Algorithms for AI 3
Lab Work 2: “Recommender Engine”

By

Ananya Mehta, Jaanvi Das, Henil Shah

INTRODUCTION

Recommender systems have become an integral part of today's online experiences, guiding users toward content they are likely to enjoy. From movies and music to shopping and news, these systems analyze user behavior and item characteristics to offer personalized suggestions. This lab project focuses on the implementation and evaluation of two fundamental approaches to recommendation: cosine similarity-based matching and the k-Nearest Neighbors (k-NN) classifier.

TASK 1:

I. THEORY

A. Definition:

Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. It quantifies how similar two vectors are, regardless of their magnitude.

B. Mathematical Formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \, ||\mathbf{B}||}$$

where:

1. A · B = Dot product of A and B
2. ||A||, ||B|| = Euclidean norms (magnitudes) of A and B

C. Key Properties:

1. Range: [-1, 1], where
   - 1: Identical vectors (angle = 0°).
   - 0: Orthogonal vectors (no similarity).
   - -1: Opposite vectors (angle = 180°).
2. Works well for sparse, high-dimensional data (e.g., text).
3. Efficient for comparing item-item similarities.

II. METHODOLOGY

A. Libraries used:

1. pandas: For data manipulation.
2. sklearn.feature_extraction.text.CountVectorizer – For converting text into a bag-of-words numerical format.
3. sklearn.metrics.pairwise.cosine_similarity – To calculate similarity scores between movies.
4. rapidfuzz – For fuzzy matching between user input and movie titles.
5. re – For cleaning and standardizing text data using regular expressions.
6. random – To shuffle movie groups with identical similarity scores, adding diversity to recommendations.
7. itertools.groupby – To group similarity scores when multiple movies have the same value.

B. Data Loading and Preprocessing:

1. Irrelevant columns such as 'Audience score %', 'Profitability', 'Rotten Tomatoes %', 'Worldwide Gross', 'index', 'rating' are dropped.
2. Unique Values for columns ‘Genre’ and ‘Lead Studio’ are extracted for analysis.
3. Common typos and formatting issues are fixed.
4. Studio names are cleaned to ensure they are treated as a single token.
5. Assigned higher importance to genres by duplicating them in the feature string.
6. Created a single text feature by concatenating weighted genres and normalized studio names.

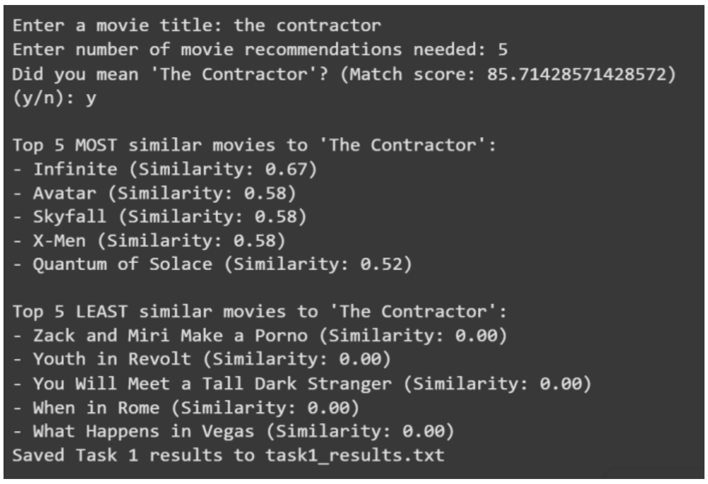
C. *Text Vectorisation:*

1. A custom tokenizer was implemented and passed to CountVectorizer to control how textual features are tokenized. The tokenizer simply splits the text based on whitespace (text.split()), which is effective for the pre-cleaned data where compound terms (e.g., romcom, sonypictures) are already formatted.
2. The CountVectorizer converts combined_features into a sparse matrix of token counts, enabling numerical representation of text.

D. *Cosine Similarity Computation and Recommendation Function:*

1. The cosine_similarity() function computes a matrix of pairwise similarity scores between all movies.
2. The user provides a movie name and the number of recommendations (N).
3. Using RapidFuzz, the closest matching title is identified.
4. Based on the cosine similarity scores, the Top N most and least similar movies are identified.
5. If multiple movies had identical similarity scores, they were shuffled to avoid bias and introduce variability in the results.

