Recommender Systems for Article recommendation

Serendipite is an article recommendation platform where articles from different domains such as technology, politics, news and so on are shared by its users and then these articles are recommended on the basis of reading habits.

The objective of this exercise is to build both a non-personalised popularity-based recommender system and explore further the possibility of bringing personalised article recommendations to its customer base. For this exercise, I have used 3 different techniques of collaborative filtering, namely- user-based, item-based and Matrix factorization-based methods.

Data Overview

The training set has a total of 16731 user IDs and article IDs and their ratings with 2529 unique articles. The article info dataset has more details on the articles such as the website, title and the content. Majority of the ratings are rated 1 and rest of the ratings are split between 2, 3 & 5 with 4 ratings being the lowest.

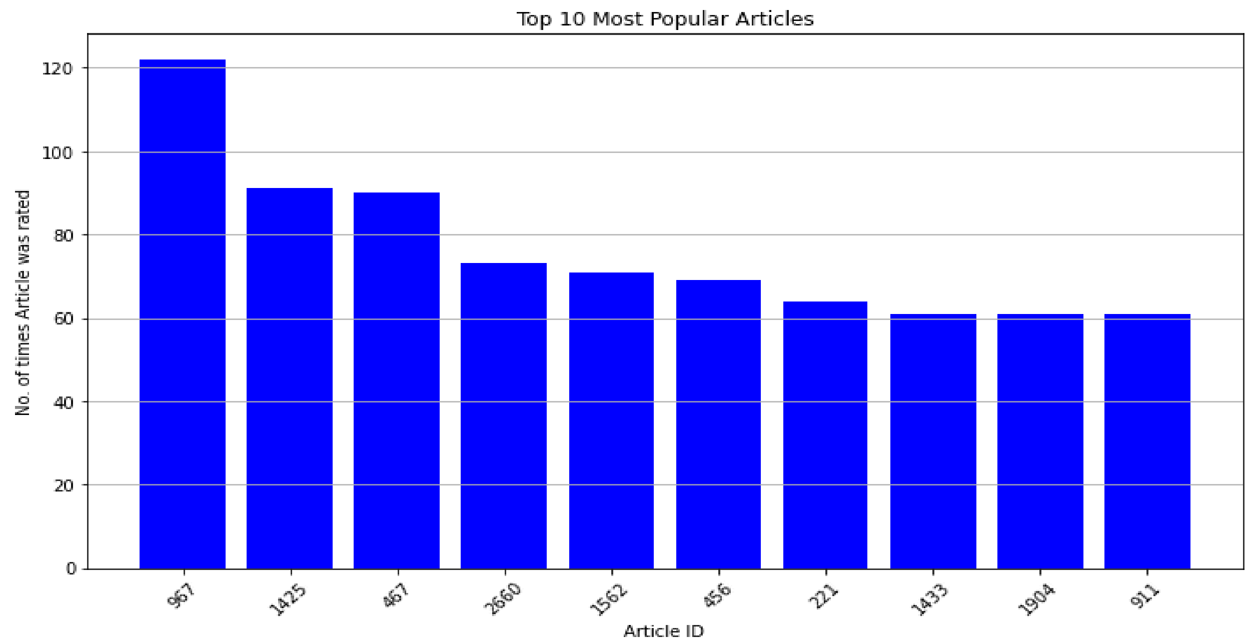
```
In [84]: df_ar['rating'].value_counts()
Out[84]:
rating
1    12822
2     1949
3      989
5      755
4      216
```

After combining the training set and the article info datasets into one file, I split the data into training and test sets using 25% for the test set. RMSE metric is used for evaluation.

Methodology

A popularity based approach looks at the top 10 most popular articles in terms of article counts.

```
Top 10 Most Popular Articles:
   article_id  article_count
0         967            122
1        1425             91
2         467             90
3        2660             73
4        1562             71
5         456             69
6         221             64
7        1433             61
8        1904             61
9         911             61
```



Top 10 articles based on average ratings are shown as below -

Top 10 articles based on average rating:

	article_id	avg_rating	num_ratings
1385	1628	5.0	1
1943	2282	5.0	1
401	479	5.0	1
692	821	5.0	1
1870	2190	5.0	1
1087	1273	5.0	1
1582	1856	5.0	1
2017	2366	5.0	1
1410	1655	5.0	1
2507	2949	5.0	1

Similarly, top 10 most read articles with average rating > 1.5 are shown as below -

