For this part of assignment, let us use the module 3 movie ratings data and attempt to predict missing ratings using non-negative matrix factorization.

**Section 1**

In [16]:
```
import pandas as pd
pd.set_option('display.max_columns', None)

# import data and display columns
data_users = pd.read_csv('users.csv')
data_movies = pd.read_csv('movies.csv')
data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
print(data_users.head())
print(data_movies.head())
print(data_train.head())
print(data_test.head())
```

```
   uID gender  age  accupation    zip
0    1      F    1          10  48067
1    2      M   56          16  70072
2    3      M   25          15  55117
3    4      M   45           7  02460
4    5      M   25          20  55455
   mID                         title  year  Doc  Com  Hor  Adv  Wes  Dra  Ani  \
0    1                     Toy Story  1995    0    1    0    0    0    0    1
1    2                       Jumanji  1995    0    0    0    1    0    0    0
2    3              Grumpier Old Men  1995    0    1    0    0    0    0    0
3    4             Waiting to Exhale  1995    0    1    0    0    0    1    0
4    5  Father of the Bride Part II  1995    0    1    0    0    0    0    0

   War  Chi  Cri  Thr  Sci  Mys  Rom  Fil  Fan  Act  Mus
0    0    1    0    0    0    0    0    0    0    0    0
1    0    1    0    0    0    0    0    0    1    0    0
2    0    0    0    0    0    0    1    0    0    0    0
3    0    0    0    0    0    0    0    0    0    0    0
4    0    0    0    0    0    0    0    0    0    0    0
    uID   mID  rating
0   744  1210       5
1  3040  1584       4
2  1451  1293       5
3  5455  3176       2
4  2507  3074       5
    uID   mID  rating
0  2233   440       4
1  4274   587       5
2  2498   454       3
3  2868  2336       5
4  1636  2686       5
```

In [17]:
```
import numpy as np
from sklearn.decomposition import NMF
from sklearn.metrics import mean_squared_error

# create user-movie matrix:
# rows: users
# cols: movies
# vals: ratings
mat_user_movie = data_train.pivot(index='uID', columns='mID', values='rating').fillna(0)

# use NMF
mod_nmf = NMF(n_components=15, max_iter=1500, random_state=42)
feat_user = mod_nmf.fit_transform(mat_user_movie)
feat_movie = mod_nmf.components_

# create predicted ratings matrix
mat_predict_ratings = np.dot(feat_user, feat_movie)

# get predicted ratings for test data
# check if movie ID exists in training data and default to 0 if it doesn't
test_uIDs = data_test['uID']
test_mIDs = data_test['mID']
train_uIDs = mat_user_movie.index
train_mIDs = mat_user_movie.columns
test_ratings = data_test['rating']
test_predicts = []
for uID, mID in zip(test_uIDs, test_mIDs):
    if (uID in train_uIDs) and (mID in train_mIDs):
        uID = train_uIDs.get_loc(uID)
        mID = train_mIDs.get_loc(mID)
        test_predicts.append(mat_predict_ratings[uID, mID])
```

```
    else:
        test_predicts.append(0)

# get RMSE
print(f'RMSE: {np.sqrt(mean_squared_error(test_ratings, test_predicts))}')
```

```
RMSE: 2.8724517237578873
```

**Section 2**

So, it seems we ended up with a rather high RMSE of ~2.87. This is worse than what the simpler baseline and similarity-based methods yielded and shows that NMF might not be a good choice for this kind of problem. In fact, there are several reasons why this might be the case.

To start with, NMF requires a rather dense matrix to be accurate. After all, it works by learning latent factors. Our matrices though, were quite sparse. This also reflects the real world where people do not usually go through a long list of movies, leaving ratings or feedback for each. Furthermore, NMF will always struggle with users/movies that have few ratings, which is very common with new users and movies. NMF is also highly susceptible to overfitting—it might try to learn too many latent factors from the training data and fail to apply that knowledge to the test data.

Some ways to potentially improve an NMF-based predictor is to leverage regularization. This might help with overfitting and allow for better generalization when working with test data. Furthermore, we could incorporate some similarity-based methods into our NMF model, creating a hybrid system. Such a system would likely use NMF to extract the latent factors but then rely on a collaborative approach for final predictions. Another way of helping NMF perform better is to give it more data in addition to ratings, such as whether a user finished the movie or how many times they watched it.

In [ ]: