**Three Types of Recommendations**

There are three methods that you have now implemented for making recommendations. These are the three most recognized methods in industry:

1. Knowledge Based Recommendations

Knowledge based recommendations frequently are implemented using filters, and are extremely common amongst luxury based goods. Filters that you might see when purchasing items like cars or homes are examples of knowledge based recommendations. In knowledge based recommendations, users provide information about the types of recommendations they would like back.

2. Collaborative Filtering Based Recommendations

Collaborative filtering uses the connections between users and items to make recommendations. Even the content based recommendation you just implemented used some collaborative filtering techniques, as you were not treating items and users independent from one another. In this lesson, you used neighborhood based collaborative filtering to find users who were alike and then recommend new movies based on these similar users.

Even in the content based recommendation, you were using collaborative filtering. You were finding items that were similar and making recommendations of new items based on the highest ratings of a user. Because you were still using the user ratings of an item, this was an example of a blend between content and collaborative filtering based techniques.

3. Content Based Recommendations

In the previous notebook, you created a matrix of similarities between items (movies) based only on the content related to those movies (year and genre). The similarity matrix that was used, was completely created using only the items (movies). There was no information used about the users implemented. For any movie, you would be able to determine the most related additional movies based only on the genre and the year of the movie. This is the premise of how a completely content based recommendation would be made.

Often blended techniques of all three types are used in practice to provide the the best recommendation for a particular circumstance.

### Different Scales

If you are in control of choosing your rating scale, think of what might be most beneficial to your scenario. If you are working alongside a team TO design the interfaces for how data will be collected, there are number of ideas to keep in mind.

* Do you need to ask questions of your user or can you collect data about their interactions with items?
* If you need to ask questions, how many do you ask?
* How do you word the questions?
* And finally, the question in the above video: what type of scale should you use?

In general, I suggest using the simplest rating that allows you to get whatever questions of interest you have, but there are some important ideas to keep in mind when choosing a particular type of rating. Ratings are a necessary part of working with different recommendation systems, but they aren't a central part of our focus. A good overview of types of ratings and when to use them is also provided [here](https://conversionxl.com/blog/survey-response-scales/).

Most of these ideas are specific to your use case, and are easy to notice in hindsight. It is simply important to think of this in advance and not completely gloss over possible issues with the data you are collecting and how it connects to the questions you want answered.

### Training and Testing Data For Recommendations

In the last lesson, you were making recommendations by providing a list of popular items, or a list of items that the user hadn't observed but that someone with similar tastes had observed. However, understanding if these recommendations are good in practice means that you have to deploy these recommendations to users and see how it impacts your metrics (sales, higher engagement, clicks, conversions, etc.).

You may not want your recommendations to go live to understand how well they work. In these cases, you will want to split your data into training and testing portions. In these cases, you can train your recommendation engine on a subset of the data, then you can test how well your recommendation engine performs on a test set of data before deploying your model to the world.

However, the cases you saw in the last lesson, where just a list of recommendations was provided, don't actually lend themselves very well to training and testing methods of evaluation. In the next upcoming pages, you will be introduced to matrix factorization, which actually does work quite well for these situations.

## Validating Your Recommendations

### Online Testing

For online methods of testing a recommender's performance, many of the methods you saw in the previous lesson work very well - you can deploy your recommendations and just watch your metrics carefully. It is common in practice to set up online recommendations to have an "old" version of recommended items, which is compared to a new page that uses a new recommendation strategy.

All ideas associated with A/B testing that you learned in the earlier lessons are critical to watching your metrics in online learning, and ultimately, choosing a recommendation strategy that works best for your products and customers.

### Offline Testing

In many cases, a company might not let you simply deploy your recommendations out into the real world any time you feel like it. Testing out your recommendations in a training-testing environment prior to deploying them is called **offline** testing.

The recommendation methods you built in the previous lesson actually don't work very well for offline testing. In offline testing, it is ideal to not just obtain a list of recommendations for each individual, because we ultimately don't know if a user doesn't use an item because they don't like it, or because they just haven't used it yet (but would like it). Rather, it would be great if we have an idea of how much each user would like each item using a predicted rating. Then we can compare this predicted rating to the actual rating any individual gives to an item in the future.

