

Amazon Recommender System

Recommender engines are the most immediately recognizable machine learning technique in use today.

How it works?

- The recommender requires input—data on which it will base its recommendations.
- This data takes the form of preferences in Mahout. Because the recommender engines that are most familiar involve recommending items to users, it'll be most convenient to talk about preferences as associations from users to items—these users and items could be anything.
- A preference consists of a user ID and an item ID, and usually a number expressing the strength of the user's preference for the item.
- IDs in Mahout are always numbers—integers, in fact.
- The preference value could be anything, as long as larger values mean stronger positive preferences.
- For instance, these values might be ratings on a scale of 1 to 5, where 1 indicates items the user can't stand, and 5 indicates favorites.

Data Cleaning:

For recommender system I needed only three data columns from AmazonReviews dataset i.e Customer ID, Product ID and Star_Ratings. So , I have used following command to get a csv file that contains only CustomerID, ProductID and ratings.

```
awk -F'\t' '{print $2","$4","$8}' /home/sayali/Downloads/AmazonReviews.tsv >> /home/sayali/Downloads/mahoutinput.csv
```

But I found that there were few missing values, Strings were present in dataset. So I have carried out multiple operations using vim editor.

- `g/[a-zA-Z]/d`
- `%s/"//g`
- `%s/,//g`

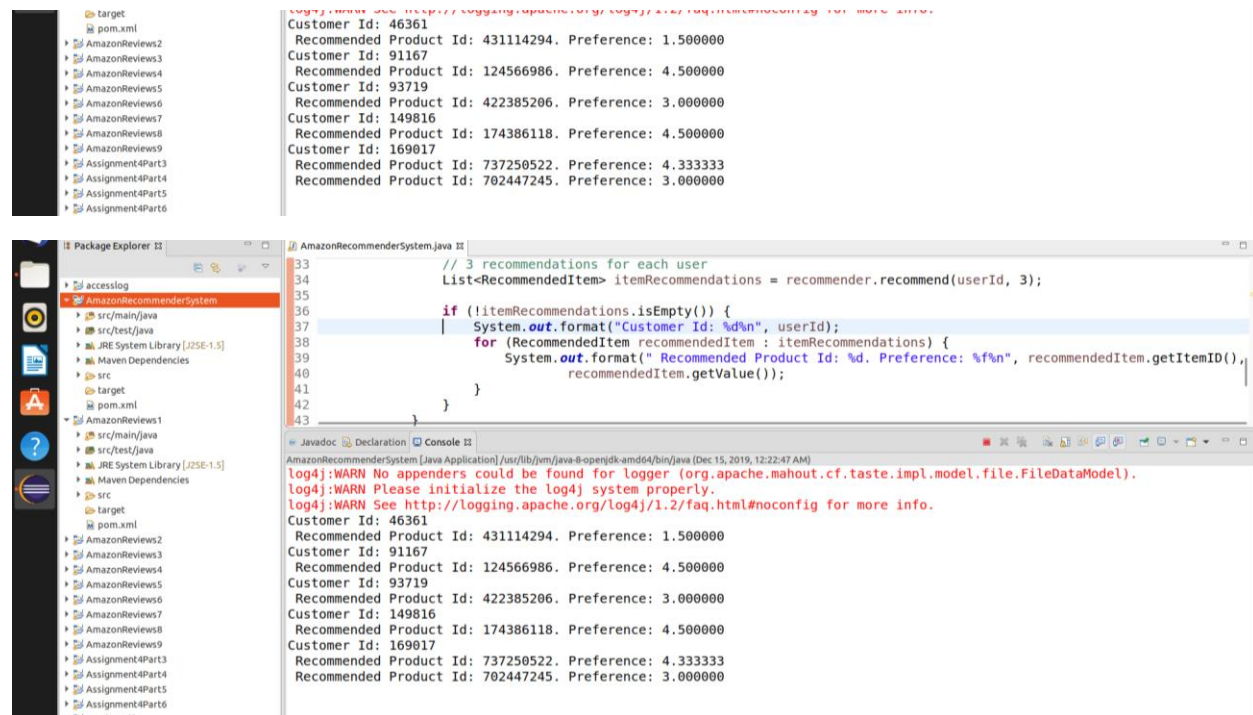
The input file looks like:



```
1797882,2102612,5
18381298,106393691,5
19242472,375449471,5
19551372,255651889,5
14802407,116158747,5
2909389,166146615,4
19397215,111742328,5
3195210,255803087,5
52216383,246816549,5
10278216,9612905,1
24655453,292127037,4
30788223,311309827,5
11257536,390030149,4
29605778,794858888,5
41238422,874223919,5
23620123,464001209,5
25564245,328680790,5
801375,620737389,5
9478730,351973146,5
18418407,10034795,5
42644737,379370722,3
4224840,371977855,5
49631440,296319484,4
15548310,585793391,5
1778076,22707685,5
4028997,709054453,5
41137196,57378789,2
1778076,343124386,5
22904338,858538418,5
```

