**Overview**

In this project, I will build an item-based collaborative filtering system using MovieLens Datasets. Specially, I will train a KNN models to cluster similar movies based on user's ratings and make movie recommendation based on similarity score of previous rated movies.

# [Recommender system](https://en.wikipedia.org/wiki/Recommender_system)

A recommendation system is basically an information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. It is widely used in different internet / online business such as Amazon, Netflix, Spotify, or social media like Facebook and YouTube. By using recommender systems, those companies are able to provide better or more suited products/services/contents that are personalized to a user based on his/her historical consumer behaviours.

Recommender systems typically produce a list of recommendations through collaborative filtering or through content-based filtering

This project will focus on collaborative filtering and use item-based collaborative filtering systems make movie recommendation

# Item-based Collaborative Filtering

Collaborative filtering-based systems use the actions of users to recommend other items. In general, they can either be user based or item based. User based collaborating filtering uses the patterns of users similar to me to recommend a product (users like me also looked at these other items). Item based collaborative filtering uses the patterns of users who browsed the same item as me to recommend me a product (users who looked at my item also looked at these other items).

Item-based approach is usually prefered than user-based approach. User-based approach is often harder to scale because of the dynamic nature of users, whereas items usually don't change much, so item-based approach often can be computed offline.

**Data Sets**

I use MovieLens Datasets. This dataset (ml-latest.zip) describes 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org/), a movie recommendation service. It contains 100836 ratings across 9742 movies. These data were created by 610 unique users.

Users were selected at random for inclusion. All selected users had rated at least 1 movie. No demographic information is included. Each user is represented by an id, and no other information is provided.

The data are contained in the files  movies.csv and ratings.csv.

## Project Content

Part A: Data Pre-processing

1. Import the libraries
2. Importing the dataset
3. Visualising the dataset
4. Filtering the dataset

Part B: Building the KNN model for collaborative filtering

1. Reshaping the data
2. Fitting the model
3. Testing the model

Part C: Calculate how spare is the movie-user matrix

**From Visualisation of dataset:**

We get,

1. The distribution of ratings among movies often satisfies a property in real-world settings, which is referred to as the long-tail property.
2. According to this property, only a small fraction of the items are rated frequently. Such items are referred to as popular items. The vast majority of items are rated rarely. This results in a highly skewed distribution of the underlying ratings.
3. We can see that roughly 2,000 out of 9,742 movies are rated more than 10 times. More interestingly, roughly 4,000 out of 9,742 movies are rated less than only 2 times.
4. So about 1% of movies have roughly 329 or more ratings, 5% have 47 or more, and 20% have 20 or more. Since we have so many movies, we'll limit it to the top 40%.
5. This is arbitrary threshold for popularity, but it gives us about 41,300 different movies. We still have pretty good amount of movies for modeling. There are two reasons why we want to filter to roughly 41,000 movies in our dataset.

* Memory issue: we don't want to run into the “MemoryError” during model training
* Improve KNN performance: lesser-known movies have ratings from fewer viewers, making the pattern noisier. Dropping out less known movies can improve recommendation quality.

1. We can see that the distribution of ratings by users is very similar to the distribution of ratings among movies. They both have long-tail property.
2. Only a very small fraction of users are very actively engaged with rating movies that they watched. Vast majority of users aren't interested in rating movies. So we can limit users to the top 40%, which is about 41,320 users.

**Training the model includes:**

* Reshaping the Data
* Fitting the Model

#### 1. Reshaping the Data

For K-Nearest Neighbors, we want the data to be in an (artist, user) array, where each row is a movie and each column is a different user. To reshape the dataframe, we'll pivot the dataframe to the wide format with movies as rows and users as columns. Then we'll fill the missing observations with 0s since we're going to be performing linear algebra operations (calculating distances between vectors). Finally, we transform the values of the dataframe into a scipy sparse matrix for more efficient calculations.

#### 2. Fitting the Model

Time to implement the model. We'll initialize the NearestNeighbors class as model\_knn and fit our sparse matrix to the instance. By specifying the metric = cosine, the model will measure similarity between artist vectors by using cosine similarity.

This is very interesting that my **KNN** model recommends movies that were also produced in very similar years. However, the cosine distance of all those recommendations are actually quite small.

This is probably because there is too many zero values in our movie-user matrix. With too many zero values in our data, the data sparsity becomes a real issue for **KNN** model.

There is about 72.64% of ratings in our data is missing

This result confirms my hypothesis. The vast majority of entries in our data is zero.

## The bottleneck of item-based collaborative filtering.

* cold start problem
* data sparsity problem
* popular bias (how to recommend products from the tail of product distribution)
* scalability bottleneck

We saw there is 72.64% of user-movie interactions are not yet recorded, even after I filtered out less-known movies and inactive users.

Apparently, we don't even have sufficient information for the system to make reliable inferences for users or items. This is called **Cold Start** problem in recommender system.

There are three cases of cold start:

1. New community: refers to the start-up of the recommender, when, although a catalogue of items might exist, almost no users are present and the lack of user interaction makes very hard to provide reliable recommendations
2. New item: a new item is added to the system, it might have some content information but no interactions are present
3. New user: a new user registers and has not provided any interaction yet, therefore it is not possible to provide personalized recommendations

We are not concerned with the last one because we can use item-based filtering to make recommendations for new user. In our case, we are more concerned with the first two cases, especially the second case.

This constitutes a **Data Sparsity** problem mainly for collaborative filtering algorithms due to the fact that they rely on the item's interactions to make recommendations. If no interactions are available then a pure collaborative algorithm cannot recommend the item. In case only a few interactions are available, although a collaborative algorithm will be able to recommend it, the quality of those recommendations will be poor.

This arises another issue, **Popular Bias** which is not anymore related to new items, but rather to unpopular items. In some cases like in this project of movie recommendations, it might happen that a handful of items receive an extremely high number of interactions, while most of the items only receive a fraction of them. This is also referred to as popularity bias.

In addition to that, **Scalability** is also a big issue in KNN model too. Its time complexity is

O(nd + kn), where n is the cardinality of the training set and d the dimension of each sample. And KNN takes more time in making inference than training, which increase the prediction latency