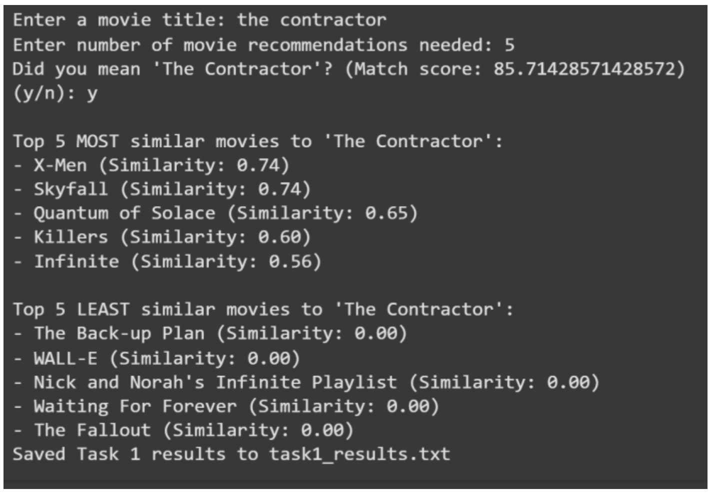
III. RESULTS



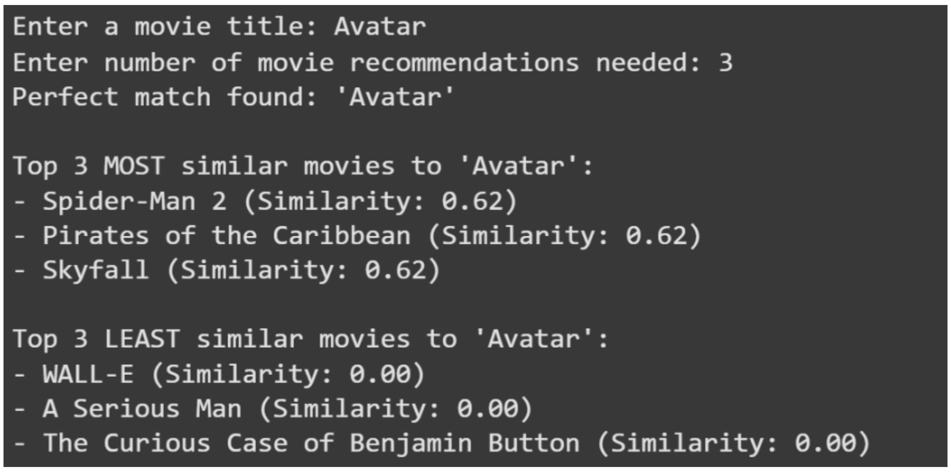
```
Enter a movie title: the contractor
Enter number of movie recommendations needed: 5
Did you mean 'The Contractor'? (Match score: 85.71428571428572)
(y/n): y

Top 5 MOST similar movies to 'The Contractor':
- Infinite (Similarity: 0.67)
- Avatar (Similarity: 0.58)
- Skyfall (Similarity: 0.58)
- X-Men (Similarity: 0.58)
- Quantum of Solace (Similarity: 0.52)

Top 5 LEAST similar movies to 'The Contractor':
- Zack and Miri Make a Porno (Similarity: 0.00)
- Youth in Revolt (Similarity: 0.00)
- You Will Meet a Tall Dark Stranger (Similarity: 0.00)
- When in Rome (Similarity: 0.00)
- What Happens in Vegas (Similarity: 0.00)
Saved Task 1 results to task1_results.txt
```

The initial results recommended the above results. The movie ‘Infinite’ was said to be the most similar to a movie called ‘The Contractor’. While the genres of ‘Infinite’ were ‘Science fiction’ and ‘Adventure’, the genres for ‘The Contractor’ was ‘Action’ and ‘Thriller’. The reason why it was the most similar was because the lead studio for both movies were ‘Paramount Pictures’.

As a lead studio house can make multiple movies with varied genres, there can be cases where this affects the count vectorizer and give not so similar movies as recommendations. To tackle this, we added more weight to genre of the movie than to lead studio (2:1). This led to better results as shown below.



