Unsupervised learning and Recommender System

TASK 1

Q1 - After performing the K means clustering while considering the Euclidean, Jaccard, and cosine similarity. I received SSEs value for all three of those similarities.

Euclidean K means SSEs	Jaccard K means SSEs	Cosine K means SSEs
437667402482.1079	55505.653594541545	22937.06211946769

According to the SSEs value, the cosine technique is the best because it has the least SSEs value.

Output image Link -
https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/output_kmeans.png

Q2 - Calculating the Euclidean-K-means, Cosine-K-means, and Jaccard-K-means predictive accuracy, I then obtained the values.

Euclidean-K-means Acc	Cosine-K-means Acc	Jaccard-K-means Acc
59.325932593259324	54.485448544854485	60.33603360336034

According to the Accuracy value I got, the Jaccard technique is the best because it has the best Accuracy value.

Output image Link -
https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/output_kmeans.png

Q3 - After setting up the stopping criteria for Euclidean-K-means, Cosine-K-means, and Jaccard-K-means, I got the number of iterations as well as the times to computer those are as follows –

Distance/Similarity	Iterations occurs	Time to compute (ns)
Euclidean-K-means	2	18069515800
Cosine-K-means	2	73808076900
Jaccard-K-means	4	83443083300

Based on the Iterations and Times to compute**, Jaccard has more iterations,** as well as taking **more time to converge.**

Output image Link -
https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/output_kmeans.png

Q4 – Comparing the SSE's

- **when there is no change in centroid position**

Distance	Iterations	SSE	Time (nano sec)
Euclidean	49	436662984324.05646	150442189300
Cosine	64	23233.096804423258	317290038800
Jaccard	51	55029.60691079484	250149351400

- **when the SSE value increases in the next iteration**

Distance	Iterations	SSE	Time (nano sec)
Euclidean	2	423961138977.1962	26231543300
Cosine	2	22182.49257903548	137036179800
Jaccard	3	54529.20523561595	99051137400

- **when the maximum preset value (e.g., 100) of iteration is complete**

Distance	Iterations	SSE	Time (nano sec)
Euclidean	100	437653377087.5899	254514367900
Cosine	100	22404.317077260956	381218152400
Jaccard	100	54836.40709981178	336672716000

Output image Link -
https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/output_kmeans.png

Q5 – The observations I got are –

- The best SSE is provided by the cosine similarity metric, however when we examine accuracy, Jaccard has better accuracy than cosine.
- Accuracy increases with iterations.
- During the change in SSE value requires less computational due to less number of iterations.
- During the option between centroid stays the same or SSE criteria, it's always preferring SSE as the stopping criteria.
- SSE converges too rapidly yet produces results with very poor accuracy.

TASK 2

A – Reading data in the asked format.

Code snapshot –

https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/Task2_code.png

B – MAE and RMSE

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

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

C – Average MAE and RMSE

- **Average MAE and RMSE of probabilistic Matrix Factorization using 5_folds CV**

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE	1.0153	1.0080	1.0135	1.0104	1.0079	1.0110	0.0030
MAE	0.7826	0.7771	0.7840	0.7808	0.7784	0.7806	0.0025
Fit time	0.80	0.81	0.82	0.82	0.80	0.81	0.01
Test time	0.09	0.09	0.09	0.09	0.09	0.09	0.00

- **Average MAE and RMSE of User Based Collaborative Filtering using 5_folds CV**

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE	0.9677	0.9675	0.9687	0.9624	0.9687	0.9670	0.0023
MAE	0.7448	0.7440	0.7443	0.7380	0.7463	0.7435	0.0028
Fit time	0.13	0.14	0.15	0.15	0.14	0.14	0.01
Test time	1.04	1.08	1.09	1.10	1.16	1.09	0.04

- **Average MAE and RMSE of Item Based Collaborative Filtering using 5_folds CV**

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE	0.9334	0.9410	0.9320	0.9352	0.9292	0.9342	0.0040
MAE	0.7216	0.7250	0.7154	0.7194	0.7196	0.7202	0.0032
Fit time	2.03	2.09	2.36	2.13	2.18	2.16	0.11
Test time	4.45	5.09	4.60	4.64	4.59	4.67	0.22

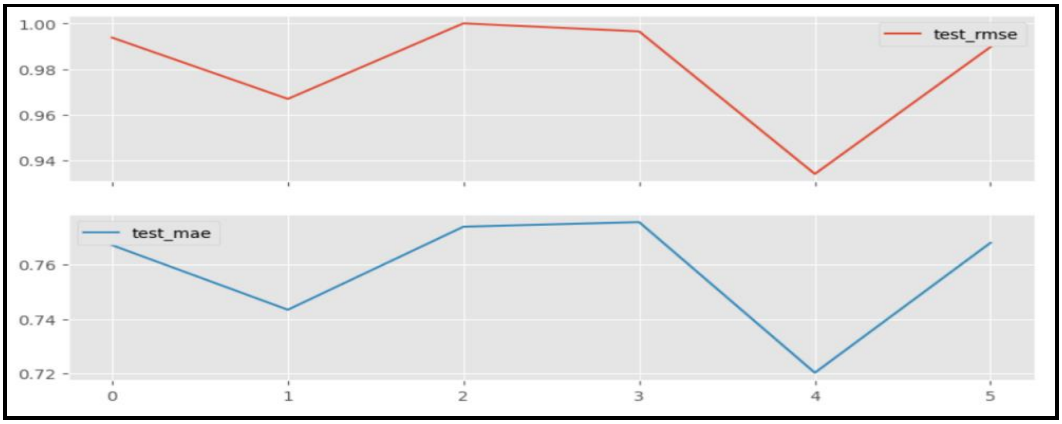
D – Comparing the mean performances of User-based collaborative filtering, item-based collaborative filtering, PMF wrt RMSE and MAE.

	test_rmse	test_mae	fit_time	test_time	Algorithm
0	1.011035	0.780587	0.811220	0.089410	PMF
1	0.966985	0.743501	0.141395	1.094149	User-basedCF
2	0.934169	0.720205	2.161015	4.674346	Item-basedCF

These benchmarks show that Item Based Collaborative Filtering is the best model since it has the lowest RMSE and MAE values.

