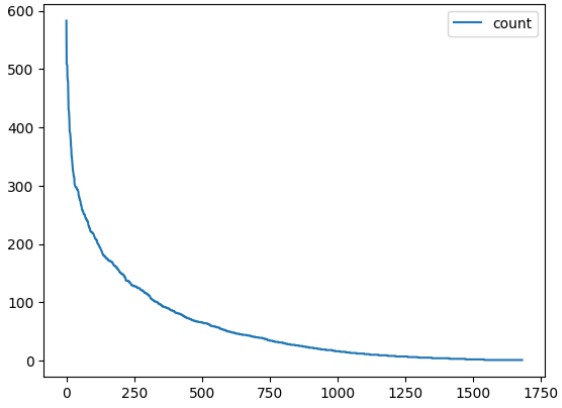
Assignment 2

Introduction

In this assignment I tried to make a movie recommendation system. To do so I ran the LightGCN and NGCF models from labs to get reference performance. After that I built a KNN algorithm to perform collaborative filtering.

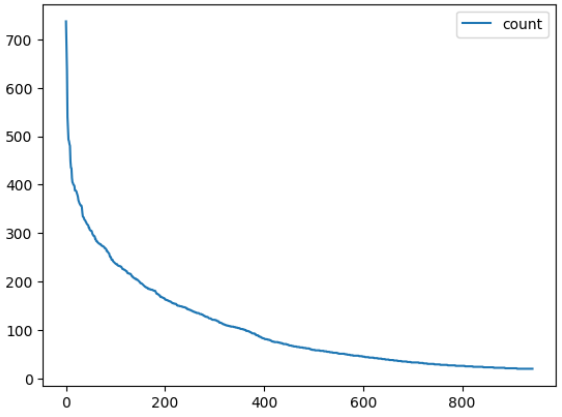
Data analysis

Review number



Number of reviews per film

max reviews: [583]
min reviews: [1]

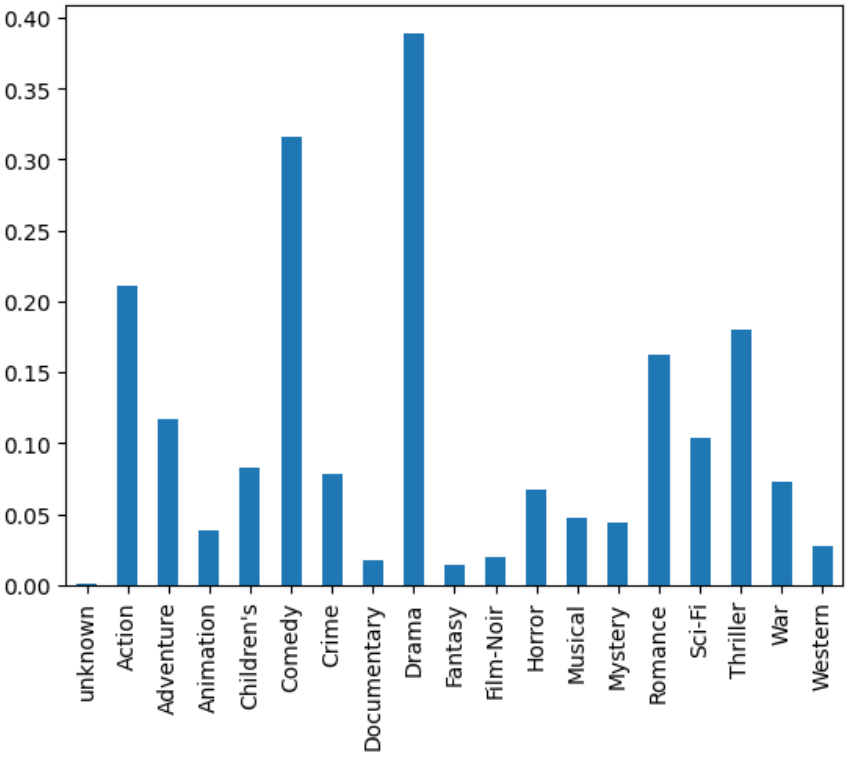


Number of films reviewed per user

max reviews: [737]
min reviews: [20]

Genre distribution

The most common genre is drama, taking almost 40% of the reviews!



