Project III: Association Rule Mining

CSE 5334: Data Mining
Name: Hamza Reza Pavel
Student ID: 1001741797
Group: 2

Introduction

In this project, we are going to use association rule mining using APRIORI algorithm on the IMDB dataset. The APRIORI algorithm does the part of frequent item set mining and association rule learning. The intuition behind the APRIORI algorithm is that any subset of a frequent itemset must also be frequent. Based on this intuition APRIORI algorithm prunes items which are not frequent in each iteration.

In this project we will be using APRIORI algorithm to do a market basket analysis on the IMDB dataset. This analysis on the dataset might give us insight such as which actors appeared together the most, or which movie genres were popular during a particular period.

Analysis of the Dataset

The dataset provided for team 2 consist of the period from 1990-1999. The dataset consists of two files. The first file, *imdb_actor.csv* contains the actor information. This file is a csv which contains the transaction id and actor name in each row. Transaction id represents the movie ID. Actors having same transaction id appeared in the same movie together. The second file, *imdb_movie.csv* contains the movie genre and year released information of the movies. This is also a csv file where each row contains the transaction id, movie genres separated by comma, and the year the movie was released. The *imdb_actor.csv* file has a total of 81,327 rows and *imdb_movie.csv* file has a total of 109682 rows.

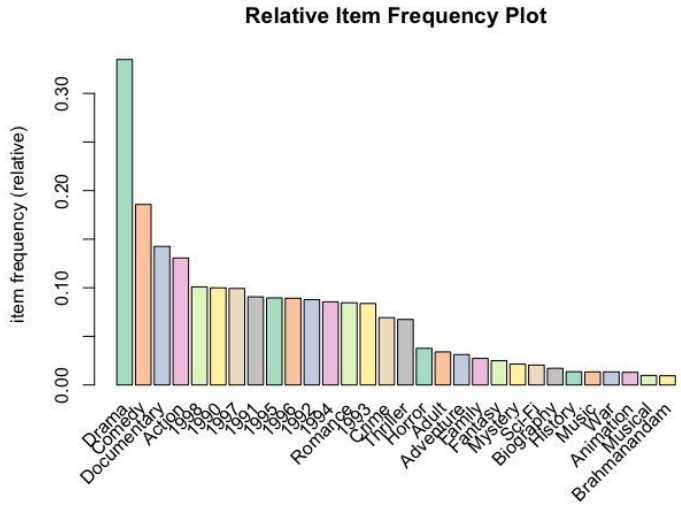


Fig. 1: The most frequent 20 items in the dataset.

Data Preprocessing

As we will be doing market basket analysis using the dataset, our goal is to transform the given dataset into a transaction dataset. In a transaction dataset, items belonging to the same transaction appear together in the same list. In case of our dataset, we have to group the

movie genre, year release, and actors of the same movie into a single row. We can use R for the preprocessing steps. The dataset was preprocessed using the following steps:
1. First, we read both csv files using R.
2. We remove empty white spaces, "??"marks which indicate missing characters, or any other special symbols from both dataset. As hyphenates are part of some names, we keep the hyphens.
3. As our transaction IDs indicate the movies, we need to group the actors who appeared in the same movie into a single row. We can do this by using the *dplyr* library of R.
4. We combine both data-frame using the *bind_rows* function.
5. Once both data-frames are bound, they will have two column for each transaction id. One indicating the movie year and genre. The other one indicating the actors who appeared in that movie. We again group the rows having same transaction ID into a single row. This way, each movie will be represented by a single transaction id. Meaning all the items of the transaction will be in the same row.
6. We remove the transaction id from the data-frame as APRIORI algorithm in R only takes the list of items as input. We write the processed output into a csv file named *processed_itemlist.csv*.

Data Collection

Once we have preprocessed the dataset, we collect data for our analysis steps. In first step, we collect the number of Candidate itemset, frequent itemset, and pruned itemset for each iteration. To find out the frequent itemset, we call the default *apriori* function in R with the parameter *target = 'frequent'*. This will give us the frequent itemset for each iteration. Once we have the frequent itemset, we can use the frequent itemset in each iteration to generate and count the candidate itemset for next iteration. This can be done by joining the frequent itemset of each iteration. Joining the frequent itemset of each iteration will give us the candidate itemset for next iteration.

Table 1: Candidate, Frequent, and Pruned Itemset count for min_sup = 0.003.

Confidence	0.05			0.07			0.08		
Iteration	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count
1	36623	33	36590	36623	33	36590	36623	33	36590
2	528	143	385	528	143	385	528	143	385
3	20449	51	20398	20449	51	20398	20449	51	20398

As our support value and confidence value gives us frequent itemset up to three iteration, we collect data for different support and confidence combination for three iteration.

Table 2: Candidate, Frequent, and Pruned Itemset count for min_sup = 0.007.

Confidence	0.05			0.07			0.08		
Iteration	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count
1	36623	30	36593	36623	30	36593	36623	30	36593
2	435	80	355	435	80	355	435	80	355
3	6400	3	6397	6400	3	6397	6400	3	6397

The collected data for *min_sup* value of 0.003, 0.007, and 0.008 are in Table 1, Table 2, and Table 3 respectively.

Table 3: Candidate, Frequent, and Pruned Itemset count for min_sup = 0.012.

Confidence	0.05			0.07			0.08		
Iteration	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count	Candidate Itemset Count	Frequent Itemset Count	Pruned Itemset Count
1	36623	28	36595	36623	28	36595	36623	28	36595
2	378	48	330	378	48	330	378	48	330
3	2304	1	2303	2304	1	2303	2304	1	2303

To collect the number of rules for each of the support and confidence value pair, we run the *apriori* function with those values without the *target = 'frequent'* parameter. Thus the function will return us the set of rules for those values. We can count the length of the set which will give us the total number of rules. As we are only interested in rules with both LHS and RHS values, we set the *minlen* parameter to *2*. The Table 4 the number of rules for each min_sup and confidence value pair.

Table 4: Number of rules for different min_sup and conf values.

	Min_sup = 0.003	Min_sup = 0.007	Min_sup = 0.012
Conf = 0.05	386	163	99
Conf = 0.07	379	161	98
Conf = 0.08	367	154	97

Analysis Tasks

In this section we are going to perform the analysis tasks for this project.

I. Change in Candidate Itemset

As we know, any valid itemset is a candidate itemset in apriori algorithm. A frequent itemset in an iteration becomes the candidate itemset for the next iteration. Thus, based on support, the candidate itemset will change, as frequent itemset for each iteration will change

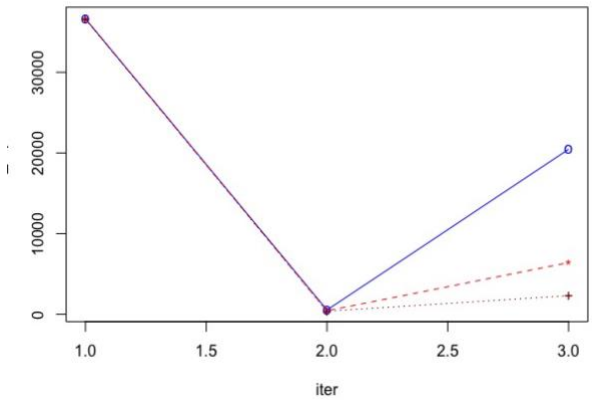


Fig. 2: Change in candidate itemset for different min_sup values and iteration. Blue line for Min_sup = 0.003, green dashed line for Min_sup = 0.007, and red dotted line for min_sup = 0.012

based on the min_sup values. As we observe from Table 1, Table 2, and Table 3, there is no change in candidate itemset for different confidence values, but the change depends on the support values.

From the graph we see that, irrespective of support value, the initial itemset for one item candidate list is same. The two item candidate list is substantially small, then again there is a surge in three item candidate list.

II. Change in Frequent Itemset

We use the *apriori* function in R to count the itemset in different iteration for different values of support. As we see from Fig. 3, the frequent itemset which fulfills the support for one item set is lower, then for two item frequent itemset increases. Then again decrease for three items. Also, the number of frequent item at each iteration is higher for lower support values.

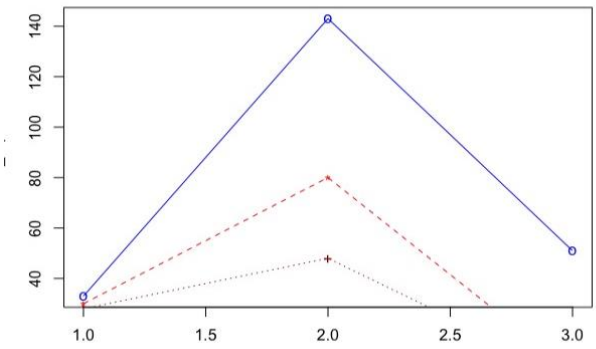


Fig. 3: Change in frequent itemset for different min_sup values and iteration. Blue line for Min_sup = 0.003, green dashed line for Min_sup = 0.007, and red dotted line for min_sup = 0.012

This is due to the constraint support value imposes. Lower the support value, higher the chance for an item set to be a frequent itemset.

III. Change in Pruned Itemset Count

We plot the pruned itemset for each support value in Fig. 4. As we can see, lower the support value, higher the count of pruned items. This is due to the cause that more item makes it to the candidate list if the support value is lower, hence there is a chance that more items will be pruned at each iteration.

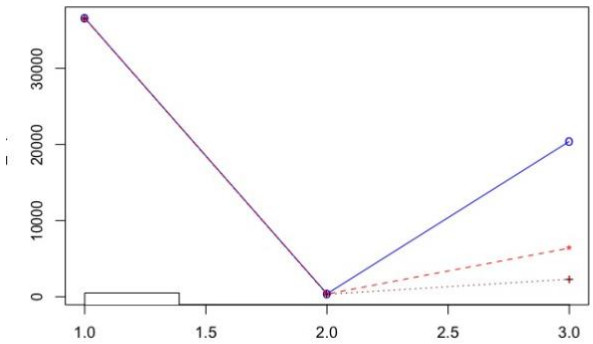


Fig. 4: Change in pruned itemset for different min_sup values and iteration. Blue line for Min_sup = 0.003, green dashed line for Min_sup = 0.007, and red dotted line for min_sup = 0.012

IV. Plotting Rules Count for Different Support and Confidence Values

We plot the rules count for different support and confidence value using the data from the Table 4. The fig. 5

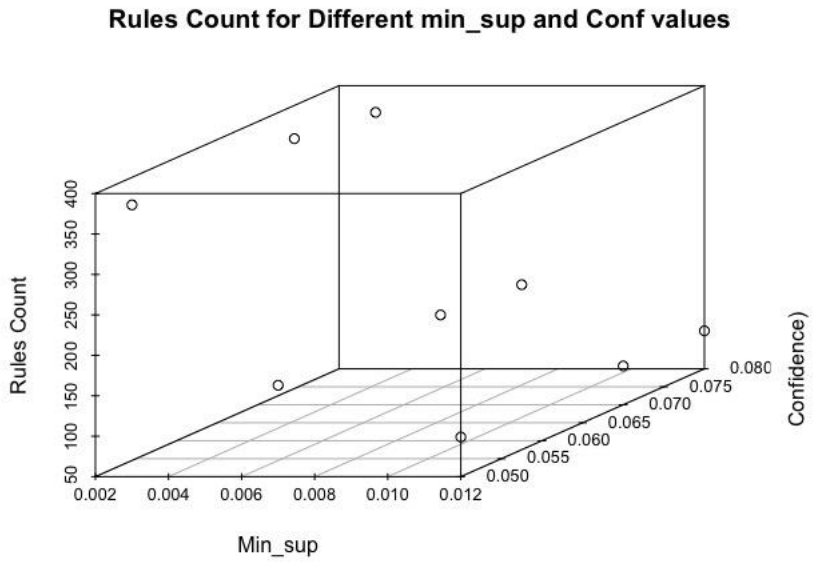


Fig. 5: Rules count for different support and confidence values.

We can observe that, there is a significant change in rules count for change in support value. As min support value increase, there is a drastic decrease in number of rules. As confidence value increase for the same support value, there is also decrease in number of rules. But this change in rules due to increase in confidence value is smaller than the change in rules count due to increase in support values. This is due to the reason that, as support values increase, the frequent item lists will decrease. Hence less chance to construct rules from those frequent item list.

V. Filter at Least 5 Rules for Lift > 1, Lift< 1, and Lift = 1

As we have most rules for min_support value = 0.003, and confidence 0.05, we filter out the rules generated for min_support = 0.003 and conf = 0.05.

For lift > 1, we get the following rules:
a. {Romance} => {Drama}
b. {Drama} => {Romance}
c. {1998} => {Drama}
d. {Drama} => {1998}
e. {1997} => {Drama}

This means there is an association between romance and drama genre. There is a high probability that a movie of drama genre is going to be also of the romance genre and vice versa. It's also noticeable that most drama movies in the dataset were released in 1998 and 1997.

For lift < 1, we get the following rules:
a. {Comedy} => {Drama}
b. {Drama} => {Comedy}
c. {Action} => {Drama}
d. {Drama} => {Action}
e. {Thriller} => {Drama}

This means there is no association between these genres. If a movie is of comedy genre, its highly unlikely that it would be of drama genre. Same goes for relationship between drama and action genre, and drama and thriller genre.

For the dataset, no rules were generated for lift = 1.

VI. Visualizing the top 100 Rules and Rules in Section V

For visualizing the top 100 rules, we use the *aRulesViz* library of R. The fig. 6 shows the top 100 rules with their confidence, support, and lift values.