OUTPUT:

Output gives Customer Id, recommended product and strength of preference.



```
Customer Id: 46361
Recommended Product Id: 431114294. Preference: 1.500000
Customer Id: 91167
Recommended Product Id: 124566986. Preference: 4.500000
Customer Id: 93719
Recommended Product Id: 422385206. Preference: 3.000000
Customer Id: 149816
Recommended Product Id: 174386118. Preference: 4.500000
Customer Id: 169017
Recommended Product Id: 737250522. Preference: 4.333333
Recommended Product Id: 702447245. Preference: 3.000000
```

```
33 // 3 recommendations for each user
34 List<RecommendedItem> itemRecommendations = recommender.recommend(userId, 3);
35
36 if (!itemRecommendations.isEmpty()) {
37     System.out.format("Customer Id: %d\n", userId);
38     for (RecommendedItem recommendedItem : itemRecommendations) {
39         System.out.format(" Recommended Product Id: %d. Preference: %f\n", recommendedItem.getItemID(),
40                             recommendedItem.getValue());
41     }
42 }
43 }
```

```
AmazonRecommenderSystem [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 15, 2019, 12:22:47 AM)
log4j:WARN No appenders could be found for logger (org.apache.mahout.cf.taste.impl.model.file.FileDataModel).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Customer Id: 46361
Recommended Product Id: 431114294. Preference: 1.500000
Customer Id: 91167
Recommended Product Id: 124566986. Preference: 4.500000
Customer Id: 93719
Recommended Product Id: 422385206. Preference: 3.000000
Customer Id: 149816
Recommended Product Id: 174386118. Preference: 4.500000
Customer Id: 169017
Recommended Product Id: 737250522. Preference: 4.333333
Recommended Product Id: 702447245. Preference: 3.000000
```

Recommendation Code

```
package sayali.AmazonRecommenderSystem;
import java.io.File;
import java.io.IOException;
import java.util.List;
import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;
import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.neighborhood.ThresholdUserNeighborhood;
import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.CityBlockSimilarity;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;
import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;
import org.apache.mahout.cf.taste.similarity.UserSimilarity;
public class AmazonRecommenderSystem {
    public static void main(String[] args) {
        try { DataModel model =
            new FileDataModel(new File ("/home/sayali/Downloads/mahoutinput1.csv"));
            UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(model);
            UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, userSimilarity,
            model);
            UserBasedRecommender recommender =
                new GenericUserBasedRecommender(model, neighborhood, userSimilarity);

            for (LongPrimitiveIterator iterator = model.getUserIDs(); iterator.hasNext();)
            {
                long userId = iterator.nextLong();
                // 3 recommendations for each user
                List<RecommendedItem> itemRecommendations = recommender.recommend(userId, 3);
                if (!itemRecommendations.isEmpty()) {
                    System.out.format("Customer Id: %d%n", userId);
                    for (RecommendedItem recommendedItem : itemRecommendations) {
                        System.out.format(" Recommended Product Id: %d. Preference: %f%n",
                        recommendedItem.getItemID(), recommendedItem.getValue());
                    } } }
            catch (IOException ex) {
                System.out.println("Exception: " + ex.getMessage());
            }
            catch (TasteException e) { e.printStackTrace(); }
        } }
```