**Understanding Parameters**

We start off by understanding the required fields to build a recommendation system. The required fields would be movieId , userId and rating. Since the database contains extra fields that are non-related to the recommendation system, we remove these fields and refine the received data.

**The approach**

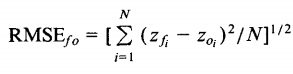
Machine learning algorithms in recommendation systems are typically classified into two categories — content based and collaborative filtering methods although modern recommenders combine both approaches. Content based methods are based on similarity of item attributes and collaborative methods calculate similarity from interactions. From our requirement we chose collaborative method as the best approach. Collaborative methods work with the interaction matrix that can also be called rating matrix in our case since we are developing our first algorithm based on the ratings that are given by the user. The task of machine learning is to learn a function that predicts utility of items to each user.

**Selection of Tools**

For conversion of the json documents received from database into mathematical data frames we chose pandas. Pandas data frames are , arguably, the best way to easily carry out necessary refinement such as dropping columns, changing the datatype of values of particular columns etc. There are a lot of machine learning libraries available in python, and there are several recommendation system based libraries as well. We chose scikit-surprise as the best pick because of the simplicity, diversity and efficiency of the library.

**Algorithm**

After refining data , we start by choosing the best algorithm that is suitable for our dataset. We ran the GridSearchCV method of surprise to compare different algorithms with a suitable accuracy measure. The accuracy measurement we are using here is RMSE - Root Mean Squared Error. RMSE is a measure of how spread out residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Residuals are a measure of how far from the regression line data points are. The formula is:



Where:

Σ = summation (“add up”)

(zfi – Zoi)2 = differences, squared

N = sample size.

The analysis lead to two best suited algorithms namely, K-Nearest Neighbour (K-NN ) and Matrix Factorization using Single Value Decomposition ( SVD ). Both these have an RMSE ranging from 0.8 to 0.9, which is very impressive. Hence we consider the fit time of dataset which resulted in 32.0 for SVD and 0.07 for K-NN. Hence for better performance we chose K-NN.

The software creates a machine learning model using K-NN. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. The new case here is the new user and similarity is calculated between the movies rated by the using using the ratings field as a deciding metrics. We are now creating a similarity matrix using the above algorithm. There are two choices to understand the distance ( similarity metrics ) between two datapoints in vector space namely Euclidean Distance and Cosine Similarity Matrix. We chose cosine similarity matrix since in vector space it gives more accurate measurements for our dataset. Cosine similarity between two vectors corresponds to their dot product divided by the product of their magnitudes. If x and y are vectors as defined above, their cosine similarity Rendered by QuickLaTeX.com is:

IMG_256

The relationship between cosine similarity and the angular distance which we discussed above is fixed, and it’s possible to convert from one to the other with a formula:

Rendered by QuickLaTeX.com

After creating the similarity matrix we will fit the dataset into the similarity matrix creating a similarity matrix with our dataset. This similarity matrix is the Machine Learning Model that we , in the future, use to predict movies for the user based on the user’s rating for related movies.