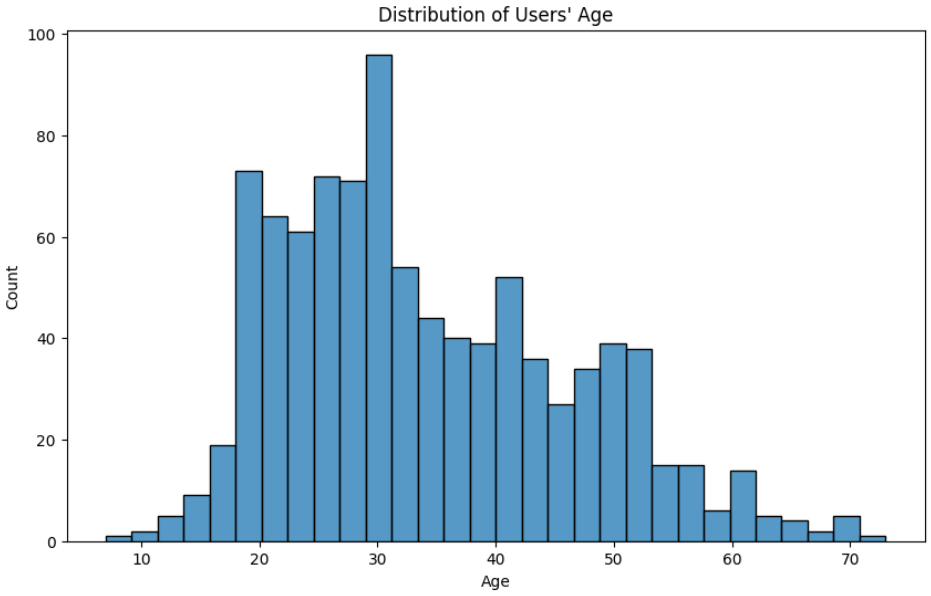
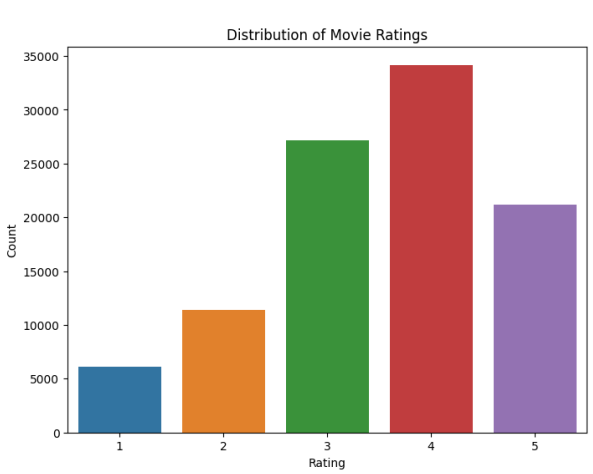
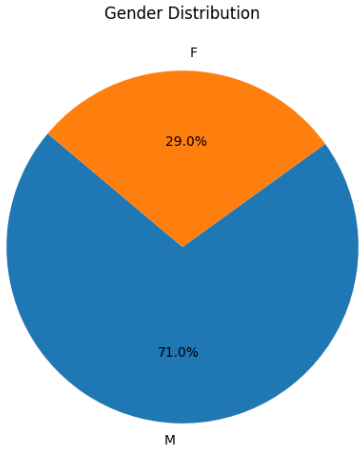
Data analysis:

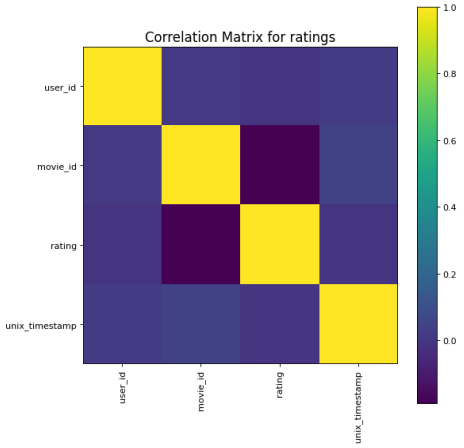






After conducting an in-depth analysis of the dataset, I have come to several noteworthy observations. Firstly, it has become evident that I am unable to provide recommendations for movies intended for two specific age groups: children under the age of 7 and elderly individuals over the age of 73. The reason for this limitation is the absence of any data pertaining to these particular age demographics within the dataset. This absence of relevant information makes it impractical to make tailored movie recommendations for these age brackets.

