Movie Recommender System with DSS

Team 11: Sudeep Kurian, Yachu Liu, Muthuraja Palaniappan, Shubhansh Jain

Shiny App: <https://sudeep2711.shinyapps.io/final/>​

Shiny App Presentation Video: <https://www.youtube.com/watch?v=iPxmS8fYdEo&t=332s>

Github Link: <https://github.com/liu1498/Movie-Recommender-System-ShinyApp>

Dropbox Link: <https://www.dropbox.com/sh/n74uiz900mrz7fd/AABpWnbHvSz2-Y0jvheNk_qqa?dl=0>

Business Problem Definition

* Users of movie rentals prefer a movie recommendation that will match their ‘taste’ in movies
* Movie rental businesses want to recommend movies for new users whose preferences are not known beforehand
* Collaborative Filtering approach can be used whereby the similarity of the ratings of the new user with that of the ratings of the other users are identified and make recommendations
* Based on the genre of the movies that the user likes and the movies that he/she likes in those genres will give an idea of what cluster of users he/she belongs to
* With our Recommender Systems, a movie rental website can identify which movie the user will like the most and recommend and increase their business and build customer loyalty
* Solving this will be a game changer for movie rental businesses and hence they are keen

## Analytics Problem Definition

* Given the matrix of users and their ratings and a new user’s rating, identify the k nearest neighbor users that are similar to the new user and recommend movies that they rated high
* With User Based Collaborative Filtering (UBCF) approach, find Cosine similarity of new user with other users and identify movies that users in the cluster liked
* Assumptions: Genres of movies in database are right; database has good collection of movies
* The recommendation should match the similarity of ratings given as input. It should correspond to the filters selected like the genres and years
* The movie rental businesses are eager to use Analytics to solve this problem and gain revenues

## Data

* Need large dataset to find k nearest neighbors with high similarity of new user with other users and provide good recommendations. Large Dataset of users and ratings required to train model.
* Obtained data from Movie Lens, IMDB and TMDB databases that is sufficient to find k nearest neighbors with highest similarity.
* Reduced bias by rescaling data of users who always give higher or lower ratings for all movies and removing movies with less than 10 ratings,Extracted year from movie title.
* Databases merged with movie id as the connector.
* Ratings for most popular movies such as Shawshank Redemption are high for most users; number of ratings for such movies is also high; Hence, most popular movies are recommended due to this. The performance of our app is quite fast.

## Methodology Selection

* Item Based Collaborative Filtering (IBCF), UBCF and Content based were considered for solving this problem. Content Based recommenders were not apt since we did not possess data about each movie other than ratings. Preferred UBCF over IBCF as it will cluster similar users instead of clustering similar movies. Since we cluster movies based on Genre filter IBCF was not required. After finding cosine similarities UBCF picks up k nearest neighbors, and then uses the ratings of similar users as weights to calculated weighted average of movies and picks up the top 10 movies.
* R is apt for this problem because it is an open-source and has pre-built libraries for Recommender Systems. This allows us to better data preparation that helps us provide better recommendations. We got our motivation for data preparation and model building from Jekaterina Novikova PhD <https://rpubs.com/jeknov/movieRec>
* We initially test with a limited dataset (100 top users vs 100 top movies) to see the performance of UBCF algorithm. We find it to be satisfactory as the recommendations are relevant.

## Model Building

* Performed Validation, Bootstrapping and K-fold validation to evaluate the model. Plots of ROC curves shows that UBCF Cosine distance performs better than UBCF Pearson Correlation similarity. Also, its Precision-Recall curve is better.
* We calibrated the model by trying multiple k values for nearest neighbors and identified 30 as the best performing one.
* Since we identified Cosine similarity as the best one, we used that.
* Assumptions: Genres of movies in database are right; Users with similar preferences will rate items similarly.
* Limitations: The whole database has to be kept in memory; Expensive similarity computation between active user and all other users.
* Constraints: recommendation should correspond to the filters selected like the genres and years

## Functionality

* Our digital support systems have three tabs. There are two panels on our home tab. The users can select up to three genres of movie that they are interested in on the left panel, then the users can choose a favorite movie of each genre on the right panel. Our system has another two features that allow users to filter the recommended movies by year and by genre. After selecting all the options, our systems will generate ten recommended movies.
* If the users are curious about the recommended movies, they can go to the second tab, and search for the movie. Our system will provide the overview and poster of the movie. If the users are interested in more details, the system also provides the IMDB link of the movie.
* This tool does not only provide us the results, but also provide an interactive platform for the end users.
* RecommenderLab is one of packages we found extremely useful to build this system.

## GUI Design & Quality

* The tool works well without any error.
* Our Shiny App appears better than the Shiny example templates. We work on the details and try to provide end users as much information as possible. At the meantime, we design our tab to be user-friendly. Therefore, any user can understand our App easily and uses it without confusion.
* We are confident in our Shiny App, and we believe our Shiny App appears high quality.