OBSERVATIONS BEFORE HYBRID FILTERING:

- The recommender we built earlier (**Popularity based Filter**) suffers some severe limitations. For one, it gives the same recommendation to everyone, regardless of the user's personal taste. If a person who loves romantic movies (and hates action) were to look at our Top 10 Chart, s/he wouldn't probably like most of the movies.

  For instance, consider a person who loves *Dilwale Dulhania Le Jayenge*, *My Name is Khan* and *Kabhi Khushi Kabhi Gham*. One inference we can obtain is that the person loves the actor Shahrukh Khan and the director Karan Johar. Even if s/he were to access the romance chart, s/he wouldn't find these as the top recommendations.

- To personalise our recommendations more, we used movie metadata (or content) to build another recommender engine, known as **Content Based Filtering.** It was considerably better than a simple Popularity Based filter as it used most of the keywords used in a movie to determine whether it should be recommended or not.

  The content-based engine suffers from some severe limitations. It is only capable of suggesting movies which are *close* to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres. Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who s/he is.

- Therefore, we used a technique called **Collaborative Filtering** to make recommendations to Movie Watchers. Collaborative Filtering is based on the idea that users similar to me can be used to predict how much I would like a particular product or service those users have used/experienced but I have not.

  One startling feature of this recommender system is that it doesn't care what the movie is (or what it contains). It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie.

Considering all the above observations, I have tried to bring together all these techniques to get more personalized and tailored recommendations for particular users.

HYBRID FILTERING

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of pure recommendation systems.

The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm. Using multiple recommendation techniques can suppress the weaknesses of an individual technique in a combined model.

The combination of approaches can be done in any of the following ways:

a. separate implementation of algorithms and combining the result

b. utilizing some content-based filtering in collaborative approach

c. utilizing some collaborative filtering in content-based approach

d. creating a unified recommendation system that brings together both approaches.

TASK DETAILS

Based on the dataset, use a Hybrid approach to generate recommendations for the given dataset .

DATA SET Provided

1. **Final Data:-**

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   book_id  999 non-null    int64
 1   authors  999 non-null    object
 2   title    999 non-null    object
 3   Genres   999 non-null    object
```

2. **Average Ratings:-**

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   book_id  10000 non-null  int64
 1   rating   10000 non-null  float64
```

3. **Rating Counts:-**

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   book_id  10000 non-null  int64
 1   rating   10000 non-null  int64
```

APPROACH

I have applied content-based filtering here along with Popularity based filtering.

I have achieved this by following the below steps: -

1. Merged all the datasets into a single one so that the current data set could hold more information.
2. Concatenated all the text-based information like authors, genres into a single column.
3. Based on the new column, have computed cosine similarity, and created a cosine similarity matrix.
4. Defined a method “get_recommendation” to compute the similarity scores for each “book_id”. Also in this method, have created another data frame having a new column for “Similarity Score”
5. Sorted this new data frame based on Similarity score as well as Average Ratings for each book.

OBSERVATIONS

This recommendation code filters out top 10 books based on keywords and popularity.

However, upon test running for few sample books, have observed that, I personally would like maybe 6 or 7 out of those 10 recommendations. As a result, this approach needs more refinement.

CONCLUSION

This may not be a very optimal approach to recommend new books, since it is based on the book popularity and few keywords that define this book.

However, it is better than standalone – demographic based filtering, popularity-based filtering or content-based filtering.