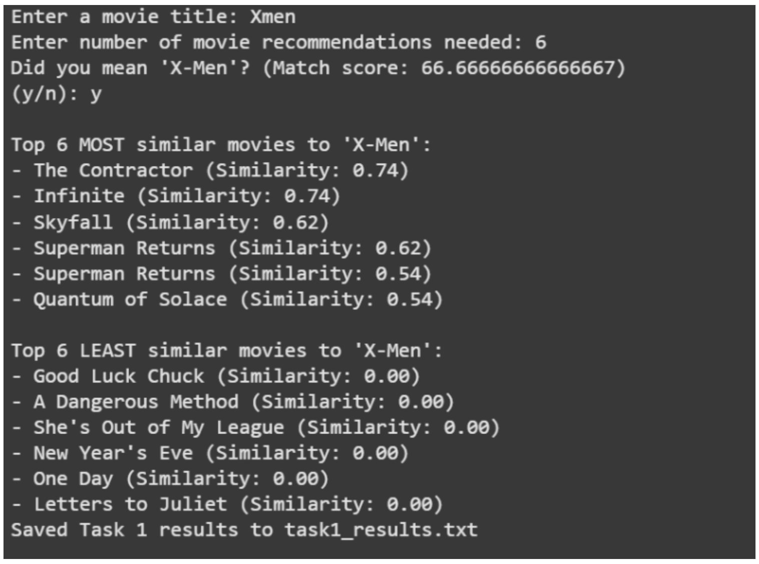
```
Enter a movie title: the contractor
Enter number of movie recommendations needed: 5
Did you mean 'The Contractor'? (Match score: 85.71428571428572)
(y/n): y

Top 5 MOST similar movies to 'The Contractor':
- X-Men (Similarity: 0.74)
- Skyfall (Similarity: 0.74)
- Quantum of Solace (Similarity: 0.65)
- Killers (Similarity: 0.60)
- Infinite (Similarity: 0.56)

Top 5 LEAST similar movies to 'The Contractor':
- The Back-up Plan (Similarity: 0.00)
- WALL-E (Similarity: 0.00)
- Nick and Norah's Infinite Playlist (Similarity: 0.00)
- Waiting For Forever (Similarity: 0.00)
- The Fallout (Similarity: 0.00)
Saved Task 1 results to task1_results.txt
```



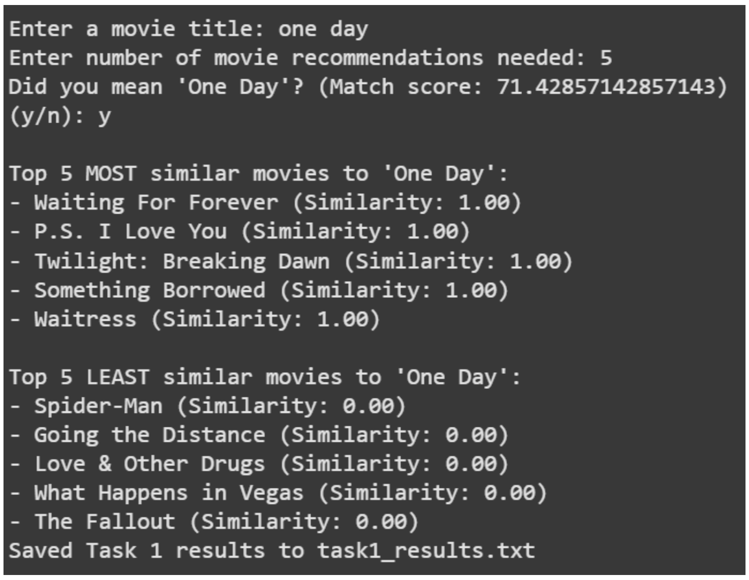
```
Enter a movie title: Avatar
Enter number of movie recommendations needed: 3
Perfect match found: 'Avatar'

Top 3 MOST similar movies to 'Avatar':
- Spider-Man 2 (Similarity: 0.62)
- Pirates of the Caribbean (Similarity: 0.62)
- Skyfall (Similarity: 0.62)

Top 3 LEAST similar movies to 'Avatar':
- WALL-E (Similarity: 0.00)
- A Serious Man (Similarity: 0.00)
- The Curious Case of Benjamin Button (Similarity: 0.00)
```

