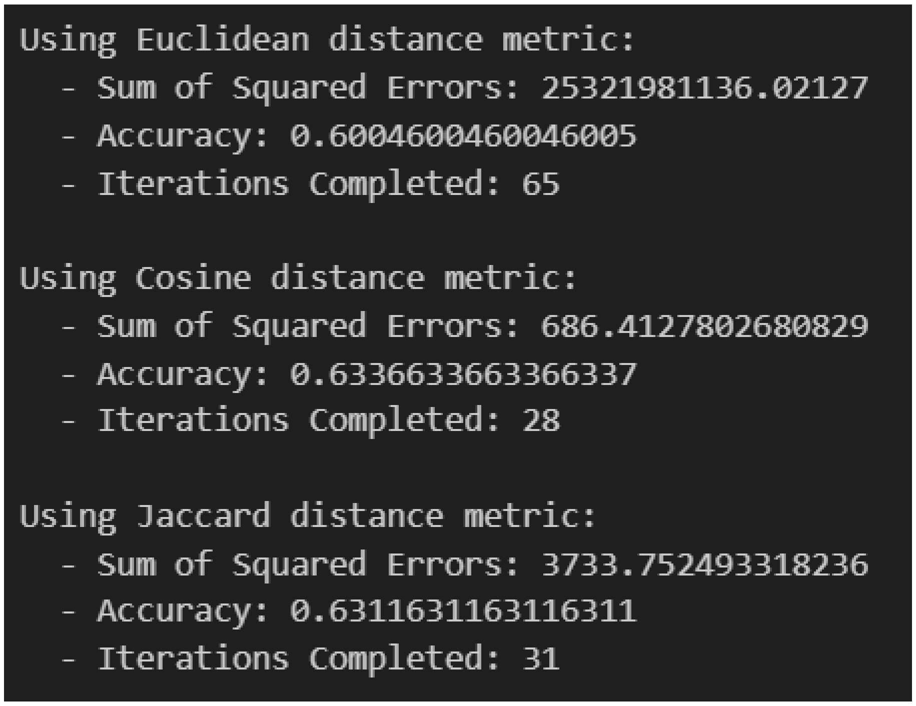
CSE 572 Data Mining
Dhinesh Babu Ramachandran
1229624580
Spring 24

Question 1

Stopping condition: Same as Question 3.



```
Using Euclidean distance metric:
  - Sum of Squared Errors: 25321981136.02127
  - Accuracy: 0.6004600460046005
  - Iterations Completed: 65

Using Cosine distance metric:
  - Sum of Squared Errors: 686.4127802680829
  - Accuracy: 0.6336633663366337
  - Iterations Completed: 28

Using Jaccard distance metric:
  - Sum of Squared Errors: 3733.752493318236
  - Accuracy: 0.6311631163116311
  - Iterations Completed: 31
```

Results run on given dataset. All questions answered based on this.

Task 1

The **Cosine-K-means** method has the lowest Sum of Squared Errors (SSE), making it the most effective in minimizing within-cluster variances. The high SSE for Euclidean-K-means indicates that data points are, on average, farther from their centroids, suggesting less cohesion within clusters.

Task 2

The **Cosine-K-means** method achieves the highest accuracy, suggesting it's more effective at grouping data points into clusters that align with the true categorizations. Given that almost all the 3 have a similar accuracy compared to SSE which changes Very drastically, **Accuracy** is the better metric.