10 most read articles with avg rating > 1.5:

	article_id	avg_rating	num_ratings
1220	1433	1.639344	61
2364	2781	1.700000	60
1067	1249	1.706897	58
488	580	1.833333	48
2012	2361	1.523810	42
1163	1366	1.731707	41
1858	2178	1.578947	38
1918	2248	1.514286	35
1460	1716	1.794118	34
1372	1614	1.515152	33

And finally, using a minimum rating threshold of 2, a weighted rating average was calculated using the following formula -

$$W = (R * v + C * m)/(v + m)$$

where :

R = average rating of the article
v = number of ratings for the article
m = minimum number of ratings for an article to be on recommendation list
C = mean ratings for all the articles

The top 10 articles based on weighted rating are as follows -

```
Top 10 articles based on weighted rating:
      article_id  avg_rating  num_ratings  weighted_rating
1944        2283    5.000000            2         3.246837
199          239    5.000000            2         3.246837
1826        2141    4.000000            4         3.164558
107          129    4.000000            4         3.164558
2363        2779    3.666667            6         3.123418
789          931    3.500000            8         3.098735
18            24    4.000000            3         2.997469
617          739    4.000000            3         2.997469
729          861    4.500000            2         2.996837
1774        2079    3.333333            6         2.873418
```

<u>User-Based Collaborative Filtering</u>

User-based collaborative filtering is based on similarity between users.

GridSearchCV is used to find the best k (neighborhood size) in a range of 1 to 50 and similarity functions using both cosine and Pearson correlation are provided.

Best parameters for Used-based collaborative filtering are seen for k= 11 and similarity measure using pearson correlation, i.e. the model uses 11 most similar users based on Pearson correlation to predict how a target user will rate an article. This implies moderate user similarity.

RMSE on the test set is seen at 0.97.

```
In [76]: print("Best Parameters (User-Based):", gs_user.best_params['rmse'])
Best Parameters (User-Based): {'k': 11, 'sim_options': {'name': 'pearson', 'user_based': True}}

In [77]: print("User-Based CF - Best RMSE:", rmse_user)
User-Based CF - Best RMSE: 0.9694270087312462
```

Item-Based Collaborative Filtering

Item-Based Collaborative Filtering is based on similarity between articles. Similar to user-based CF, I have used a GridSearchCV with the same choices to find the best parameters.

Best parameters for Item-based collaborative filtering are seen for k= 41 and similarity measure using cosine similarity, i.e. it needs a larger neighborhood (41 articles) to perform reasonably, likely due to sparsity or fewer commonalities between items. Cosine similarity works better here probably because article rating vectors are sparse and binary (many 1s and at 2s at a distant second place).

RMSE on the test set is seen at 1.01, which is slightly higher than that seen for user-based collaborative filtering.

```
In [78]: print("Item-Based CF - Best RMSE:", rmse_item)
Item-Based CF - Best RMSE: 1.0098593023025169

In [79]: print("Best Parameters (Item-Based):", gs_item.best_params['rmse'])
Best Parameters (Item-Based): {'k': 41, 'sim_options': {'name': 'cosine', 'user_based': False}}
```

Matrix Factorization (SVD)

SVD defines a shared vector space for items and users.

Best parameters are n_factors = 1 and n_epochs = 10. It indicates that a single latent factor captures much of the variance in article preferences (possibly due to uniform user tastes or rating behavior). It could also hint at simple rating behavior, e.g., users rating mostly 1 or 2 which account for ~ 90% of all ratings. RMSE stands at 0.954.

```
In [80]: print("SVD (Matrix Factorization) - Best RMSE:", rmse_svd)
SVD (Matrix Factorization) - Best RMSE: 0.9539112164480148

In [81]: print("Best Parameters (SVD):", gs_svd.best_params['rmse'])
Best Parameters (SVD): {'n_factors': 1, 'n_epochs': 10, 'random_state': 42}
```

Results

Model	**Best RMSE**	**Best Parameters**
User-based CF	0.9694	K = 11, similarity = Pearson
Item-based CF	1.0099	K = 41, similarity = Cosine
Matrix factorization (SVD)	0.9539	N_factors = 1, n_epochs = 10

In summary, best model performance seems to be from SVD (Matrix factorization), even with minimal complexity (n_factors = 1). User-based collaborative filtering comes very close in terms of RMSE as a measure, which is only slightly higher than that seen for SVD. Item-Based CF, while intuitive, underperforms slightly—likely due to limited overlap in article ratings across users.

Further improvements could be explored by using hybrid models.