E – Examining the MSD, Cosine and Pearson similarities**.**

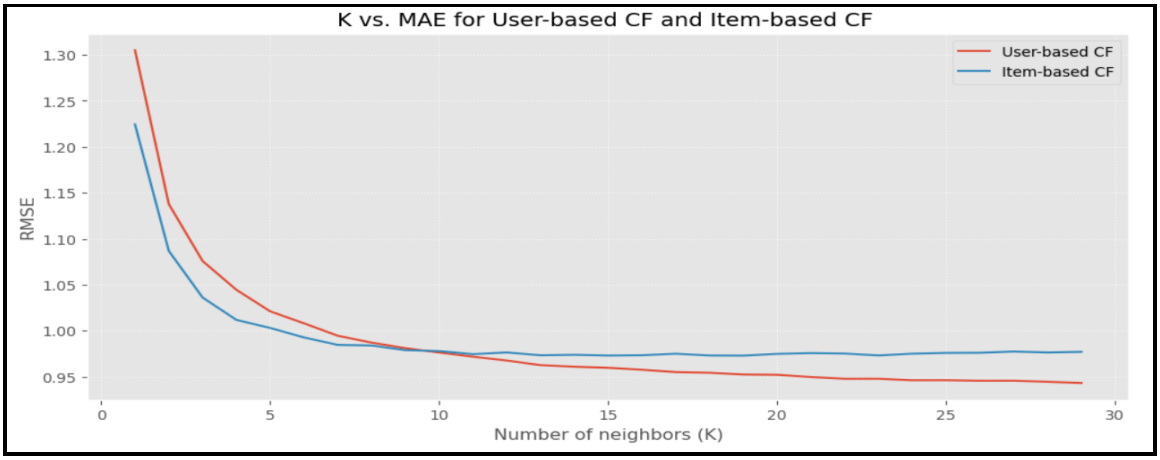
Algorithm	test_rmse	test_mae	fit_time	test_time
Pearson_User-basedCF	1.000054	0.773898	0.370466	1.034487
Pearson_Item-basedCF	0.989708	0.767996	5.458541	4.701479
MSD_User-basedCF	0.966971	0.743427	0.131374	1.019393
MSD_Item-basedCF	0.934142	0.720323	2.166329	4.820768
Cosine_User-basedCF	0.993788	0.767107	0.301041	1.051982
Cosine_Item-basedCF	0.996492	0.775567	4.066782	4.421279

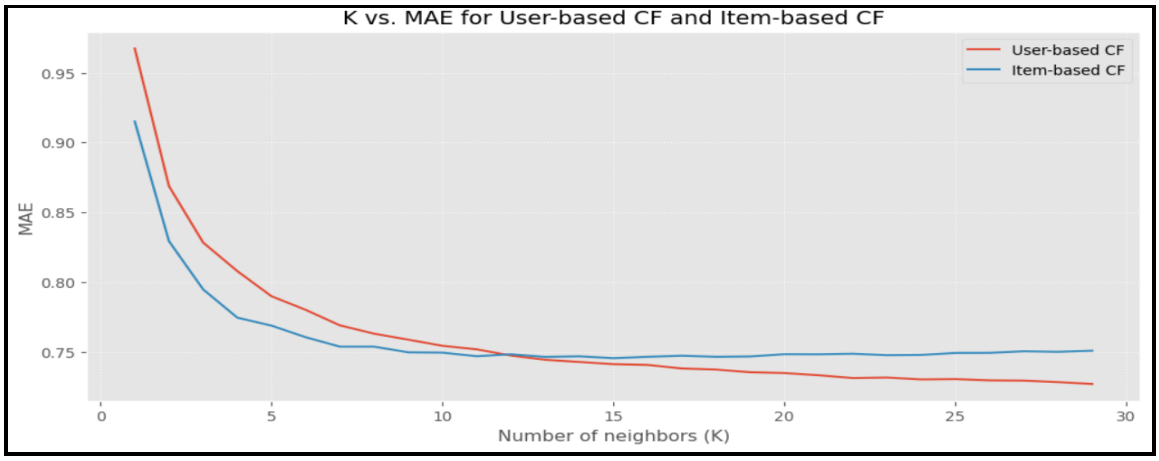


The various similarity measures do not have an equal influence on User-based and Item-based Collaborative Filtering –

- Collaborative Item-based Filtering: Cosine performs worse than Pearson.
- Collaborative Filtering for MSD is the best among all three algorithms.
- Fit as well as Test time computation for item based Collaborative Filtering is more than user based.
- Overall user based Collaborative Filtering for Pearson is worse.

F – Plotting the results based on Examining the number of neighbors has an influence on how well User-based and Item-Based CF function.





G – Getting the best no of neighbours

The best number of neighbours **K = 29** with minimum RMSE: 0.9439002371571993 and MAE: 0.7283062394504819 For **User_based CF**

The best number of neighbours **K = 16** with minimum RMSE: 0.973225414995104 and MAE: 0.7465617040160293 For **Item_based CF**

The best number of neighbors, i.e. K, varies between the two algorithms.

Task 1 & Task 2 Notebook Link –
https://github.com/Har1743/Hardik_Titanic_Data_Set_ML_Training/blob/main/HW3/HW3%20Ml.ipynb