This is another example of the ideal usage of the code. Let's say the user inputs the movie 'Avatar', the movies 'Pirates of the Caribbean', 'Skyfall', and 'Spider-Man 2' are considered as the three most similar movies in the database. When we check the attributes of these films, it can be observed that Avatar belongs to the genres 'Action', 'Adventure' and 'Fiction' and comes from the studio 'Paramount Pictures'. All the three movies that were listed as 'Most Similar' to Avatar had the same two genres common with it – 'Action' and 'Adventure', with no common Studios for any of them. The three movies that were listed as "Least Similar" to Avatar have no genres or production studio in common with it and hence we can say that the code is working as intended and the results are accurate.

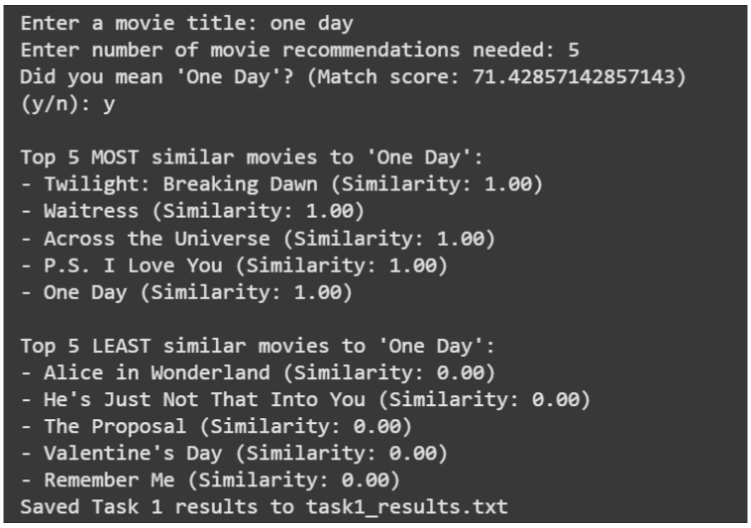
Some more test cases are shown below:



```
Enter a movie title: Xmen
Enter number of movie recommendations needed: 6
Did you mean 'X-Men'? (Match score: 66.66666666666667)
(y/n): y

Top 6 MOST similar movies to 'X-Men':
- The Contractor (Similarity: 0.74)
- Infinite (Similarity: 0.74)
- Skyfall (Similarity: 0.62)
- Superman Returns (Similarity: 0.62)
- Superman Returns (Similarity: 0.54)
- Quantum of Solace (Similarity: 0.54)

Top 6 LEAST similar movies to 'X-Men':
- Good Luck Chuck (Similarity: 0.00)
- A Dangerous Method (Similarity: 0.00)
- She's Out of My League (Similarity: 0.00)
- New Year's Eve (Similarity: 0.00)
- One Day (Similarity: 0.00)
- Letters to Juliet (Similarity: 0.00)
Saved Task 1 results to task1_results.txt
```



```
Enter a movie title: one day
Enter number of movie recommendations needed: 5
Did you mean 'One Day'? (Match score: 71.42857142857143)
(y/n): y

Top 5 MOST similar movies to 'One Day':
- Waiting For Forever (Similarity: 1.00)
- P.S. I Love You (Similarity: 1.00)
- Twilight: Breaking Dawn (Similarity: 1.00)
- Something Borrowed (Similarity: 1.00)
- Waitress (Similarity: 1.00)

Top 5 LEAST similar movies to 'One Day':
- Spider-Man (Similarity: 0.00)
- Going the Distance (Similarity: 0.00)
- Love & Other Drugs (Similarity: 0.00)
- What Happens in Vegas (Similarity: 0.00)
- The Fallout (Similarity: 0.00)
Saved Task 1 results to task1_results.txt
```



```
Enter a movie title: one day
Enter number of movie recommendations needed: 5
Did you mean 'One Day'? (Match score: 71.42857142857143)
(y/n): y

Top 5 MOST similar movies to 'One Day':
- Twilight: Breaking Dawn (Similarity: 1.00)
- Waitress (Similarity: 1.00)
- Across the Universe (Similarity: 1.00)
- P.S. I Love You (Similarity: 1.00)
- One Day (Similarity: 1.00)

Top 5 LEAST similar movies to 'One Day':
- Alice in Wonderland (Similarity: 0.00)
- He's Just Not That Into You (Similarity: 0.00)
- The Proposal (Similarity: 0.00)
- Valentine's Day (Similarity: 0.00)
- Remember Me (Similarity: 0.00)
Saved Task 1 results to task1_results.txt
```

TASK 2:

I. THEORY

A. Definition:

A supervised machine learning algorithm used for classification (or regression) that predicts a sample's label based on the majority vote (or average) of its *k* closest neighbours in feature space.

