1. Download Raw Datasets in the following websites:

complete\_dataset\_url = 'http://files.grouplens.org/datasets/movielens/ml-latest.zip'

small\_dataset\_url = 'http://files.grouplens.org/datasets/movielens/ml-latest-small.zip'

1. Python Files:
   1. movie\_app.py: Flask part to connect the front end. Two features implemented:
      1. Enter User Id and a number. The back end returns the top number of predicted movies.
      2. Enter User Id and the selected movie. The back end returns the predicted rating of the selected movie.

b) movie\_server.py: Create the server

c) movie\_engine.py: The Spark part to train the Recommendation model. Three functions implemented:

i. “target\_user\_movies\_ratings”: Given user id and the selected movie, it returns the predicted rating of the selected movie.[Front End part implemented]

ii. “recommend\_top\_movies”: Given user id and the number, it returns the number of movies with the top predicted ratings.[Front End part implemented]

iii. “ratings\_new\_user”: Given a file path of new added user with ratings(see “user\_file” under “data\_set” as an example), it re-trains the ALS model. The front end can get the user input to be saved as a file, then the back-end calls this function to add the new data into the whole dataset.[Front End part Unimplemented]

d) engine\_whole.py: All spark codes. The movie\_engine.py is a sub-part of the engine\_whole.py. Mostly all the spark-processed data are calculated before we run the server(in the “spark\_content”). Thus, you only need to run movie\_engine.py when building the server. If you need to use spark contents, just access the datasets in the spark\_content folder.

1. Spark\_content Folder:

There are four folders totally:

1. movie\_cluster:
   1. Description: Spark Movie K-means Clustering results. All the movies in the movies.csv are clustered into five different categories: Action, Comedy, Documentory&Adventure, Drama and Horror&Thriller. The “distance” in the datasets is the distance to the cluster center point. And the movies in each cluster have already been sorted by the distance. The smaller the distance, the bigger livelihood the movie belongs to the cluster.
   2. Dataset: movie.csv and rating.csv
   3. Front End Usages:
      1. The website can give links to the users so that they can click to get a list of movies in the category they interest. [Important]
      2. Distance may be visualized by using Matlab tool. (see matlab\_visualzation.pdf) [Suggested]
2. movie\_cluster\_ratings\_sort:
   1. Description: Spark SQL results.
      1. The folder “cluster” is based on the movie\_cluster folder and ratings.csv. All the five clusters in the folder is the same clusters in (1). But instead of using “distance”, it uses “ratings”. All the movies are firstly calculated to get their average ratings. And then the movies are sorted by using their average ratings.
      2. The “average\_ratings\_sorted” file lists all the movies with their average ratings. Also, the file is sorted by their average ratings.
      3. The “average\_ratings” file lists the average ratings for different clusters.

b) Datasets: folder movie\_cluster and movie.csv

c) Front End Usages:

i. The Front-End can give ten top ratings in “average\_ratings\_sorted” file to new user and let them rate the ten movies. The ten rated movies then can be saved and use the “ratings\_new\_user” function to add into the whole dataset. Then the recommendation model is able to recommend to the new user. [Important]

ii. The Front-end is able to return all the average ratings for different clusters and we can always give users the top rated movies in different clusters(genres).[Important]

1. . User\_cluster:
2. Description: K Means Clustering for users. The users are clustered into five different clusters.
3. Datasets: movie.csv and ratings.csv
4. Front End Usages:
   * 1. The Front End can give links for users to see different users in different clusters.

[important]

1. . Visualization:
   1. Description: All the users give comments to different movies. The genome-scores.csv gives all the relevance between movies and tags. The bigger the relevance, the more related the movies to tags. The “tag\_classification” sorted the tags which belongs to different movies by using the relevance. Each movie has its own file with the top 100 most relevant tags.
   2. Datasets: genome-scores.csv and genome-tags.csv
   3. Front End Usages:

i. If a user requests a movie, the front end accesses the file in “tag\_classification” and pass the content into front end. The front end then accesses a word cloud webpage “http://www.wordle.net/compose” to generate a word cloud picture. The “index.html” file is an example, I just copy the content of one file in “tag\_classification” to the index.html. If you simply open it by using a browser and click the submit button, you can see the visualization. The sample.jpg is the word cloud my example generated. In our website, we can use Flask to pass the data into html. [important]

1. . datasets: it contains the data in the ml-latest-small raw data. It can be used for testing and debugging. Also, we can use it in our final project demo.
2. . templates: all front-end files.
3. . start\_server.sh: start to run our website.
4. . previous: I also uploaded two useful pdf files of previous project:
   1. matlab\_visualzation.pdf
   2. movie\_recommendation.pdf