Task 1:

1. After running K-means clustering with Euclidean, Cosine and Jaccard similarity for **K=10**, comparing the SSEs:

Distance/Similarity	SSE
Euclidean means	442681199424.8207
Cosine K-means	**23044.41032140117**
Jaccard K-means	54963.1962767096

Cosine is better when we take SSEs into consideration.

2. Labeling each cluster using the majority vote label of the data points in that cluster, let us compare the predictive accuracies:

Distance/Similarity	Predictive accuracy
Euclidean means	0.6045604560456046
Cosine K-means	**0.6248624862486248**
Jaccard K-means	0.6213621362136214

Cosine is better when we take the predictive accuracies into consideration.

3. Setting up the same stop criteria:

when there is no change in centroid position

OR

when the SSE value increases in the next iteration

OR

when the maximum preset value (e.g., 500, here) of iteration is complete,

for Euclidean-K-means, Cosine-K-means, Jaccard-K-means and comparing the number of iterations and times to converge:

Distance/Similarity	Iterations required	Time to converge
Euclidean K-means	2	12.09s
Cosine K-means	2	30.07s
Jaccard K-means	6	55.07s

K-Means with Jaccard similarity takes more iterations and more time to converge.

4. Comparing the SSEs of Euclidean-K-means, Cosine-K-means, Jaccard-K-means w.r.t the three stop criteria:

    - when there is no change in centroid position
    - when the SSE value increases in the next iteration
    - when the maximum preset value (e.g., 100) of iterations is complete

Distance/Similarity	SSE
○ **When there is no change in centroid position**	
Euclidean K-means	437700193881.37317
Cosine K-means	23050.03207512884
Jaccard K-means	55504.60587632911
○ **when the SSE value increases in the next iteration**	
Euclidean K-means	449752970221.08386
Cosine K-means	22317.32468141656
Jaccard K-means	22317.32468141656
○ **when the 100 iterations are complete**	
Euclidean K-means	435498047162.77826
Cosine K-means	23049.285232414222
Jaccard K-means	55509.33915383961

Task 2: (Step3):

a. Read the data from "ratings_small.csv".
b. Mean Absolute Error & Root Mean Square Error

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}}$$

c. The average RMSE and MAE of the **Probabilistic Matrix Factorization** using SVD, under the 5-fold CV.

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0123  0.9987  1.0053  1.0085  1.0076  1.0065  0.0045
MAE (testset)     0.7817  0.7713  0.7773  0.7785  0.7784  0.7774  0.0034
Fit time          0.47    0.44    0.42    0.45    0.43    0.44    0.02
Test time         0.06    0.19    0.06    0.06    0.12    0.10    0.05
```

The average RMSE and MAE of the **User-based Collaborative Filtering** using KNNBasic, under the 5-fold CV.

```
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9713  0.9730  0.9582  0.9638  0.9719  0.9676  0.0057
MAE (testset)     0.7457  0.7477  0.7375  0.7396  0.7469  0.7435  0.0042
Fit time          0.04    0.04    0.05    0.04    0.04    0.04    0.00
Test time         0.73    0.73    0.86    0.73    0.73    0.76    0.05
```

The average RMSE and MAE of the **Item-based Collaborative Filtering** using KNNBasic, under the 5-fold CV.