B. Mathematical Concept:

k-NN relies on a distance function (commonly Euclidean distance) to measure similarity between instances. The shorter the distance, the more similar the instances are considered.

C. Key Properties:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

1. Choosing k:
   - Small k: High variance, sensitive to noise.
   - Large k: High bias, oversmoothing.
   - Optimal k: Selected via cross-validation (as done in the project).
2. Simple to implement and interpret.
3. Adapts to new data without retraining.

II. METHODOLOGY

A. Libraries used:

1. pandas, numpy: For data preparation.
2. sklearn.model_selection.train_test_split: For splitting data.
3. sklearn.neighbors.KNeighborsClassifier: For classification.
4. sklearn.metrics: For accuracy and evaluation metrics.
5. matplotlib, seaborn: For visualization.

B. Data Preparation:

1. Each group member manually labelled the rating column (1 = like, 0 = dislike) based on their preference.
2. 10% of the movies are randomly removed to form the test set, simulating "newly released" movies.
3. The remaining 90% is used as training and validation data.
4. Irrelevant columns like identifiers, studio info, and auxiliary metrics are dropped.
5. Features are defined as all genre columns.
6. The target variable is the user rating column (rating_J or rating_H).

C. Model Training and Tuning:

1. The reduced dataset is split into train (80%) and validation (20%) sets.
2. This process is repeated three times with different random seeds to ensure robustness.

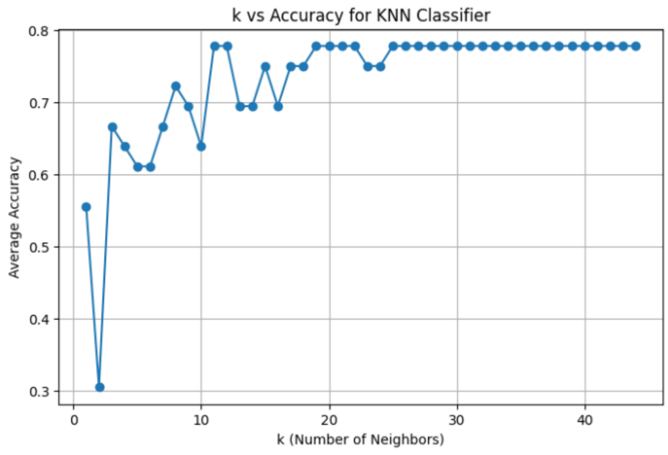
3. The value of k is iterated from 1 to 45 and the model is trained for each k. The accuracies are recorded and plotted to determine the optimal k value for the model.

D. Final Testing:

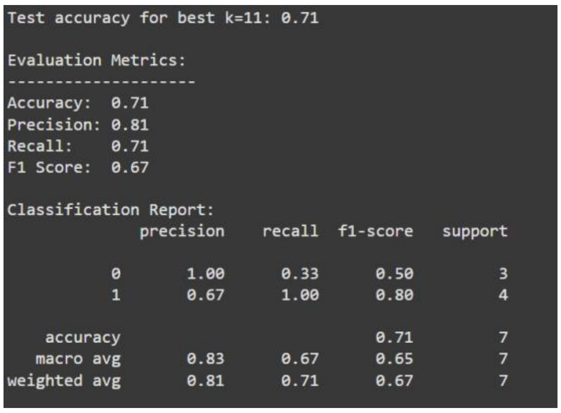
1. The model is retrained using the reduced data and the best performing value of k from the three runs.
2. Predictions are made on the unseen 10% test set.
3. Performance is measured using Accuracy, Precision, Recall, F1 Score and a Confusion Matrix.