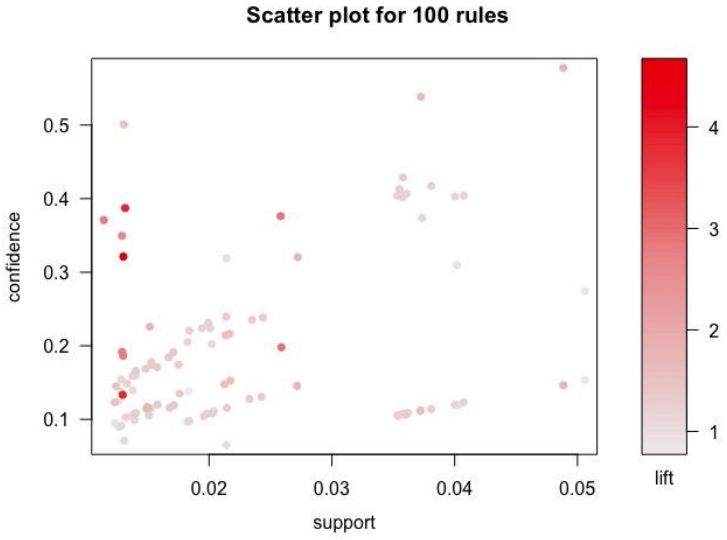
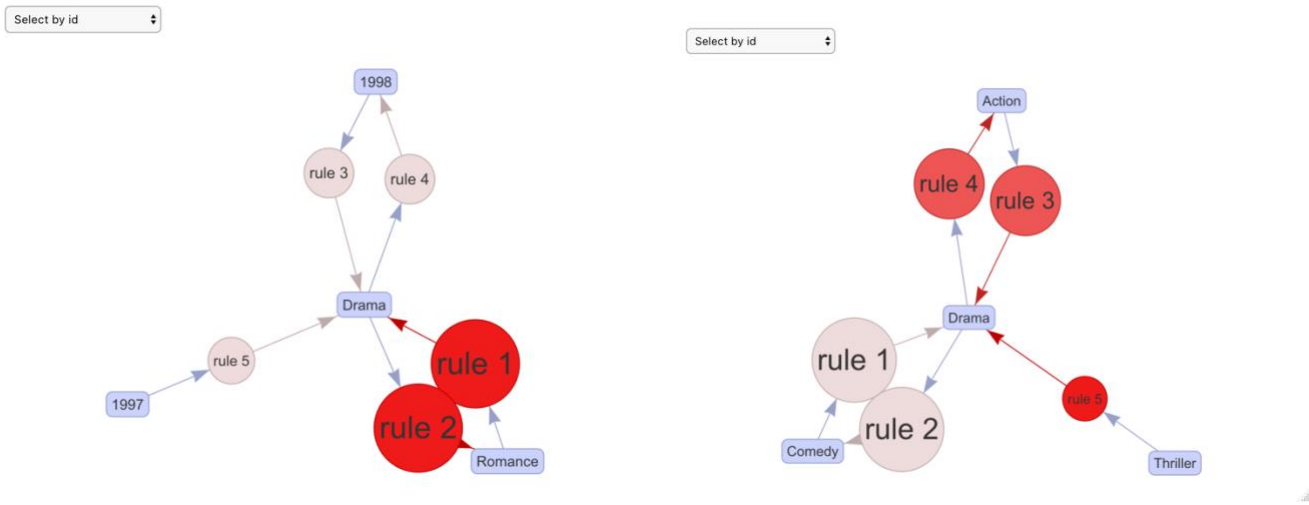


Fig. 6: Top 100 rules with their support and confidence values.

The Fig. 7a shows the 5 rules with lift >1 and Fig. 7b shows the 5 rules with lift < 1.



(a) (b)

Fig. 7: (a) Rules with lift > 1 (b) Rules with lift < 1

Overall Status

Finished the project in entirety by finishing the mentioned tasks. First, downloaded the dataset and preprocessed it. Then got familiar with APRIORI algorithm, support, confidence, lift, frequent itemset, candidate itemset, and how support and confidence impact the output of the algorithm. After that played around with the "beer-diaper" dataset used in class to get a better understanding of the algorithms output. Finally, ran the analysis on the processed IMDB dataset.

File Descriptions

Apart from the project report file (this file), there are two additional directory. The directory named *source_code* contains the R scripts used for the project. The *source_code* folder contains two R script files:

a. preprocessing.R: This file contains code related to preprocessing the provided dataset. Once the dataset is preprocessed, this script writes the output in a CSV file.

b. apriori_analysis.r: This script contains the analysis and plotting related code of the project. The output CSV files from *preprocessing.R* script is required to run this script.

The directory named *input_output* contains the dataset and preprocessed output by the *preprocessing.R* script.

Division of Labor

The entire project was done alone.

Challenges Encountered

In terms of analysis, this project was a little bit harder. Following are some of the challenges I faced during working on this project:

1. Preprocessing the dataset required considerable effort compared to the other projects. As each genre and actor names were a separate rows, first each csv file had to be preprocessed

separately, joined into a single data-frame, and finally grouped into a single row so that each actor or genre having the transaction ID belong to the same row.
2. Identifying cause of invalid rules. Initially passed the processed output file without removing the transaction ID which were giving incorrect rules.
3. The rules generated by the support and confidence values for our team were hard to interpret. A lot of the rules with very high confidence values were mostly movie genre with year. Only a few actor name showed up in the generated rules. Even so, the rules with actor names associated an actor with genre, or year, not another actor.