CSE506: Introduction to Data Mining
Assignment-2: Report

Nimisha Gupta (2019315)
Vani Sikka (2019395)

Assumption

- So the transaction array has rows as all the movieid and columns as userid. So each cell represents the rating a particular user id has given to a particular movie id.
- if a user has not given a rating to a particular movie in that case 0.0 rating is awarded to that movie for that particular user.
- Our rating threshold value is 2.5
- min_support= 0.075
- If for a given input no rule exist then movie 1 recommendations are written in output.csv

Q1) Exploratory Data Analysis

Exploratory data analysis is a way of analyzing any given datasets. Through this approach, we summarise the main characteristics of the datasets and highlight these features using data visualization methods such as graphs.
The given dataset describes 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org).

Importing the dataset values
We were given datasets, ‘movies.csv’, ‘ratings.csv’, ‘tags.csv’, ‘links.csv’ in the form of comma separated files. We imported these using the pandas.read_csv() function and converted them into dataframes.

Initial Analysis
The following commands were used:

1. pandas.DataFrame.head(n) - to view first n rows of the dataframe
2. pandas.DataFrame.describe() - gives basic statistical details of the table (count, mean, min, max, etc.)
3. pandas.DataFrame.shape - gives number of rows and columns of the dataframe
4. pandas.DataFrame.columns - shows the names of columns
5. pandas.DataFrame.isnull().sum() - returns the number of missing values in each column.
6. pandas.DataFrame.info() - prints a summary of the dataframe
7. pandas.DataFrame.pivot() - reshape a DataFrame organized by given column values

Further Analysis
We have done this in broadly 2 ways:

1. From a particular table
2. By merging 2 or more tables

1. From a particular table

a) Visualizing all user ratings of movie_id 1:

This is done by adding the condition 'ratings.movieId==1' as follows

```
new_data= ratings[ratings.movieId==1]
new_data.head(10)
```

b) Check the count of each rating given to movie_id 1:

This is done using the groupby() function on newly created 'new_data' dataframe

```
new_data.groupby('rating').size()
```

We plotted a histogram of the same.

c) Counting the number of NaN values in each column of each table:

NaN values are calculated using the following command:

```
DataFrame[column_name].isnull().sum()
    Eg: movies['movieId'].isnull().sum()
```

All columns of all tables had 0 NaN values except for 'tmdbId' of 'links'

Thus we printed the rows of 'links' which had NaN values using

```
links[links.isnull().any(axis=1)]
```

d) Checking duplicates in each row:

This was done using the following command:

```
DataFrame.duplicated().sum()
```

2. By merging two or more tables

a) I merged the 2 dataframes: movies and ratings:

This is done using the following command

```
merged= pd.merge(movies,ratings,on='movieId',how='inner')
```

The newly formed dataframe 'merged' was viewed using

```
merged.head(5)
```

Other commands like describe(), shape, columns, isnull().sum() were also used.

b) Checking number of ratings given to each movie:

This is done using the groupby() function on newly created 'merged' dataframe

```
merged.groupby(movieId).size()
```

c) Dataset merged by userId and plotted graphs:

Again groupby() function is used on 'merged' dataframe

```
ratings_by_user= merged.groupby('userId').agg([np.size, np.mean])
```

This will now allow us to view the size and mean of each column's values against each userId

A bar graph is plotted which shows the mean rating given by the first 40 users.

```
ratings_by_user['rating']['mean'].head(40).sort_values(ascending=False)
.plot(kind='bar',figsize=(10,5))
```

d) Graph for movies with rating more than 100:

This was made using the following command:

```
ratings_by_user[ratings_by_user['rating']['size']>100]['rating']['size']
.sort_values(ascending=True).head(40).plot(kind='bar',figsize=(11,5))
```

e) Plotting double bar graph:

```
_____ratings_by_user.head(25).plot(kind='bar',figsize=(8,5))
```

f) Movies with highest & lowest ratings:

For this the grouping had to be done by 'movieId'

```
temp= merged.groupby('movieId').agg([np.mean])
```

All the other columns other than rating were dropped.

```
temp= temp.drop('userId',axis=1)      [similarly other columns]
```

This temp when viewed in ascending order would give movies with lowest ratings

```
temp['rating']['mean'].sort_values(ascending=True).head(100)
```

This temp when viewed in descending order gives movies with highest ratings

```
temp['rating']['mean'].sort_values(ascending=False).head(100)
```

g) I merged the 2 dataframes: movies and tags:

The newly formed dataframe was called 'merged2'

Similar analysis was done on 'merged2'

h) Determine 10 Least and Most frequently used tags:

For this the grouping had to be done by 'tag'

```
temp= merged2.groupby('tag').agg([np.size])
```

All the other columns other than movieId were dropped.

```
temp= temp.drop('userId',axis=1)      [similarly other columns]
```

This temp when viewed in ascending order would give the least frequent tags

```
temp['movieId']['size'].sort_values(ascending=True).head(10)
```

This temp when viewed in descending order gives the most frequent tags

```
temp['movieId']['size'].sort_values(ascending=False).head(10)
```

A graph for the same was plotted.

