HW4: Movie Recommender System

Details:

Name: Aniket Pandey
Miner2 Username: 414
GMU Id: apandey7
Submission Details:

- Rank: 55
- Public Score: 0.91

Steps to run the code:

1. Download MovieRecommender.ipynb file
2. Open JupyterLab on Anaconda Navigator or Google Colab Notebook
3. On the menu bar click File -> Open -> Filename
4. Keep the MovieRecommender.py and data file in the same directory
5. One by one run each cell in the notebook

Introduction:

Recommender systems are among the most popular applications of data science today. They are used to predict the "rating" or "preference" that a user would give to an item. In this assignment we will develop a movie recommendation system which will predict the rating a movie will receive based how others rate the similar movie.

Solution:

We implemented two types of method for movie recommendation system:
1. Content Based - Suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person likes a particular item, he or she will also tend to like an item that is similar to it. And to recommend that, it will make use of the user past item metadata.

- Use tfidfVectorizer() is used to convert movie details to a sparse matrix of word representation
- The sparse matrix is then used to find the inverse document frequency, it reduces the priority of frequently occurring words
- Call the tfidfVectorizer() and then find the cosine similarity
- Iterate the test dataframe and for each row do the following:
  -> Get the index of the movie that matches the movieId
  -> Get the pairwise similarity scores of all movies similar to that movie
  -> Sort the movies based on the similarity scores
  -> Get the scores of the 'K' most similar movies
  -> Get the movie indexes
  -> Write the prediction in output file

2. Correlative filtering - These systems are widely used, and they try to predict the rating or preference that a user would give an item-based on past ratings and preferences of other users. Collaborative filters do not require item metadata like its content-based counterparts.

- Normalize the movie ratings by calling the preprocess() function
- Get the cosine similarity and pearson correlation similarity
- Iterate the test dataframe and for each row do the following:
  -> Get nearest best neighbors
  -> Get the user indexes
  -> Get the scores of the 'K' most similar users
  -> Write the prediction in output file

Process/Approach:

1. Import necessary packages and libraries.
   - Numpy for working mathematical operations on arrays
   - Pandas for Dataframe
   - sklearn.metrics.pairwise for cosine similarity
   - sklearn.model_selection for K Fold cross validation
   - sklearn.feature_extraction.text for TF-IDF Vectorizer
2. Read Data files using Pandas: Additional Files[Actors, Directors, Genres, Tags] + Train + Test
3. Arrange the tables read above by merging the columns based on 'movieID'
4. Form a final movies table by merging Actors + Directors + Genre + Tags based on MovieId as arranged above.
5. Clean the data read above by calling the cleanData() function. It will replace spaces with empty value and convert all strings to lowercase.
6. Perform K fold cross validation to find the best K value once for content based and then for correlative filtering. We split dataset into 10 consecutive folds each fold was then used once as a validation while the k-1 remaining folds form the training set and then RMSE for each recommender method was calculated. Following result were observed after performing cross validation:

	Content Based	Correlative Filtering
K = 50	1.167	0.983
K = 500	1.017	0.911
K = 1500	0.994	0.915

Table 1: RMSE after Cross Validation

7. From the above table we get that the RMSE for K=500 Correlative Filtering method is the least and therefore we will perform the actual recommendation with K=500 Correlative Filtering.

Conclusion:

Correlative filtering requires rating from other users to find similarities between users and then provide recommendations. In contrast, the content based approach only needs to analyze the

item and individual user personal data to make recommendations. Correlative filtering provides recommendations based on other unknown users who have the same taste as a given user, while content-based filtering items are recommended on a feature basis. For the given dataset correlative filtering result had less root mean square error and therefore it was used for actual prediction of ratings.