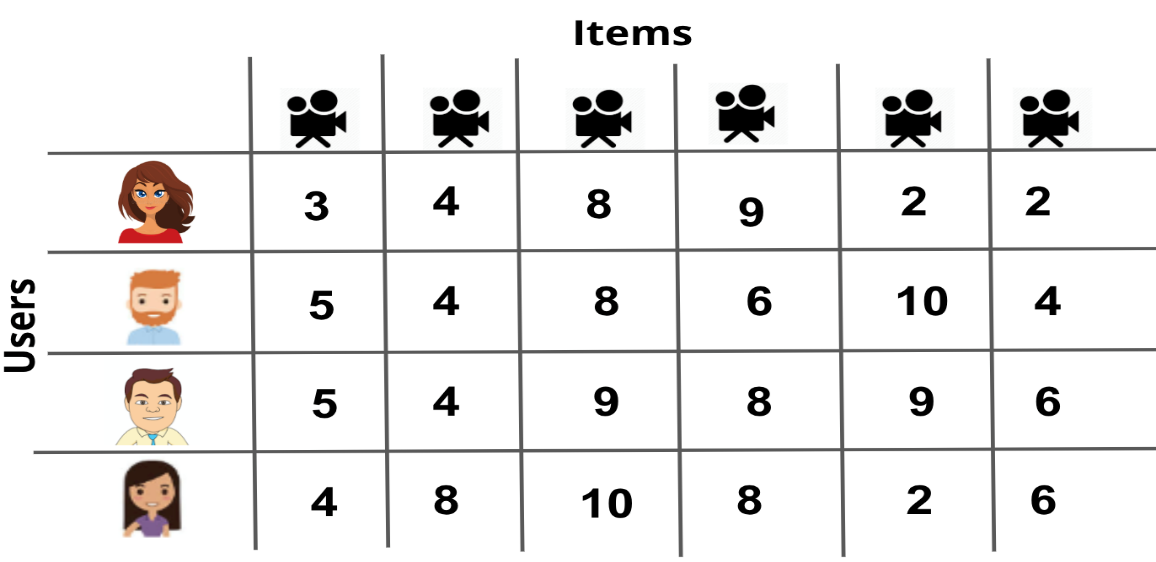
In the previous video, you saw an example of a user to whom we gave a list of movies that they still hadn't seen. Therefore, we couldn't tell how well we were doing with our recommendations. Techniques related to matrix factorization lend themselves nicely to solving this problem.

### User Groups

The final (possible) method of validating your recommendations is by having user groups give feedback on items you would recommend for them. Obtaining good user groups that are representative of your customers can be a challenge on its own. This is especially true when you have a lot of products and a very large consumer base.

### Latent Factors

When performing SVD, we create a matrix of users by items (or customers by movies in our specific example), with user ratings for each item scattered throughout the matrix. An example is shown in the image below.

[[](https://classroom.udacity.com/nanodegrees/nd025-ent/parts/347da0f8-5670-4bbf-8587-b02b7c0fe111/modules/db86b648-24c4-4589-8755-458885e0503c/lessons/012dcc67-eace-483d-99ff-8a580aa39050/concepts/aeab8348-cf4d-4749-a469-a95ee6d81397)](https://classroom.udacity.com/nanodegrees/nd025-ent/parts/347da0f8-5670-4bbf-8587-b02b7c0fe111/modules/db86b648-24c4-4589-8755-458885e0503c/lessons/012dcc67-eace-483d-99ff-8a580aa39050/concepts/aeab8348-cf4d-4749-a469-a95ee6d81397)

You can see that this matrix doesn't have any specific information about the users or items. Rather, it just holds the ratings that each user gave to each item. Using SVD on this matrix, we can find **latent features** related to the movies and customers. This is amazing because the dataset doesn't contain any information about the customers or movies!