Insights about the DataSet:

1. All csv files other than links does not have any NaN values. Links.csv also has only 8 NaN values. This indicates that even in such big datasets there are no missing or undefined values.
2. The dataset is arranged and demarcated in a very systematic way (One file for movieIds and titles, another for their ratings, third for the tags, fourth for links). This gives clarity and ease for the users to refer to any kind of information about any of the movies in the dataset.
3. There are no duplicate values in any of the 4 csv files.
4. The given dataset can be interpreted in many different ways so as to get the desired information. This means it will cater to the needs of a large number of people. Two or more of these files can be merged, studied and interpreted together.

Q2)Movie Recommendation System

Flow of Logic

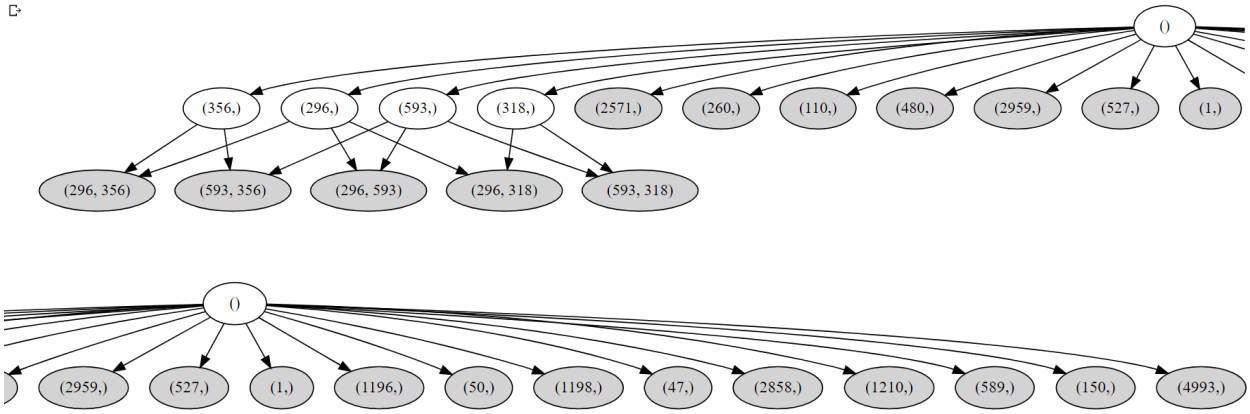
1. The datasets were imported by the method mentioned in the above section.
2. In order to make a transaction array we combined the movies data and ratings data using pivot function as explained in the initial analysis part. So the transaction array has rows as all the movieid and columns as userid. So each cell represents the rating a particular user id has given to a particular movie id.
3. Now if a user has not given a rating to a particular movie in that case 0.0 rating is awarded to that movie for that particular user.
4. Now in order to convert transaction data as an array of 0 and 1 , the code iterates to each cell of the transaction data to see if the rating is less than 2.5 then it awards 0 otherwise gives 1.
5. Now when we have converted that data frame into a matrix of 0 and 1 then we generate frequent movie item set,
6. From those frequent movie item set we then generate association rules using the respective python library
7. Once the rule is generated, then for every rule there are antecedents and consequences along with lift, confidence ,support and other values.
8. Now when we get a list of movies as input , the program first goes for complete prediction, that is it would select all those rules that have exactly the same antecedents as the input sets. From all the collected rules it would sort them by Decreasing order of their lift count.From that list top four movies are written as recommendations to output.csv
9. Let suppose the complete prediction does not give four recommendations then the program goes for complete partial predictions where it looks for rules that have antecedents which are a subset of the input provided .

Reference:

http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/fpgrowth/
https://www.kaggle.com/victorbonilla/apriori-recommender-system

Q3) Visualizing Maximal frequent pattern set

- Transaction matrix is generated by the same way mentioned in Q2
- Then frequent item set are generated from the transaction matrix
- Then those frequent itemsets are arranged according to their length. So all the item sets that are of size one are grouped together like this the program does for all the lengths till the maximum possible length
- Then In order to make visualize and identify maximal item sets, the program iterterates to the overall ith length itemsets and for each one it checks whether it is the subset of any of the sets of (i+1)th itemset [as len of i+1 th itemset is always bigger than ith items]
- If we find a subset then we make the superset as the parent of the subset in the graph
- If the Itemset at ith length could not find a super set in item sets of i+1 th length then that item set is maximal hence we identify that by changing the color of the node of that itemset.



Reference:
https://towardsdatascience.com/how-to-find-closed-and-maximal-frequent-itemsets-from-fp-growth-861a1ef13e21

Learnings:

- If the data set is big then it is important to make functions and reuse the code as since the data set provided was big and the rules generated were also of a big size , there were cases when program ran out of ram, in that case we had to optimize the code
- Main takeaway is how the frequent datasets got generated and they could get organised in a tree and gave us a deep and working understanding of how association rules get generated.
- We learnt how to interpret such a vast dataset which spread through and divided into multiple files. We learnt to interpret this dataset in many different ways so as to get the desired information. Two or more of these files can be merged, studied and interpreted together.
- Thinking of all the corner cases that the recommendation system would face also gave us a good learning practice.
- Interpreting data in the form of graphs makes it easier to understand and is sometimes more informative than textual and numeric data.