Steps to Execute the experiments: -

There are 4 files at the below path which are used to trigger the experiments:

- /CI-ACF-2017/src/alg/ib/ExecuteIB_ML20M_Experiment_1.java
- /CI-ACF-2017/src/alg/ib/ExecuteIB_ML20M_Experiment_2.java
- /CI-ACF-2017/src/alg/ib/ExecuteIB_ML20M_Experiment_3.java
- /CI-ACF-2017/src/alg/ib/ExecuteIB_ML20M_ContentBasedCF.java

Each file has a main method, executing which will trigger the experiment and post the output on console.

Experiment 1:

On triggering the experiment one it will create 4 prediction output files at the following path:

- /CI-ACF-2017/results/Deviation_From_Mean_With_Neighbourhood_predictions.txt
- /CI-ACF-2017/results/Non_Personalised_With_Neighbourhood_predictions.txt
- /CI-ACF-2017/results/Simple_Average_With_Neighbourhood_predictions.txt
- /CI-ACF-2017/results/Weighted_Average_With_Neighbourhood_predictions.txt

These files contain the prediction for neighborhood size 100 for each of the predictors.

Experiment 2:

On triggering the experiment two it will create 1 prediction output files at the following path:

- /CI-ACF-2017/results/Deviations_From_Mean_Using_Neighbourhood_Thresholding_predictions.txt

This file contains the prediction for thresholding size 0.8 for Deviation From Item Mean Predictor

Experiment 3:

On triggering the experiment three it will create 3 prediction output files at the following path:

- /CI-ACF-2017/results/Cosine_Similarity_Using_100_Neighbours_predictions.txt
- /CI-ACF-2017/results/Pearson_Correlation_With_100_Neighbours_predictions.txt
- /CI-ACF-2017/results/Pearson_Correlation_With_Significance_Weighing_Using_100_Neighbours_predictions.txt

These files contain the prediction for neighborhood size 100 for using different metrics

Part 2: Content Based

On triggering the experiment, it will create 2 prediction output files at the following path:

- /CI-ACF-2017/results/Content_Based_predictions.txt
- /CI-ACF-2017/results/Content_Based_With_Ratings_GreaterThan_4_predictions.txt

These files contain the prediction for Content based for using different predictors (base and one with rating greater than 4)

Code Explanation:

A new package was added called **alg.constants** which contains the file **Constants.java** . This file contains all the constants which are used in the project. This was done to make the code clean and have lesser hard coding in the Code.

In order to implement the Content Based Recommender and also define the files for all the 3 experiments to be conducted, I have created the below files in the **alg.ib** package:

ExecuteIB_ML20M_Experiment_1.java - This is the class which is used to run the experiment for the 1st Experiment in which we are testing the different predictors against varying neighborhood sizes using Cosine Similarity. In this, I implemented the method `generatePredictionAndCalculateRMSE` which takes in the item file path, item genome scores file path , training and test data file , output file name and a string parameter to specify what type of Predictor is to be used in the experiment .The output file name changes based on the type of experiment which is being executed. Also in this method, I have implemented a for loop to run it from neighborhood size 5 to 100.

ExecuteIB_ML20M_Experiment_2.java - This is the class which is used to run the experiment for the 2nd Experiment in which we are testing the Deviation From Item Mean Predictor using Cosine Similarity against varying similarity thresholds. In this, I implemented the method `generatePredictionAndCalculateRMSE` which takes in the item file path, item genome scores file path , training and test data file , output file name. Also in this method, I have implemented a for loop to run it from threshold size 0 to 0.8 .

ExecuteIB_ML20M_Experiment_3.java - - This is the class which is used to run the experiment for the 1st Experiment in which we are testing the different similarity metrics against varying neighborhood size of 100 using Deviations from Item Mean Predictor. In this, I implemented the method `generatePredictionAndCalculateRMSE` which takes in the item file path, item genome scores file path , training and test data file , output file name and a string parameter to specify what type of similarity metric is to be used in the experiment .The output file name changes based on the type of experiment which is being executed.

ContentBasedCF.java – This is the class which implements the CFAlgorithm interface and passes the genome scores file to be used for calculating the similarity and generating the predictions.

ExecuteIB_ML20M_ContentBasedCF.java – This is the class which is used to run the experiment for Content Based CF. In this, I implemented the method `generateContentBasedPredictionsAndCalculateAccuracy` which takes in the itemfile file path, item genome scores file path , training and test data file , output file name and a Boolean parameter to check if the experiment needs to generate predictions for ratings greater than 4 or not. The output file name changes based on the type of experiment which is being executed.

I also implemented a class **NearestNeighbourhoodWithSimilarityThresholding.java** in the package **alg.ib.neighbourhood** . This class is responsible for finding neighbors based on the similarity threshold passed in the constructor and has a default value of Interger.MAX_VALUE which is a safety net in case no value is passed. In this I have defined a method called `computeNeighbourhoods` which checks if the value of the similarity is greater than the threshold value, only then does it add to the set of the neighbors.

In the package alg.ib.predictors , I implemented the below classes :

ContentBasedCosinePredictor.java – This class implements Predictor and in its constructor we pass a Boolean to the paramter `isRatingGreaterThanFour` to specify if the predictor needs to be executed considering only ratings greater than 4 or should it execute as the default predictor . It then implements the formula for Content based CF

DeviationsFromMeanPredictor.java - This class implements Predictor and in the getPrediction method it implements the formula for Deviations from Item mean where the similarity of an item is multiplied by the difference of the rating and the mean value of the item for which the rating has been found, the overall result is summed up and divided by the similarity weight which is then added to the mean value of the item and the result is returned.

NonPersonalisedPredictor.java - This class implements Predictor and in the getPrediction method the mean value of the item is returned.

WeightedAveragePredictor.java - This class implements Predictor and in the getPrediction method we multiply the similarity of the item with its rating and them divide the overall sum by the similarity weights to find the predicted rating.

In the **similarity.metric** package , I have implemented two classes :

CosineSimilarityMetric.java – This class implements `SimilarityMetric` and in the `getSimilarity` method we implement the cosine similarity by taking the dot product of the ratings and then using the dot product of the normalized values.

PearsonMetricWithSignificanceWeighting.java - This class implements `SimilarityMetric` and in the `getSimilarity` method we calculate the Pearson metric and then check the number of common items then divide it with a fixed value of 100 to calculate the weight .