III. RESULTS

After applying the KNN - classifier and running the model thrice to find the optimal k value for ratings given by Jaanvi (rating_J), we can clearly see when k=11 the Training accuracy reached the peak of about 78% The following is a graph for training accuracy (mean error of “predicted Y” and “rated Y”) vs k.



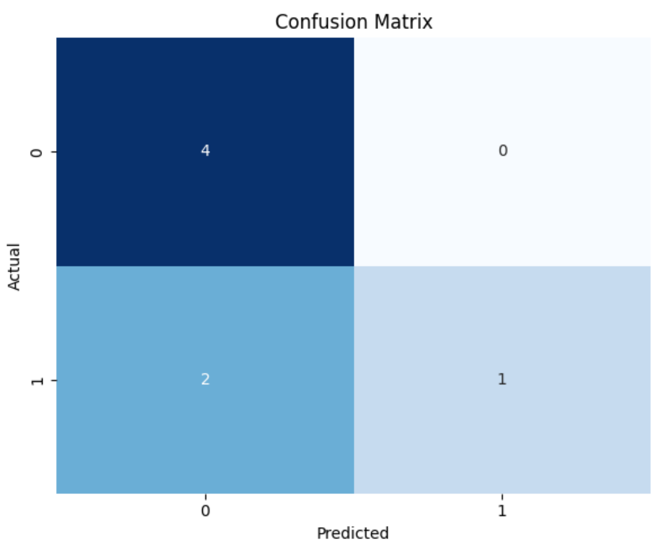
Using this value for testing gave the following results:



```
Test accuracy for best k=11: 0.71

Evaluation Metrics:
--------------------
Accuracy:  0.71
Precision: 0.81
Recall:    0.71
F1 Score:  0.67

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.67      1.00      0.80         4

    accuracy                           0.71         7
   macro avg       0.83      0.67      0.65         7
weighted avg       0.81      0.71      0.67         7
```

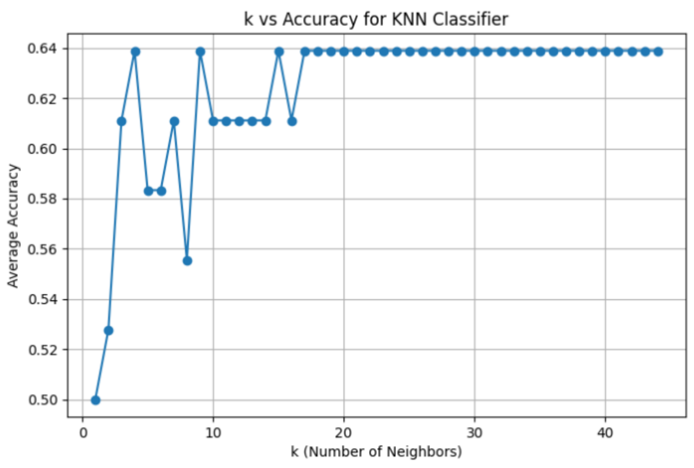
Using this value of k on the newly released movies gave the following results:

It predicted 4 movies which were disliked by Jaanvi correctly and 1 movie which was liked by her correctly but it predicted 2 movies which were liked by Jaanvi incorrectly as disliked.

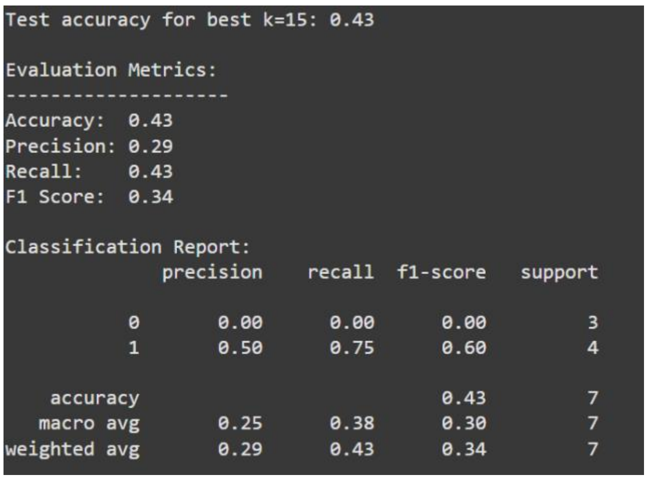


Next we tested the same test to find the optimal k value for ratings given by Henil (rating_H). The results were very underwhelming.