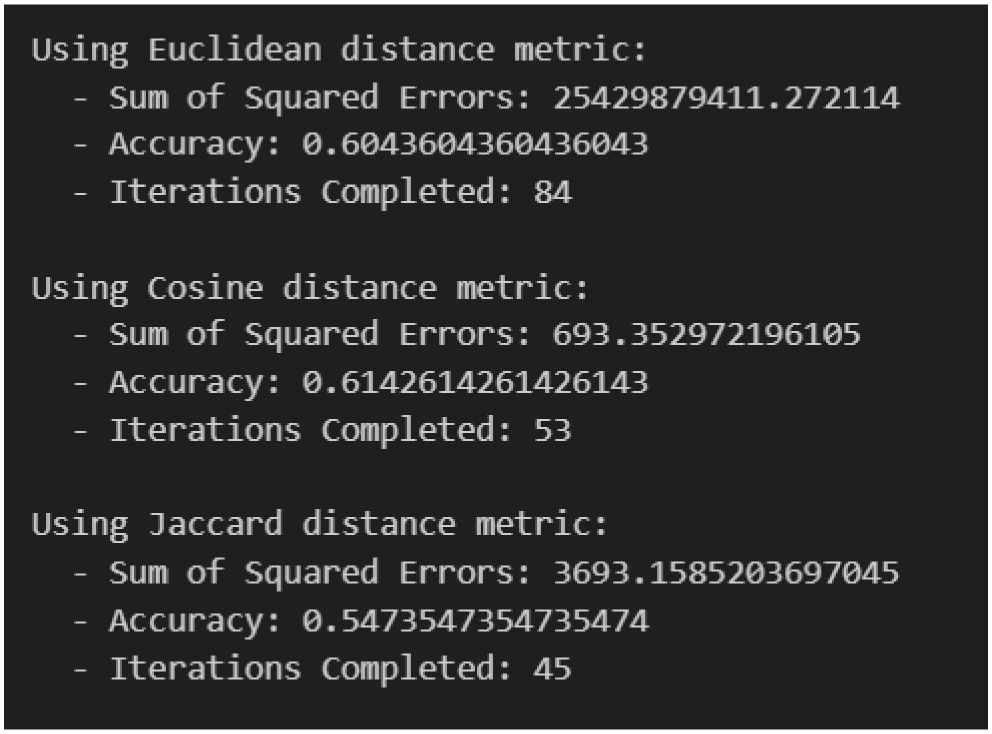
Task 3

The **Euclidean-K-means** method requires significantly more iterations and time to converge, suggesting it may have a harder time finding stable centroids compared to the other methods.

Task 4

SSE

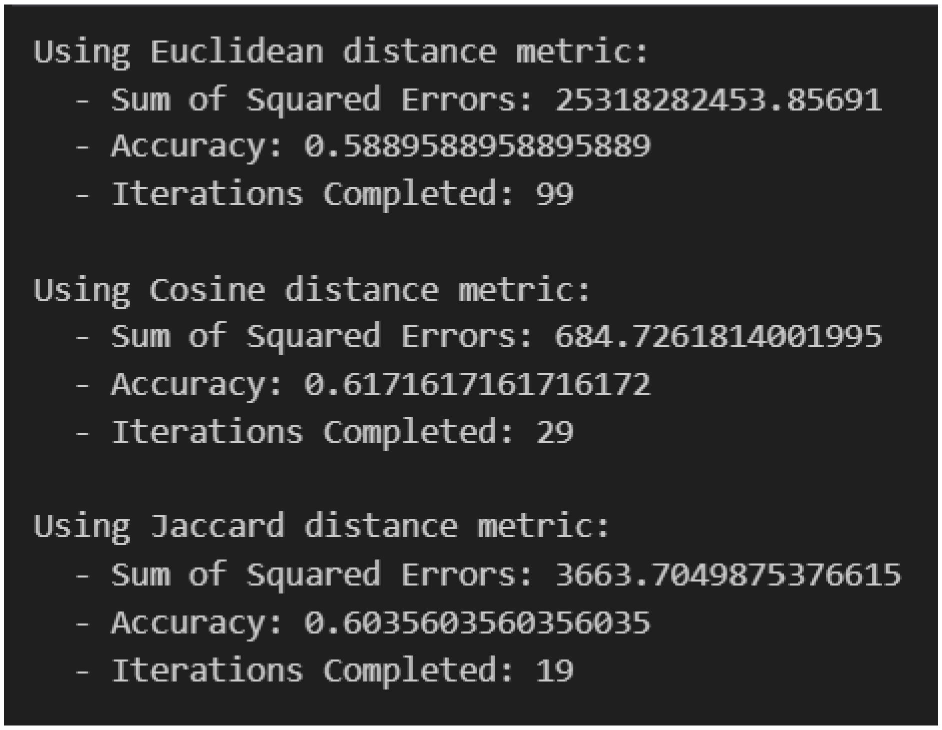
No change in Centroid



```
Using Euclidean distance metric:
  - Sum of Squared Errors: 25429879411.272114
  - Accuracy: 0.6043604360436043
  - Iterations Completed: 84

Using Cosine distance metric:
  - Sum of Squared Errors: 693.352972196105
  - Accuracy: 0.6142614261426143
  - Iterations Completed: 53

Using Jaccard distance metric:
  - Sum of Squared Errors: 3693.1585203697045
  - Accuracy: 0.5473547354735474
  - Iterations Completed: 45
```

Cosine < Jaccard < Euclidean

SSE value increases

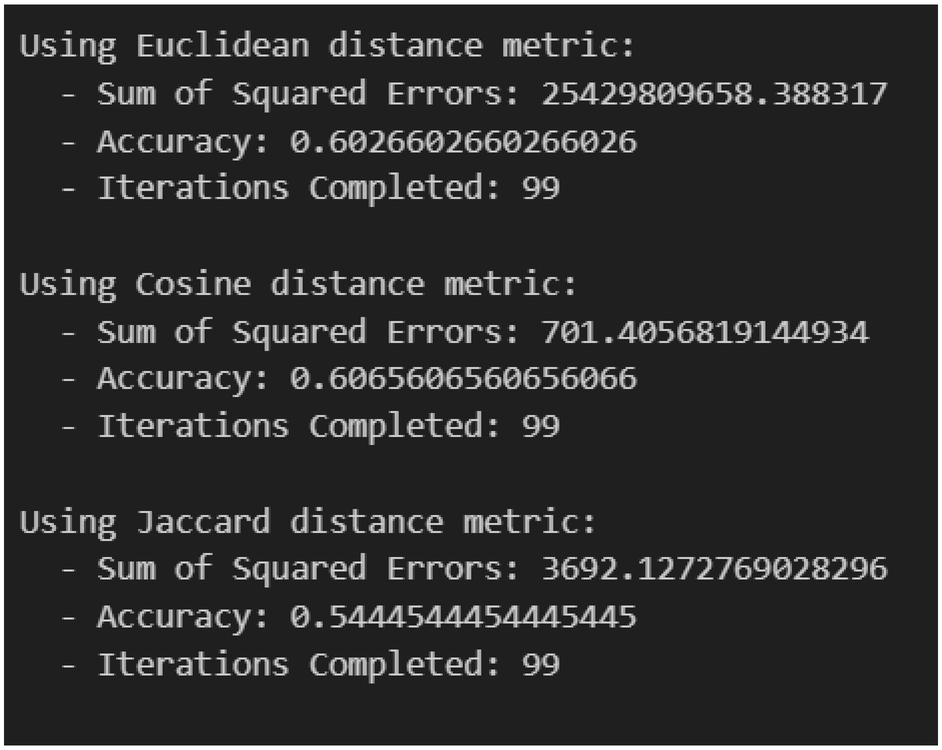


```
Using Euclidean distance metric:
  - Sum of Squared Errors: 25318282453.85691
  - Accuracy: 0.5889588958895889
  - Iterations Completed: 99

Using Cosine distance metric:
  - Sum of Squared Errors: 684.7261814001995
  - Accuracy: 0.6171617161716172
  - Iterations Completed: 29

Using Jaccard distance metric:
  - Sum of Squared Errors: 3663.7049875376615
  - Accuracy: 0.6035603560356035
  - Iterations Completed: 19
```

Cosine < Jaccard < Euclidean

(Note: It hit iteration limit i set of 100 so that it doesn't run for a long time.)

Maximum preset value. (100)



```
Using Euclidean distance metric:
  - Sum of Squared Errors: 25429809658.388317
  - Accuracy: 0.6026602660266026
  - Iterations Completed: 99

Using Cosine distance metric:
  - Sum of Squared Errors: 701.4056819144934
  - Accuracy: 0.6065606560656066
  - Iterations Completed: 99

Using Jaccard distance metric:
  - Sum of Squared Errors: 3692.1272769028296
  - Accuracy: 0.5444544454445445
  - Iterations Completed: 99
```

Cosine < Jaccard < Euclidean

Task 5

Key takeaways for this dataset:

1. Sum of Squared Errors (SSE):
   - Cosine similarity outperformed the rest with the lowest SSE, suggesting a superior fit to the data over Euclidean and Jaccard.
2. Accuracy:
   - Cosine similarity yielded the highest accuracy consistently, confirming its effectiveness in precisely grouping data points according to the dominant label in each cluster.
3. Iteration and Convergence:
   - The Euclidean measure took the longest to reach convergence, hinting at potential computational inefficiencies in contrast to other measures.
4. Stopping Conditions and SSEs:
   - Cosine similarity maintained lower SSE values across various stopping conditions, underscoring its robustness and better clustering performance relative to Euclidean and Jaccard.
5. Overall Insights:
   - Cosine similarity stands out as the most advantageous for this particular dataset, with commendable lower SSEs, superior accuracy, and faster convergence rates.
   - The traditional Euclidean measure falls short in efficiency for this dataset, evident from more iterations to convergence and higher SSEs.
   - Jaccard's performance sits between Cosine and Euclidean regarding convergence speed and SSE, but it doesn't surpass Cosine in any evaluated aspect.

Key takeaways in general:

1. **Choice of Distance Metric**: The performance of k-means is greatly influenced by the distance metric used. While Euclidean is standard, Cosine and Jaccard metrics can offer higher accuracy for certain data types, highlighting the importance of selecting a metric that aligns with the dataset's characteristics.
2. **Efficiency and Accuracy**: Implementing a nuanced stopping criterion that considers changes in SSE, centroid movements, and a maximum iteration limit tends to enhance the algorithm's efficiency and accuracy. This approach, particularly with Cosine and Jaccard metrics, resulted in fewer iterations without compromising accuracy.
3. **Impact of Stopping Criteria**: The algorithm's effectiveness and efficiency are sensitive to the stopping criteria. A combined approach, considering several factors like SSE change and centroid stability, generally yields better results than using a single criterion.
4. **Adaptability to Data**: The varying performance across different metrics and stopping criteria underscores the necessity of adapting the k-means algorithm to the dataset's unique needs, ensuring an optimal balance between computational efficiency and clustering accuracy.

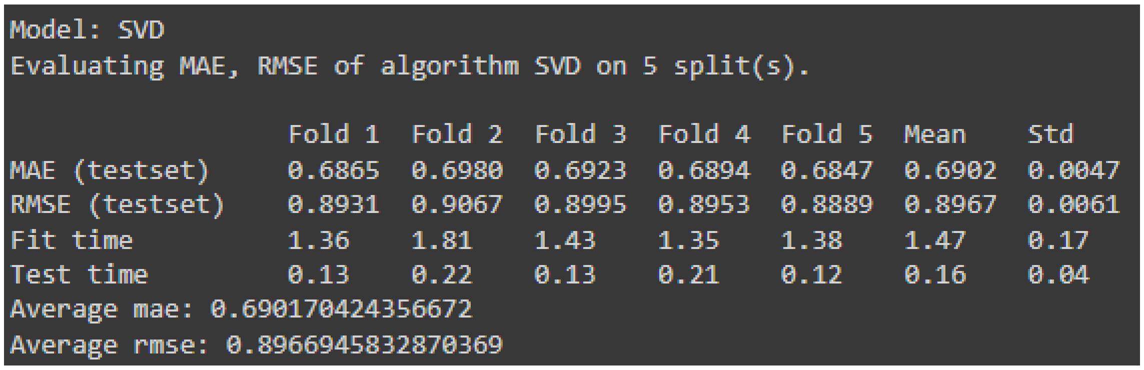
In summary, the analysis emphasizes the importance of carefully selecting both distance metrics and stopping criteria for the k-means algorithm to achieve the best balance of efficiency and accuracy tailored to the specific dataset.

Question 2

Compute the average MAE and RMSE of the Probabilistic Matrix Factorization (PMF), User based Collaborative Filtering, Item based Collaborative Filtering, under the 5-folds cross-validation (10 points)

PMF

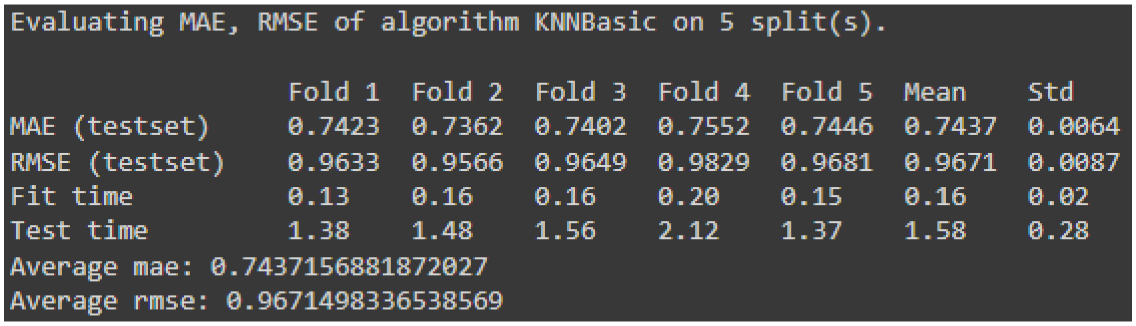
Using SVD for PMF.



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

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MAE (testset)     0.6865  0.6980  0.6923  0.6894  0.6847  0.6902  0.0047
RMSE (testset)    0.8931  0.9067  0.8995  0.8953  0.8889  0.8967  0.0061
Fit time          1.36    1.81    1.43    1.35    1.38    1.47    0.17
Test time         0.13    0.22    0.13    0.21    0.12    0.16    0.04
Average mae: 0.690170424356672
Average rmse: 0.8966945832870369
```

User based Collaborative Filtering

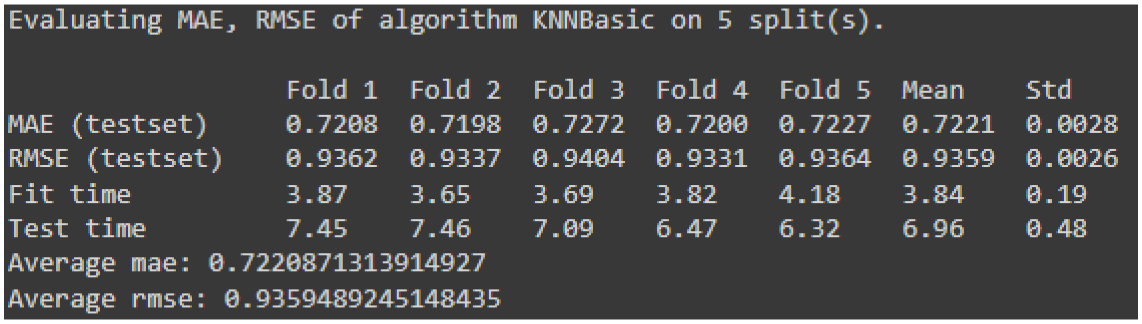
Using KNN for User based Collaborative Filtering.



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

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MAE (testset)     0.7423  0.7362  0.7402  0.7552  0.7446  0.7437  0.0064
RMSE (testset)    0.9633  0.9566  0.9649  0.9829  0.9681  0.9671  0.0087
Fit time          0.13    0.16    0.16    0.20    0.15    0.16    0.02
Test time         1.38    1.48    1.56    2.12    1.37    1.58    0.28
Average mae: 0.7437156881872027
Average rmse: 0.9671498336538569
```

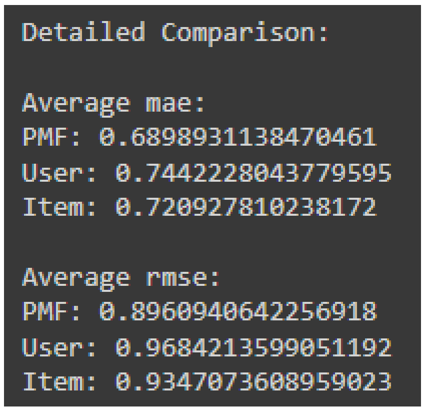
Item based Collaborative Filtering

Using KNN for Item based Collaborative Filtering.



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

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
MAE (testset)     0.7208  0.7198  0.7272  0.7200  0.7227  0.7221  0.0028
RMSE (testset)    0.9362  0.9337  0.9404  0.9331  0.9364  0.9359  0.0026
Fit time          3.87    3.65    3.69    3.82    4.18    3.84    0.19
Test time         7.45    7.46    7.09    6.47    6.32    6.96    0.48
Average mae: 0.7220871313914927
Average rmse: 0.9359489245148435
```

Compare the average (mean) performances of User-based collaborative filtering, item-based collaborative filtering, PMF with respect to RMSE and MAE. Which ML model is the best in the movie rating data? (10 points)

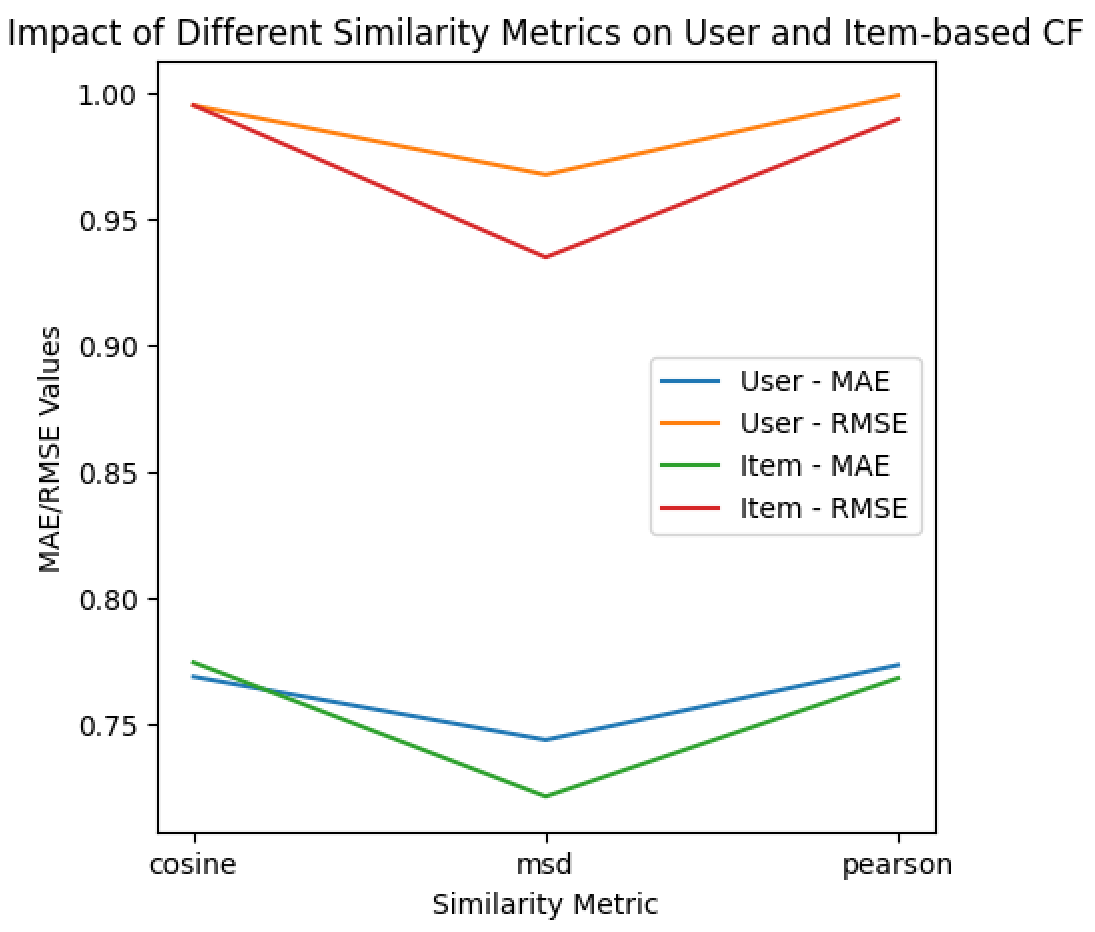


```
Detailed Comparison:

Average mae:
PMF: 0.6898931138470461
User: 0.7442228043779595
Item: 0.720927810238172

Average rmse:
PMF: 0.8960940642256918
User: 0.9684213599051192
Item: 0.9347073608959023
```

The lowest error rate is in the PMF model for both metrics.
Hence,**PMF** is the best model in this situation.

Examine how the cosine, MSD (Mean Squared Difference), and Pearson similarities impact the performances of User based Collaborative Filtering and Item based Collaborative Filtering. Plot your results. Is the impact of the three metrics on User based Collaborative Filtering consistent with the impact of the three metrics on Item based Collaborative Filtering? (10 points)

Performance of User-based and Item-based Collaborative Filtering (CF) algorithms, as evaluated by the MAE (Mean Absolute Error) and RMSE (Root Mean Square Error).



From the graph, we observe the following trends:

For User-based CF:

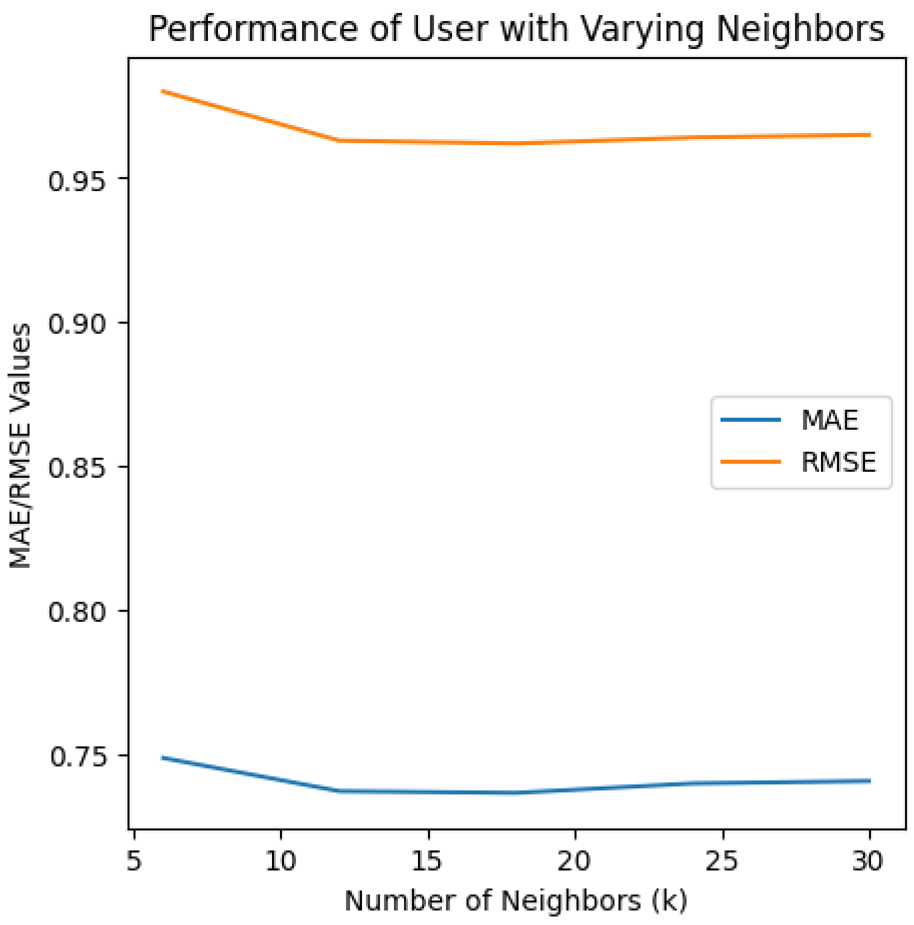
- Both MAE and RMSE are relatively stable across the three similarity metrics.
- The RMSE appears slightly lower with the MSD similarity metric, suggesting that it may provide a better performance for User-based CF.
  For Item-based CF:
- The MSD similarity metric results in a notably lower MAE and RMSE compared to cosine and Pearson, indicating that MSD is more effective for Item-based CF.
- Cosine and Pearson metrics result in higher error rates, with Pearson having the highest, implying these may not be as effective as MSD for Item-based CF.

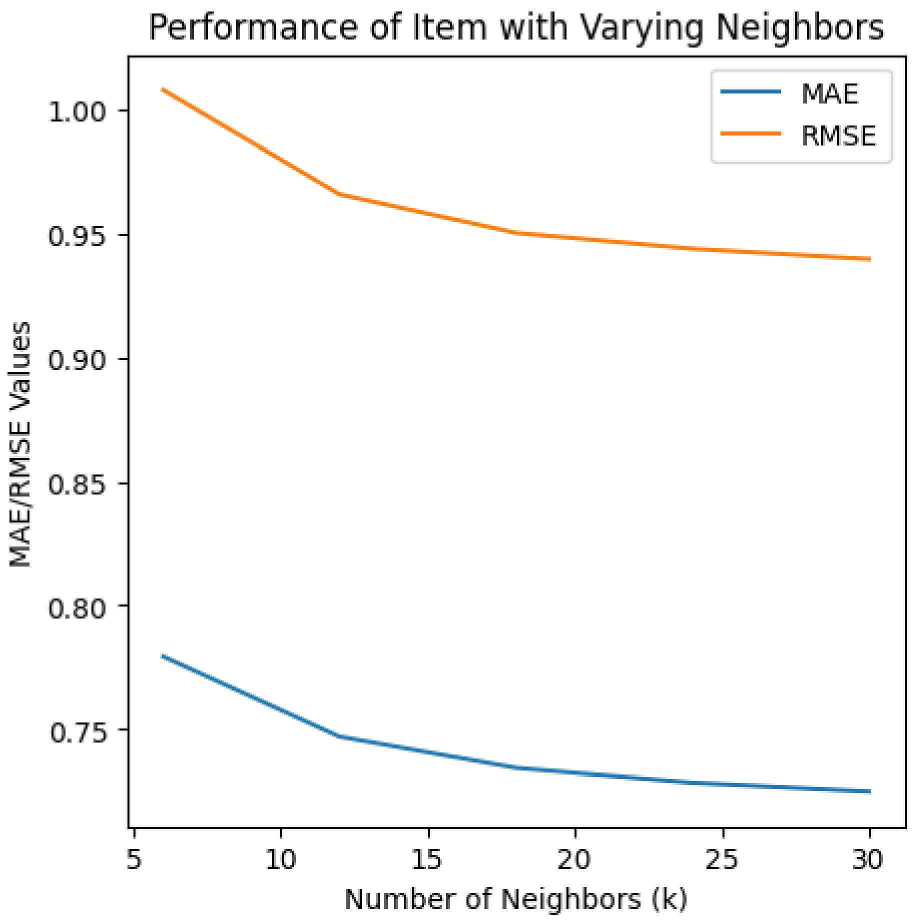
When comparing the impact of the metrics between User-based and Item-based CF, it seems that MSD is consistently the best performer for both in terms of resulting in the lowest MAE and RMSE. However, the difference in performance between the metrics is more pronounced for Item-based CF than for User-based CF.

The consistency is that MSD remains the superior metric for both CF approaches. The inconsistency lies in the magnitude of the impact: it is significantly more beneficial for Item-based CF, improving performance to a greater degree than for User-based CF. This suggests that the choice of similarity metric can be especially crucial for Item-based CF.

In conclusion, while there is a consistent trend that MSD is the best similarity metric across both types of CF, the impact is more pronounced for Item-based CF, suggesting that Item-based CF may be more sensitive to the choice of similarity metric.

Examine how the number of neighbors impacts the performances of User based Collaborative Filtering and Item based Collaborative Filtering? Plot your results. (10 points)





User-based Collaborative Filtering:

- **MAE**: Decreases significantly as the number of neighbors increases from 5 to around 10, then flattens out, indicating that additional neighbors beyond this point do not contribute much to reducing error.
- **RMSE**: Stays relatively consistent across different k values, showing only a slight decrease as the number of neighbors increases.

Item-based Collaborative Filtering (Second Graph):

- **MAE**: Decreases as the number of neighbors increases, showing a steeper descent compared to the User-based CF graph, indicating that more neighbors contribute more consistently to improving accuracy.
- **RMSE**: Also decreases as the number of neighbors increases but levels off after k reaches around 15. This suggests that beyond 15 neighbors, additional neighbors do not significantly improve the RMSE.

Comparison:

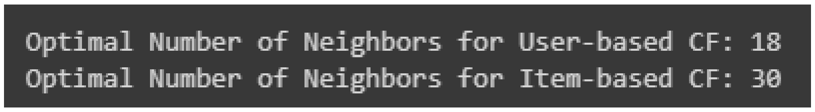
- For both User-based and Item-based CF, increasing the number of neighbors initially leads to a decrease in MAE, with the effect being more pronounced for Item-based CF.
- RMSE is relatively insensitive to the number of neighbors in User-based CF but shows improvement with an increase in neighbors up to a point in Item-based CF.

Conclusions:

- **Item-based CF** seems to be more sensitive to the number of neighbors. It benefits from a larger k, especially in terms of MAE.
- The optimal k for **User-based CF** could be around 10 based on the MAE plateau, while for **Item-based CF**, it seems to be around 15 for RMSE and continues to improve for MAE even at 30.
- The "best" k is not necessarily the same for both methods; it varies depending on the specific performance metric (MAE or RMSE) and the type of CF (User-based or Item-based).

The optimal number of neighbors depends on both the type of collaborative filtering used and the performance metric considered. The best k for each approach would be where the improvement in the error rates starts to plateau, indicating the diminishing returns of adding more neighbors.

Identify the best number of neighbor (denoted by K) for User/Item based collaborative filtering in terms of RMSE. Is the best K of User based collaborative filtering the same with the best K of Item based collaborative filtering? (10 points)



```
Optimal Number of Neighbors for User-based CF: 18
Optimal Number of Neighbors for Item-based CF: 30
```

No it the best k is not the same between the 2 filtering. User filtering performs worse as more neighbors are added after 18 while item-based seems to perform better as more neighbors are added according to the graph.

Code link https://github.com/Dhinesh-Babu/DataMining/tree/main/hw-3