```
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9383  0.9420  0.9305  0.9433  0.9265  0.9361  0.0066
MAE (testset)     0.7224  0.7254  0.7199  0.7274  0.7149  0.7220  0.0044
Fit time          1.70    1.55    1.51    1.51    1.55    1.56    0.07
Test time         3.19    3.10    3.14    3.13    3.26    3.16    0.05
```

d. Comparing mean performances

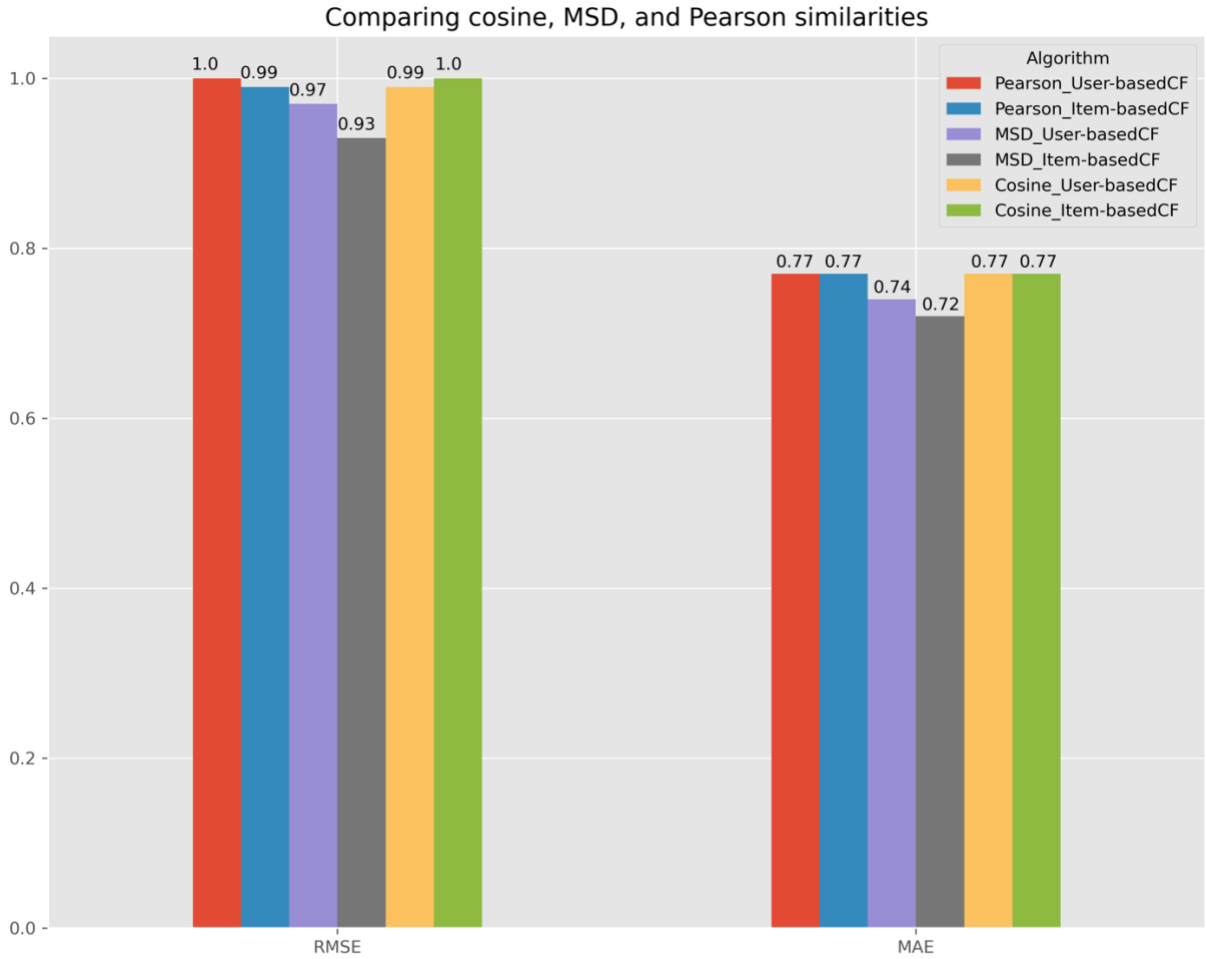
Algorithm	RMSE	MAE
PMF	1.009266	0.779577
User-based CF	0.967081	0.743234
Item-based CF	0.934426	0.720547

We can see here that **Item-based Collaborative filtering** is the best in movie rating data.

e. Examining the cosine, MSD, and Pearson similarities

Similarity	Algorithm	RMSE	MAE
Cosine	User-based CF	0.993192	0.767742
	Item-based CF	0.995095	0.774655
MSD	User-based CF	0.967186	0.743777
	Item-based CF	0.935363	0.721327
Pearson	User-based CF	0.998651	0.773473
	Item-based CF	0.988831	0.767704

The plot:

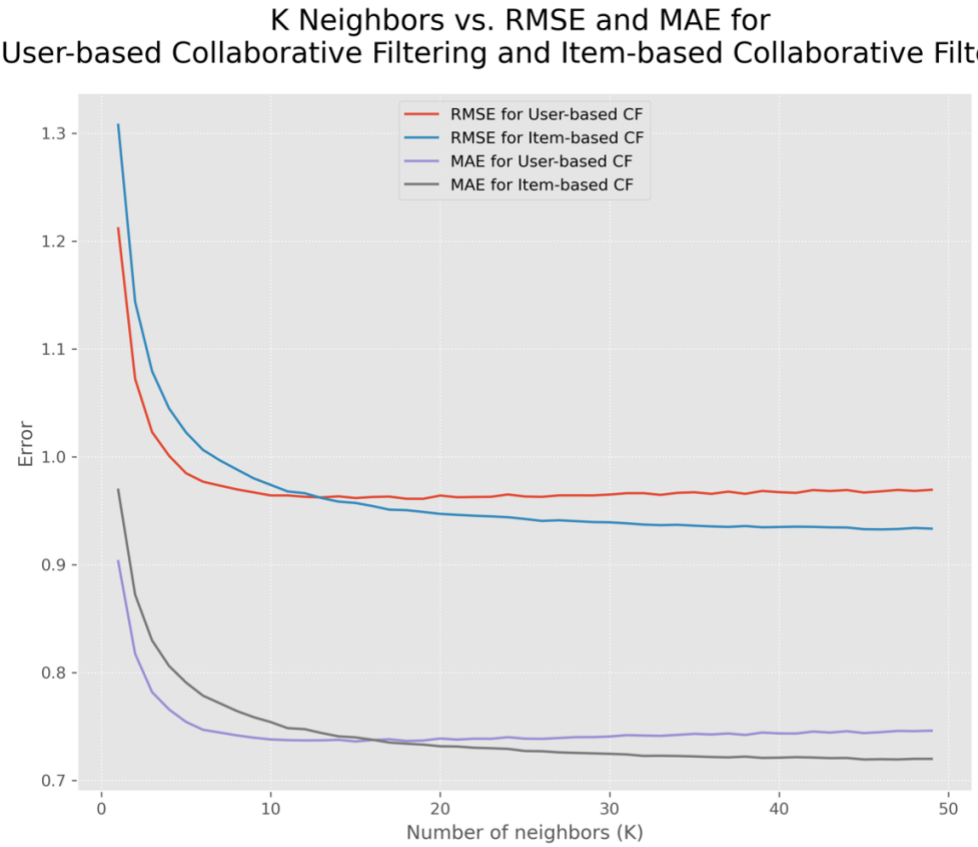


For **Pearson and MSD similarities**, the RMSE and MAE for **User-based Collaborative Filtering is more than Item-based Collaborative Filtering**.
However, for **Cosine similarity**, the RMSE and MAE for **User-based Collaborative Filtering is less than Item-based Collaborative Filtering**.

We see here, that the RMSE and MAE is the **least for MSD similarity** metric for **Item-based Collaborative Filtering**.

f. Examining how the number of neighbors impacts the performances of User-based Collaborative Filtering and Item-based Collaborative Filtering.

The plot:



g. For User-based CF,
the best number of neighbours **K = 19** with minimum **RMSE = 0.9610671476692758**

For Item-based CF,
the best number of neighbours **K = 46** with minimum **RMSE = 0.9326898721356003**

The best K of User-based collaborative filtering is **not the same** with the best K of Item-based collaborative filtering.

Please find my notebook with the K-Means Algorithm from scratch and the Recommender System, here: https://github.com/soumikg08/CAP5610_HW3/blob/main/Homework4.ipynb