I also observed a notable absence of the "video_release_date" column in the dataset. This absence implies that there is no information available regarding the release dates of movies in the dataset. This could potentially impact the ability to perform certain types of analyses or make temporal recommendations based on release dates, however, I noticed the movie release year is included in the title.



Furthermore, when conducting a correlation analysis on the rating dataset, I found that there is a lack of significant correlations between the various columns. This implies that there are no strong linear relationships or dependencies between the different attributes or features included in the dataset.

Model Implementation:

Method1- -Simple-Collaborative-Filtering:

The system makes recommendations for products based on its search for comparable users or films. For example, if two people both enjoy the same movies—A, B, and D—it is likely that the first will purchase item D and the second will purchase item C because of their many similarities.

Since it suggests that they reside in closer neighborhoods, similarity-based approaches identify the objects that are the most similar and have the greatest values. There are several measures based on similarity. However, since our matrix is sparse—that is, the majority of the movies have ratings of 0—we will be utilizing Pearson's correlation in this instance. In order to make 0 the default rating, we will therefore center all of the ratings at 0.

Here, the user similarity and the movie rating are multiplied to create a new (score). The fundamental notion is that the user who is most like you will be more likely to recommend films that are better. As a result, a film that has received a high rating from the user who is most like it will rank higher in our recommendation.

Model Advantages and Disadvantages:

advantages:

1. his approach can handle the "cold start" problem, which occurs when there is limited or no historical data for a new user or item. By finding similar users or items, the system can still provide recommendations, even for newcomers.
2. The use of Pearson's correlation can be advantageous in situations where most of the data is missing or incomplete.

Disadvantages:

3. Since this approach primarily relies on finding users with similar tastes, it may not introduce users to entirely new or unexpected items. It can sometimes lead to "filter bubbles" where users are recommended items that are very similar to what they have already consumed.
4. While it can handle the cold start problem for new users, it may still face challenges when new items are introduced to the system, as there may not be enough data or similar items to make recommendations.

Method2- -Collaborative-Filtering-Recommendation-System-in-Surprise-SVD-method:

Surprise is a very valuable tool that can be used within Python to build recommendation systems. Its documentation is quite useful and explains its various prediction algorithms' packages. Before we start building a model, it is important to import elements of surprise that are useful for analysis, such as certain model types (SVD, KNNBasic, KNNBaseline, KNNWithMeans, and many more), Dataset and Reader objects (more on this later), accuracy scoring, and built-in train-test-split, cross-validation, and GridSearch.

The SVD model is one of the strongest recommendation systems available. SVD is a type of matrix factorization that minimizes the error between the expected and actual ratings from our original utility matrix by using gradient descent to provide predictions for users' ratings. Gradient descent therefore minimizes RMSE in the process of forecasting these new ratings.

Training Process:

Model1-training

1. Merge movies with Ratings
2. Calculate user_movie_matrix
3. Calculate similarity matrix

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

Model2-training

1. Merge movies with Ratings
2. Since out movies rating is on scale from 1 to 5 we put the rating scale (1,5) in the reader
3. Split the dataset to train and test

4. We trained at first on one SVD(n_factors=200) the result RMSE: 0.9412
5. We trained again on This parameters = {'n_factors':[20,50,80],
'reg_all':[0.04,0.06],
'n_epochs':[10,20,30],
'Lr_all':[0.002,0.005,0.01]}
GridSearchCV(SVD,param_grid=parameters,n_jobs=-1)
The result is RMSE: 0.91
With best parameters {'n_factors': 50, 'reg_all': 0.06, 'n_epochs': 20, 'lr_all': 0.01}

Evaluation:

Benchmark: Mean square Error between predicted rank and true rank

For the First Method, I predicted the ranking using the most similar users and took their ranking to the movies, the average MSE score after training and tested u1.base, u1.test, u2.base, u2.test, u3.base, u3.test, u4.base, u3.test, u5.base, u5.test separately is ~1.7
Mean Squared Error: 1.79995 of dataset 1
Mean Squared Error: 1.79115 of dataset 2
Average Mean Squered Error: 1.79555 of all datasets

For The second Method, I used RMSE score that is included in the surprise library, I training and tested on the same above datasets the result as following:
dataset1 RMSE = 0.9566
dataset2 RMSE = 0.9418
dataset3 RMSE = 0.9364
dataset4 RMSE = 0.9387
dataset5 RMSE = 0.9385
AVG = 0.9424

Results:

The best Average MSE score is 0.9424 from surprise SVD

To conclude I learned that research and experiments are good for improving skills, but also do not work hard but work smart and use surprise.