Beda.DSC630.Week10

April 3, 2025

[2]:
```
import pandas as pd

# Load each dataset
links = pd.read_csv("https://raw.githubusercontent.com/cheribeda/
 ↪Predictive-Analytics/main/links.csv")
movies = pd.read_csv("https://raw.githubusercontent.com/cheribeda/
 ↪Predictive-Analytics/main/movies.csv")
ratings = pd.read_csv("https://raw.githubusercontent.com/cheribeda/
 ↪Predictive-Analytics/main/ratings.csv")

# Display the first few rows of each dataset
print("Links Dataset:")
display(links.head())
print("\nMovies Dataset:")
display(movies.head())
print("\nRatings Dataset:")
display(ratings.head())
```

```
Links Dataset:
```

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

```
Movies Dataset:
```

	movieId	title	\
0	1	Toy Story (1995)	
1	2	Jumanji (1995)	
2	3	Grumpier Old Men (1995)	
3	4	Waiting to Exhale (1995)	
4	5	Father of the Bride Part II (1995)	

	genres
0	Adventure|Animation|Children|Comedy|Fantasy

```
1                    Adventure|Children|Fantasy
2                                Comedy|Romance
3                          Comedy|Drama|Romance
4                                        Comedy
```

Ratings Dataset:

```
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```

Step 1: Data Preparation

Merge Ratings and Movies Data: To create a user-movie matrix, merge the ratings and movies datasets on the movieId.

Pivot Data: Convert the merged data into a user-movie matrix, where each row represents a user and each column represents a movie, with the values being the user's rating for that movie.

[4]:
```
# Data Preparation
# Merge ratings and movies data to create a user-movie matrix.
# Merge the ratings and movies datasets on the movieId.

Pivot Data: Convert the merged data into a user-movie matrix, where each row␣
 ↪represents a user and each column represents a movie, with the values being␣
 ↪the user's rating for that movie.

# Merge ratings and movies data
data = pd.merge(ratings, movies, on="movieId")

# Create user-movie matrix
user_movie_matrix = data.pivot_table(index='userId', columns='title',␣
 ↪values='rating').fillna(0)
```

Step 2: Compute Cosine Similarity

To find movies similar to a given movie, calculate the cosine similarity between movie vectors in the user movie matrix. This will allow you to identify movies that received similar ratings from users.

[6]:
```
from sklearn.metrics.pairwise import cosine_similarity

# Compute cosine similarity between movies
movie_similarity = cosine_similarity(user_movie_matrix.T)
# Create a DataFrame for similarity scores for easier access
similarity_df = pd.DataFrame(movie_similarity, index=user_movie_matrix.columns,␣
 ↪columns=user_movie_matrix.columns)
```

Step 3: Define the Recommendation Function

This function: Takes a movie title as input. Sorts movies by similarity score in descending order. Returns the top ten similar movies excluding the input movie itself.

[12]:
```
def get_recommendations(movie_title, similarity_df, n=10):
    if movie_title not in similarity_df.columns:
        return "Movie not found in the dataset."
    # Get similarity scores and sort them
    similar_scores = similarity_df[movie_title].sort_values(ascending=False)
    # Exclude the input movie from recommendations
    recommended_movies = similar_scores.index[1:n+1]
    return recommended_movies

# Example
movie_title = "Amityville Horror, The (1979)"
recommended_movies = get_recommendations(movie_title, similarity_df)
print("Recommended movies for", movie_title, ":", recommended_movies)
```

```
Recommended movies for Amityville Horror, The (1979) : Index(['Friday the 13th
Part IV: The Final Chapter (1984)',
       'Friday the 13th Part 3: 3D (1982)', 'Child's Play 3 (1991)',
       'Amityville Curse, The (1990)', 'Child's Play 2 (1990)',
       'Christine (1983)', 'Child's Play (1988)', 'Amityville 3-D (1983)',
       'Poltergeist II: The Other Side (1986)',
       'Amityville: A New Generation (1993)'],
      dtype='object', name='title')
```

Referenes:

Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TIIS), 5(4), 1-19.

McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (pp. 51-56)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. Available from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

[ ]: