Ailin Chu

Weihao He

Xiao Qin

MSAI 349-Homework #2

2 November 2024

K-Nearest Neighbors and K-Means - Homework Report

Part I

Basic Function Implementation & KNN classifier

We transform our data using normalization, which scales data to a range [0,1], which can speed up convergence, improves training stability and reduces sensitivity to feature magnitude.

```
_____Using euclidean distance_____
euclidean program execution time: 35.99636101722717 s
Accuracy: 89.50%
Confusion Matrix:
[[18  0  0  0  0  0  0  0  0]
 [ 0 27  0  0  0  0  0  0  0]
 [ 0  1 17  0  0  0  0  1  0]
 [ 0  1  0 16  0  0  0  0  1]
 [ 0  0  0  0 22  0  0  0  3]
 [ 0  0  0  1  0 10  1  0  1]
 [ 0  0  0  0  0  0 13  0  0]
 [ 0  1  0  0  0  0  0 23  0]
 [ 1  1  1  2  0  2  0  0 14]
 [ 0  2  0  0  0  0  1  0  0 19]]
```

```
_____Using cosim similarity_____
cosim program execution time: 68.42428493499756 s
Accuracy: 92.00%
Confusion Matrix:
[[18  0  0  0  0  0  0  0  0]
 [ 0 27  0  0  0  0  0  0  0]
 [ 0  0 18  0  0  0  0  1  0]
 [ 1  0  0 15  0  0  0  0  2]
 [ 0  0  0  0 22  0  0  0  3]
 [ 0  0  0  0  0 12  1  0  0]
 [ 0  0  0  0  0  0 13  0  0]
 [ 0  0  0  0  0  0  0 24  0]
 [ 1  0  1  1  0  0  0  0 18]
 [ 1  1  0  0  1  1  1  0  0 17]]
```

We also adopt binarization technique, which converts data to a binary format by setting a threshold 60. Binarization can benefit our data transformation to reduce noise and improve memory efficiency.

```
_____Using euclidean distance_____
euclidean program execution time: 38.34145426750183 s
Accuracy: 91.50%
Confusion Matrix:
[[18  0  0  0  0  0  0  0  0]
 [ 0 27  0  0  0  0  0  0  0]
 [ 0  1 17  0  0  0  0  1  0]
 [ 0  1  0 16  0  0  0  0  1]
 [ 0  0  0  0 23  0  0  0  0  2]
 [ 0  0  0  0  0 11  1  0  1  0]
 [ 0  0  0  0  0  0 13  0  0  0]
 [ 0  1  0  0  0  0  0 23  0  0]
 [ 0  1  1  1  0  1  0  0 17  0]
 [ 1  2  0  0  0  0  1  0  0 18]]
```

```
_____Using cosim similarity_____
cosim program execution time: 75.0033609867096 s
Accuracy: 93.00%
Confusion Matrix:
[[18  0  0  0  0  0  0  0  0  0]
 [ 0 26  0  0  0  0  1  0  0  0]
 [ 0  0 18  0  0  0  0  0  1  0]
 [ 1  0  1 15  0  0  0  0  1  0]
 [ 0  0  0  0 23  0  0  0  0  2]
 [ 0  0  0  1  0 11  1  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  0]
 [ 0  0  0  0  0  0  0 24  0  0]
 [ 1  0  0  1  0  0  0  0 19  0]
 [ 2  1  0  0  0  0  0  0  0 19]]
```

However, when we combine these two data transformation tools, our accuracy drops to around 21%, that is what we are still trying to figure out.

K-means classifier

As for the K-means algorithm, we adjust the number of K clusters from 50 to 100 to seek better performance. (Matrix: Cosine Similarity; Iteration = 5)

	k=50	k=80	k=100
Accuracy	76%	79%	82%
Homogeneity Score	0.672	0.718	0.756

We use Homogeneity score as the quantitative matrix to measure how well our clusters align with the given label. Homogeneity score is a metric that measures how well a clustering algorithm groups data points into clusters. It is bounded between 0 and 1, with lower values indicating lower homogeneity, which represent a worse performance in capturing the true class structure of the data. As we can see in the table above, as we increase the number of clusters, accuracy and homogeneity score increase as well. This might show that our clustering algorithm learns more detailed information from our training data, and performs an improvement in prediction accuracy.

Part II

Collaborative Filtering Algorithm Description

Our collaborative filtering approach uses a user-based collaborative filtering model. The core idea is to recommend items(movies) to a user based on the preferences of other users with similar interests. This is achieved by calculating a similarity score between users based on their historical ratings.

The algorithm we implement follows these main steps:

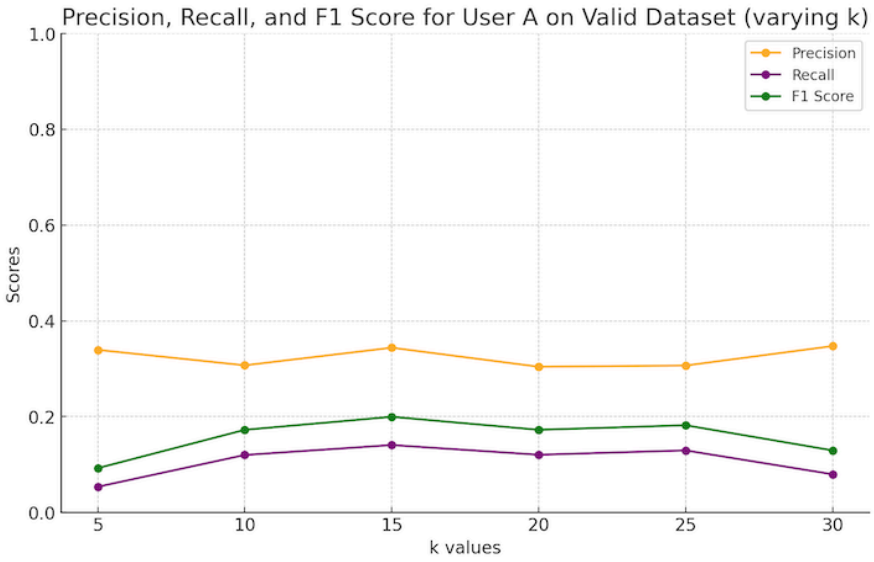
1. **Construct the Rating Matrix:** We build a sparse matrix where each row represents a user, each column represents a movie, and each cell contains the user's rating for that movie. If a user hasn't rated a movie, the cell remains zeros.
2. **Z-score Normalization (Optional):** Since users have different rating scales, we normalize each user's ratings by applying a z-score standardization, transforming each user's ratings to have a mean of 0 and a standard deviation of 1. This helps ensure that differences in individual rating scales don't affect the similarity calculations.
3. **Calculate User Similarities:** Using the normalized rating matrix, we calculate the similarity between the target user and other users. We employ cosine and euclidean metric, which measures the similarity between two users rating vectors.

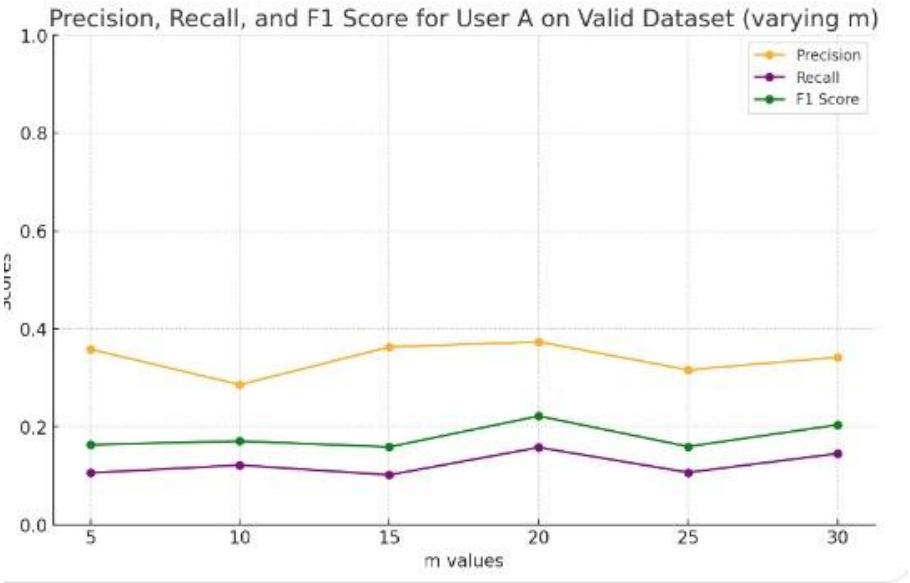
4. **Identify Similar Users:** Based on the similarity scores, we select the top k most similar users to the target user. These users' ratings are used to predict the target user's ratings for items they haven't seen,
5. **Score and Rank Items:** For each movie the target user hasn't rated, we calculate a weighted average rating using the ratings from the k most similar users, weighted by the similarity scores. This gives an estimated rating for each unseen movie, allowing us to rank the movies in descending order of preference. We then recommend the top m movies with the highest scores.

Hyperparameters:

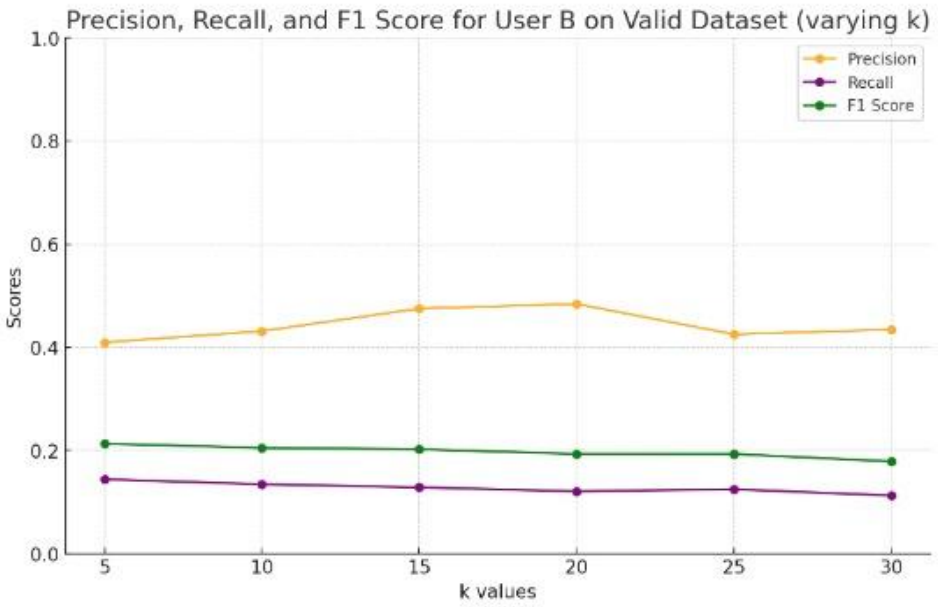
1. **k :** how many similar users to include when calculating a target user's potential ratings. A higher k value captures a broader preference base, potentially increasing diversity in recommendations, but may dilute the influence of the most similar users. A smaller k may potentially enhance recommendation relevance.
2. **m:** This parameter defines the number of top-ranked items(movies) recommended to the user. It determines the length of the recommendation list shown to the user. A suitable m provides enough variety to the user without overwhelming them.

The experiments results on user a,b and c are as follows:

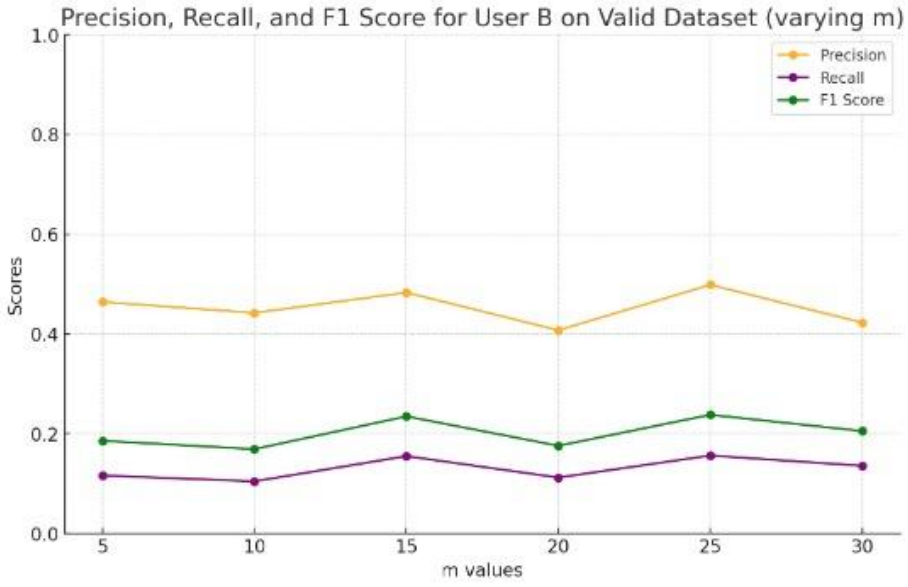


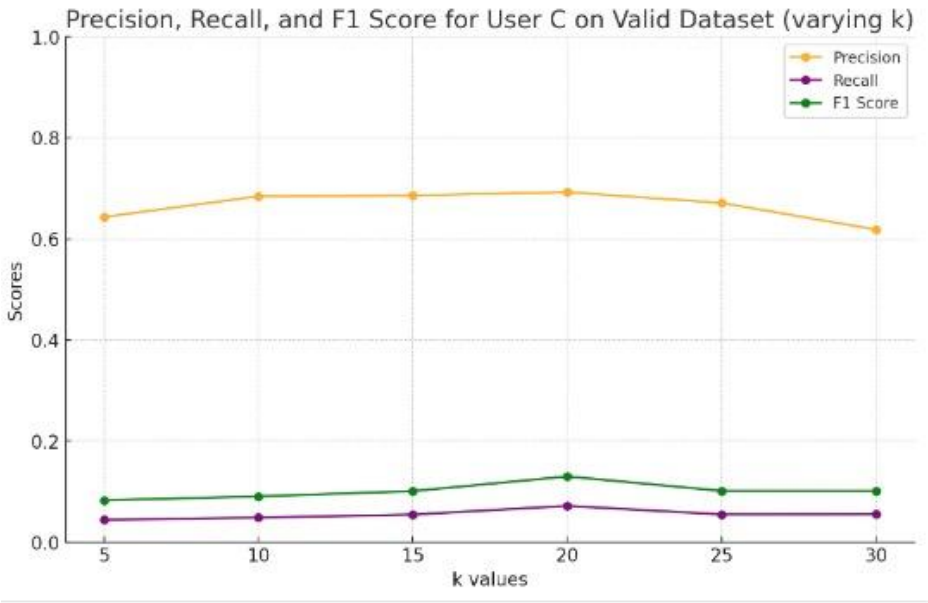


In the first image, varying k with m=20, second image varying m with k =20

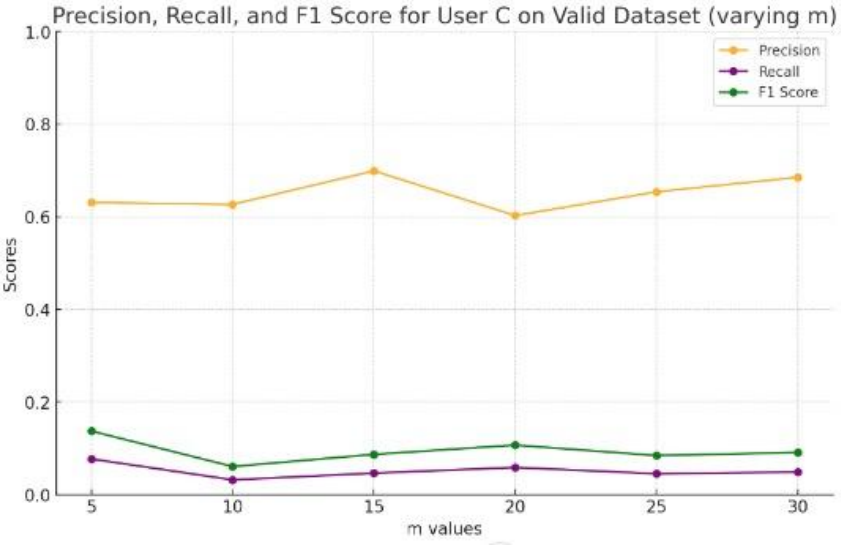


In the first image, varying k with m=20, second image varying m with k=20





In the first image, varying k with m=20, the second image, varying m with k=20



The results on the test set exhibit a similar pattern of variation to those on the respective validation sets, with comparable metrics.

After experiments, we noticed that adjusting m affects the precision, recall, and F1 score in the following ways:

1. As m increases, **precision** may decline. Precision is the ratio of relevant items in the recommended list, so including more items typically dilutes the relevance if the system starts recommending items that are less likely to be of interest. Smaller values of m focus on only the top, most confidently recommended items, leading to higher precision, whereas larger m values often introduce items that may not be as relevant.
2. Recall generally improves as m increases. **Recall** measures the proportion of relevant items successfully included in the recommendations out of all relevant items. By recommendating more items, the system is more likely to cover a larger share of relevant items, increasing recall.
3. When m is low, **F1 score** can be higher if the recommended items are highly relevant(high precision) but may decrease as recall is limited. As m increases, recall improves, potentially improving F1 if the decline in precision isn't too severe. The optimal m typically balances both precision and recall to maximize F1 score.

Enhancing Collaborative Filtering with Movie Genre and Demographics

1. **Construct User and Movie Feature Vectors:** User Demographics: Encode user features into a demographic feature vector; Movie Genres: Create a genre vector for each movie using one-hot encoding.
2. **Calculate Similarities Separately:** Rating Similarity: Compute user similarity based on their rating vectors; Demographic Similarity: Calculate similarity between users based on demographic vectors to find users with similar profiles. Genre Similarity: Measure similarity between movies based on genre vectors, ensuring recommendations match user genre preferences.
3. **Combine Similarities via Averaging :** Define weights to balance the contributions of rating, demographic, and genre similarities.
4. **Generate Recommendations:** Use the combined similarity scores to recommend movies to users, integrating both user preferences and content features.

Experiments results(K=20,m=20)

	Precision (Valid)	Precision (Test)	Recall (Valid)	Recall (Test)	F1-Score (Valid)	F1-Score (Test)
User a	0.32	0.27	0.05	0.07	0.09	0.11
User b	0.57	0.54	0.13	0.11	0.22	0.18
User c	0.69	0.61	0.09	0.08	0.16	0.14

After observation, we found that the collaborative filtering model showed a slight improvement overall after incorporating additional conditions, which supports the

validity of our approach. However, the effect was less noticeable for user c; in some parameter settings, the collaborative filtering performance even weakened. This may be because user c has seen too few movies and with too limited genre diversity, so introducing movie genres does not have a significant impact.