Model Implementation

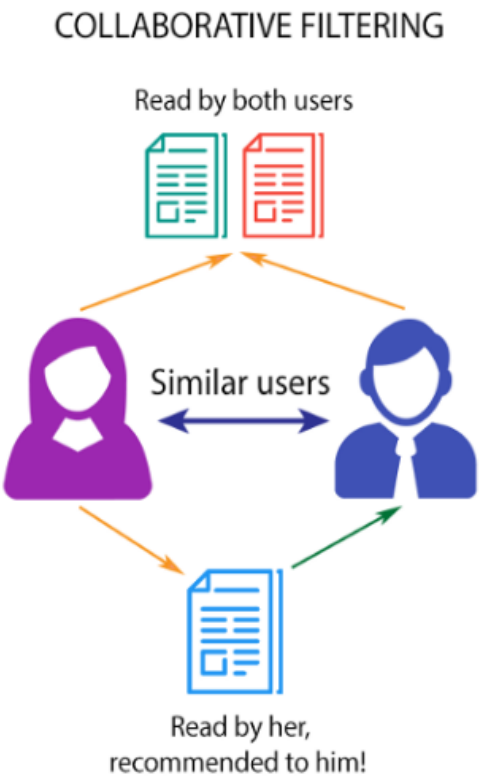
There are three main types of recommender systems:

1. **Collaborative recommender system**
2. **Content based recommender system**
3. **Knowledge based recommender systems**

There are also hybrid implementations.

Item based Collaborative Recommended System using KNN

Using KNN we can determine a similar movie or a user based on a cosine similarity, yielding K closest neighbors.




```
Top 5 users who are very much similar to the User- 510 are:

1 . User: 525 separated by distance of 0.4848303019016639
2 . User: 821 separated by distance of 0.6466306053203914
3 . User: 801 separated by distance of 0.666862674201284
4 . User: 741 separated by distance of 0.6787580098588736
5 . User: 402 separated by distance of 0.6823285747225906
```

Using the distance from each user we understand how similar those users are and can calculate weighted sum of rating for the films similar K users reviewed to propose them for our initial user.

Model Advantages and Disadvantages

Advantages:

1. **Simplicity:** KNN is a simple and intuitive algorithm. It is easy to understand and implement, making

Disadvantages:

1. **Computational Complexity:** As the dataset grows, the computational cost of finding the nearest neighbors increases.

it a good choice for beginners or for quick prototyping.

2. **No Training Phase:** KNN is a lazy learner, meaning it doesn't require a training phase. The model is built at the time of prediction, which can be advantageous when dealing with dynamic datasets that change frequently.
3. **Effectiveness with Sparse Data:** KNN performs well even when dealing with sparse data, where many users have rated only a small subset of items. This makes it suitable for real-world recommendation scenarios where users typically rate only a few items.
4. **User-Item Flexibility:** KNN can be used for both user-based and item-based collaborative filtering. In the case of item-based recommendation, it is particularly flexible and well-suited for handling large item datasets.

Calculating distances between items for every prediction can become computationally expensive, especially for large datasets.

2. **Scalability:** KNN may face challenges in terms of scalability when dealing with a large number of items or users. The search for neighbors becomes more time-consuming as the dataset size grows.
3. **Cold Start Problem:** KNN may struggle with the "cold start" problem, which occurs when there are new items in the system that have not yet received enough ratings to establish meaningful similarities. Similarly, it may face challenges when new users join the system.
4. **Uniform Sensitivity to Ratings:** KNN treats all user ratings equally, which might not be suitable for cases where certain users' opinions are more influential or reliable than others. More advanced models may take into account user preferences and confidence levels.
5. **Need for Proper Similarity Metric:** The performance of KNN is highly dependent on the choice of a similarity metric. Selecting an appropriate metric for a specific domain or dataset is crucial for achieving accurate recommendations.

6. **Data Sparsity Issues:** In highly sparse datasets, finding similar items based on user interactions can be challenging. Some items may have very few or no common users, leading to less reliable similarity measures.

Training Process

There were no training for KNN. The only important step — is converting training dataset into scipy's Compressed Sparse Row matrix to reduce memory size and computation time.

Evaluation

LightGCN and NGCF model came with built-in recall/precision evaluation. Since test set is quite sparse, the most important metric here is recall. While precision tends to be quite low due to the sparsity of the dataset.

In the meantime I did not get how to calculate accuracy and precision for evaluation of other algorithms.

Results

The KNN algorithm I have implemented does indeed recommends movies using **item-based Collaborative recommender system**

```
print("Movies recommended based on similar users are: ")
pprint(movies_list[recommend_movies(510, 5, knn_size=5, verbose=False)])
```

✓ 0.0s

```
Movies recommended based on similar users are:
153    Monty Python's Life of Brian (1979)
160                         Top Gun (1986)
20            Muppet Treasure Island (1996)
285             English Patient, The (1996)
88                      Blade Runner (1982)
Name: movie title, dtype: object
```