that uses Nearest Neighbors algorithm and Cosine Similarity
5. **Build a Recommender System based on Matrix Factorization. (30 Points)**
    1. Create a Recommender System using the Matrix Factorization method
    2. Evaluate the model in terms of the Root Mean Squared Error and Mean Absolute Percentage Error

3. Use embeddings for visualization and similarity-based models.

6. **Build a Recommender System based Pearson Correlation. (Optional)**

    1. Use the User-based approach to create a recommender system that uses Pearson Correlation

Questionnaire:

1. Users of which age group have watched and rated the most number of movies?
2. Users belonging to which profession have watched and rated the most movies?
3. Most of the users in our dataset who've rated the movies are Male. (T/F)
4. Most of the movies present in our dataset were released in which decade?

    1. 70s b. 90s c. 50s d.80s

5. The movie with maximum no. of ratings is ___.
6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.
7. On the basis of approach, Collaborative Filtering methods can be classified into ___-based and ___-based.
8. Pearson Correlation ranges between ___ to ___ whereas, Cosine Similarity belongs to the interval between ___ to ___.
9. Mention the RMSE and MAPE that you got while evaluating the Matrix Factorization model.
10. Give the sparse 'row' matrix representation for the following dense matrix -

[[1 0]
[3 7]]