The optimal k value after 3 runs was k=4 with only 64% training accuracy. The following is a graph for training accuracy (mean error of "predicted Y" and "rated Y") vs k.



Using this value for testing gave the following results:

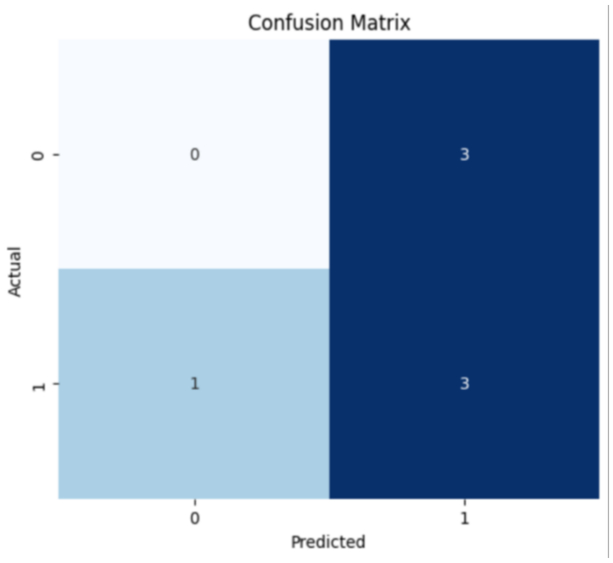


```
Test accuracy for best k=15: 0.43

Evaluation Metrics:
--------------------
Accuracy:  0.43
Precision: 0.29
Recall:    0.43
F1 Score:  0.34

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.50      0.75      0.60         4

    accuracy                           0.43         7
   macro avg       0.25      0.38      0.30         7
weighted avg       0.29      0.43      0.34         7
```

The results showed us a problem, there were not enough data points with disliked (0) label. This gave very biased results for testing.

Using this value of k on the newly released movies gave the following results:

It predicted 3 movies which were liked by Henil correctly and it predicted 3 movies which were disliked by him incorrectly as liked, it also predicted 1 movie as disliked which actually was liked.



Conclusion

Hence, we fulfil our task to make recommender engine using two different methods: Count Vectorizer and cosine similarity and k-Nearest Neighbours (k-NN) classifier.

The cosine similarity model proved effective in identifying similar movies based on textual attributes like genre and studio, especially after adjusting the weight of genre data to improve recommendation quality. Meanwhile, the k-NN classifier demonstrated how personalized recommendations can be derived from user-labelled preferences. Although performance varied depending on the amount and distribution of user data, the method highlighted the potential of supervised learning in recommender systems.

Overall, both approaches illustrated key principles behind modern recommender engines. The exercise emphasized the importance of data cleaning, feature engineering, model tuning, and evaluation metrics in building practical AI systems. This lab serves as a foundational step toward developing more advanced and scalable recommendation solutions in real-world scenarios.

Reflection

This lab was a great first dive into building recommender systems. Implementing cosine similarity and k-NN gave us hands-on experience with core recommendation algorithms. While challenging at times—especially cleaning messy genre data and tuning k-NN parameters—it was rewarding to see the systems work.

The real-world relevance stood out. Seeing how these techniques power actual platforms made the concepts click. Visualizing results like similarity scores and accuracy curves helped too.

For next time, a larger dataset and more features (like actors or keywords) could make recommendations even better. Adding a simple interface would also improve testing.

Overall, this was a solid introduction that balanced theory with practice. It showed how much work goes into recommendations—and how satisfying it is when they work well. With some tweaks, this lab could be even stronger for future students.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[2] https://naomy-gomes.medium.com/the-cosine-similarity-and-its-use-in-recommendation-systems-cb2ebd811ce1
[3] https://www.youtube.com/watch?v=Gy8t5XIrISc
[4] https://www.datacamp.com/blog/what-is-tokenization
[5] https://github.com/naomyduarteg/LIA-FastAPI-SQL/blob/main/data/data.csv
[6] https://www.mdpi.com/1424-8220/22/13/4904
[7] https://www.w3schools.com/python/ref_string_strip.asp
[8] https://www.youtube.com/watch?v=e9U0QAFbfLI