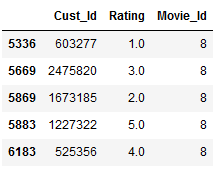
**Hybrid Recommender System**

To build this model, the team uses a Python package named, LightFM. LightFM is a Python implementation of popular recommendation algorithms for both implicit and explicit feedback. It also makes it possible to incorporate both item and user features into the traditional matrix factorization algorithms.

1. **Interaction and Weights Matrix Creation**

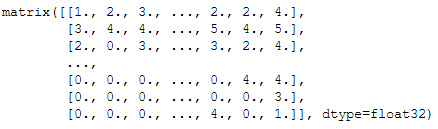
In order to reduce dimension of the sparse matrix, as well as to reduce time used for training the model, the team decided to analyse popular movies with ratings > 10000 (0.89 quantile) and active users with ratings > 3000 (0.97 quantile). The total size of the filtered dataset is 335,679 rows.



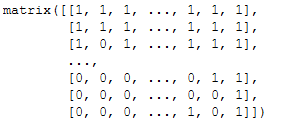
*Filtered data*

Firstly, an interaction matrix is being created, with Movie\_Id along the X-axis and Cust\_Id along the Y-axis. Each of the cells inside the interactive matrix corresponds to a weights matrix which is the ratings of the Cust\_Id according to Movie\_Id which they have watched.

Movie\_Id



Cust\_Id



*Interaction matrix: binary coded. ‘1’ signifies that the customer has watched the movie, vice versa for ‘0’.*

*Weight matrix: contains float ratings of the customer curresponding to movie*

**Train-test spit**

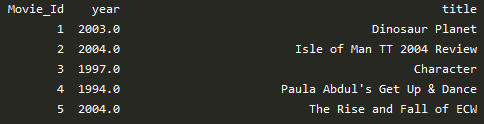
The interactions matrix is then split into 60% training dataset and 40% testing datasets.



*Random train-test sampling*

1. **Movies Feature Creation**

There are total of 17,700 movies and each of them consists of the year the movies were released and their title. To create the text-based feature, the movie title is then being used to convert into TF-IDF matrix.



*Movie titles dataset*

Prior to converting into TF-IDF matrix, text data cleaning is being done to further reduce the TF-IDF matrix dimension and to remove redundant texts. The following pre-processing steps were being carried out:

Tokenization and lowercase

Remove digits

Remove punctuation

Lemmatization and Stemming

Concatenating words into sentences

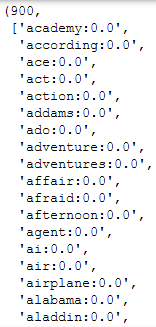
Stopwords Removal

Lastly, the sentences will be converted into a TF-IDF sparse matrices.



*TF-IDF matrices*

Each rows of TF-IDF matrix is then map corresponding to a Movie\_Id:



TF-IDF features and weightage pairing

Movie\_Id

1. **Evaluation**

After tuning the parameters of the model:

* Number of components: 50
* Learning Rate: 0.05
* Loss: “warp”
* K: 15

The model gives an evaluation score of the following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **epochs** | **auc\_train** | **auc\_test** | **mean\_precision\_train@10** | **mean\_ precision\_test@10** |
| LightFM | 1 | 0.601 | 0.564 | 0.569 | 0.337 |
| 20 | 0.608 | 0.569 | 0.587 | 0.324 |

The model is deemed overfitted because the mean\_precision\_train@10 clearly triumph over mean\_precision\_test@10.

The mean\_precision\_test@10, which average at 0.33 signifies that out of 10 movies recommended to a user, only 3 are relevant to the user liking. Thus, the performance of the model is low and ideally in the real world, the model should return a mean\_precision\_test@10 of 0.8 to prevent customer churn.

Nonetheless, the model can be further improved through introducing more users and movies to the training dataset or substituting TF-IDF sparse matrices mapping to more specific features, for instance (genre, director, actors, popularity, etc…) to reduce the sparsity of the movie features sparsity.