CSE506 - Data Mining

REPORT

Assignment 2

Submitted By
Vidhi Sharma, 2019286
Dolly Sidar, 2019304

QUESTION 1:

Step 1 - Imported the needed libraries and loaded the dataset
Step 2 - data analysis

Exploratory data analysis (EDA) of movies.csv:

Introductory details:

```
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   movieId  9742 non-null   int64 
 1   title    9742 non-null   object
 2   genres   9742 non-null   object
```

Statistical insight:

```
             movieId
count    9742.000000
mean    42200.353623
std     52160.494854
min         1.000000
25%      3248.250000
50%      7300.000000
75%     76232.000000
max    193609.000000
```

Number of Nan values per column:

```
movieId    0
title      0
genres     0
dtype: int64
```

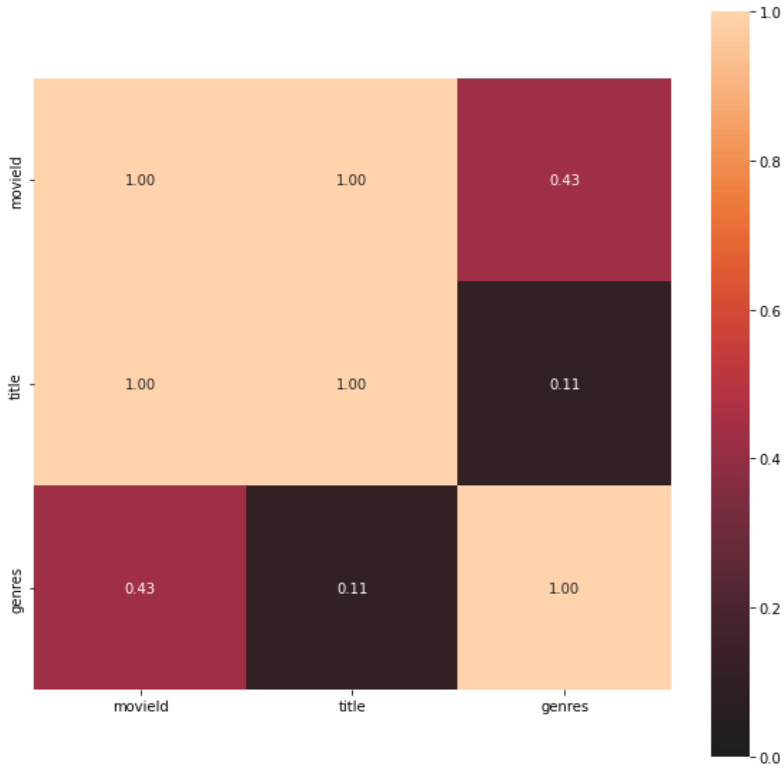
Finding frequently occurring values in categorical features:

```
5 most frequently occurring values in the title :
Emma (1996)                                 2
Saturn 3 (1980)                             2
War of the Worlds (2005)                    2
Confessions of a Dangerous Mind (2002)      2
Eros (2004)                                 2
```

```
5 most frequently occurring values in  genres :
Drama             1053
Comedy             946
```

```
Comedy|Drama        435
Comedy|Romance      363
Drama|Romance       349
```

Correlation Heatmap:



Exploratory data analysis (EDA) of ratings.csv :

Introductory details:

```
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100836 non-null  int64
 1   movieId    100836 non-null  int64
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
None
```

Statistical insight:

```
              userId        movieId         rating     timestamp
count  100836.000000  100836.000000  100836.000000  1.008360e+05
```

mean	326.127564	19435.295718	3.501557	1.205946e+09
std	182.618491	35530.987199	1.042529	2.162610e+08
min	1.000000	1.000000	0.500000	8.281246e+08
25%	177.000000	1199.000000	3.000000	1.019124e+09
50%	325.000000	2991.000000	3.500000	1.186087e+09
75%	477.000000	8122.000000	4.000000	1.435994e+09
max	610.000000	193609.000000	5.000000	1.537799e+09

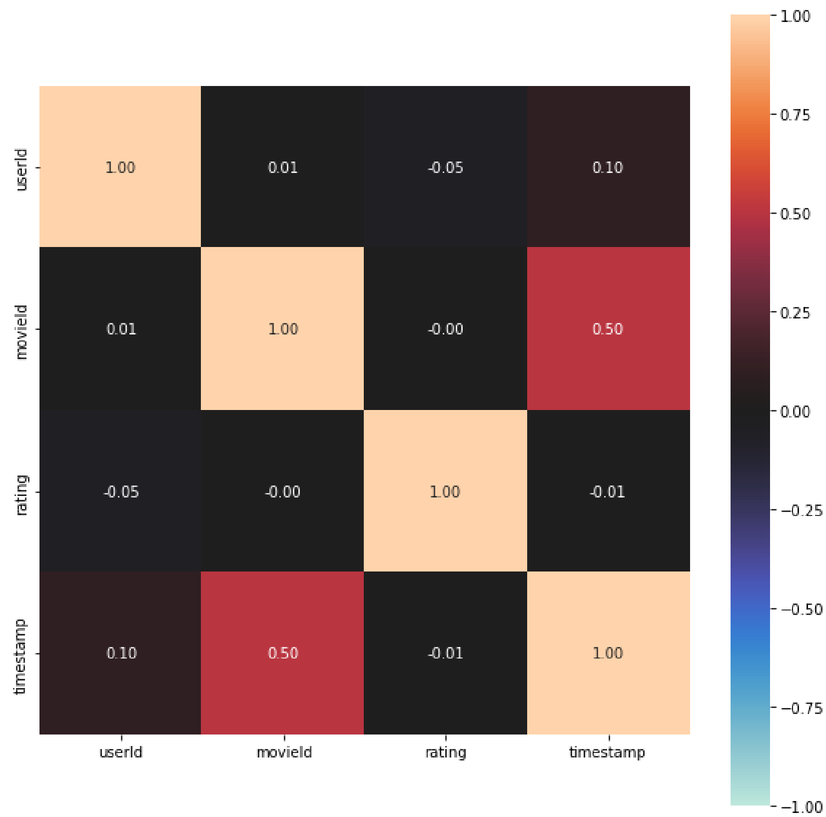
Number of Nan values per column:

```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

Finding frequently occurring values in categorical features:

```
No categorical Feature
```

Correlation Heatmap:



Exploratory data analysis (EDA) of tags.csv:

Introductory details:

```
RangeIndex: 3683 entries, 0 to 3682
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   userId     3683 non-null   int64 
 1   movieId    3683 non-null   int64 
 2   tag        3683 non-null   object
 3   timestamp  3683 non-null   int64 
```

Statistical insight:

```
 userId        movieId     timestamp
count  3683.000000    3683.000000  3.683000e+03
mean    431.149335   27252.013576  1.320032e+09
std     158.472553   43490.558803  1.721025e+08
min       2.000000       1.000000  1.137179e+09
25%     424.000000    1262.500000  1.137521e+09
50%     474.000000    4454.000000  1.269833e+09
75%     477.000000   39263.000000  1.498457e+09
max     610.000000  193565.000000  1.537099e+09
```

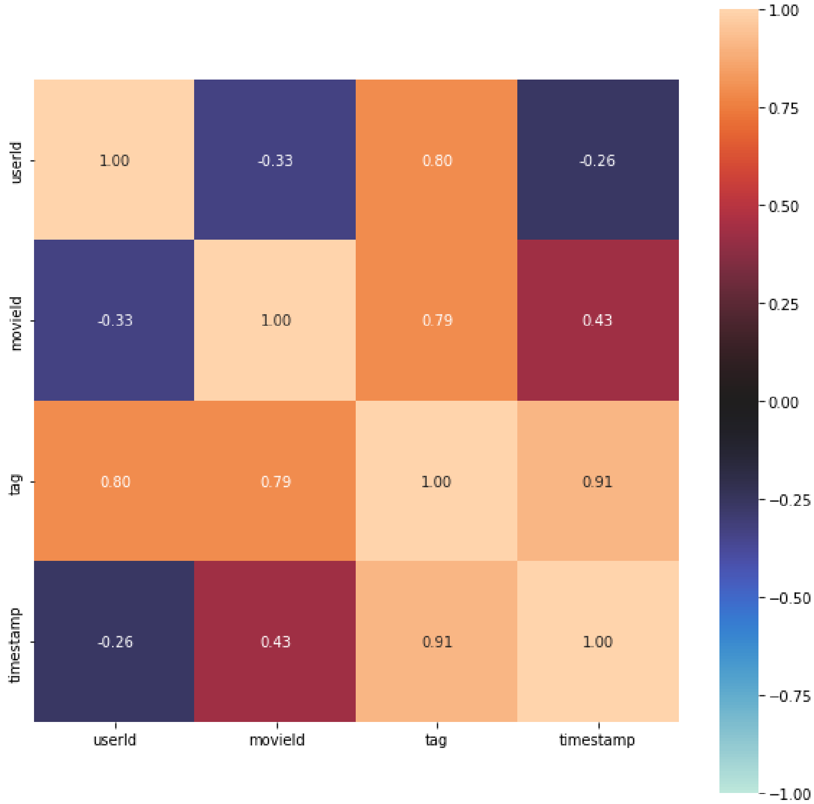
Number of Nan values per column:

```
userId       0
movieId      0
tag          0
timestamp    0
```

Finding frequently occurring values in categorical features:

```
5 most frequently occurring values in  tag :
In Netflix queue     131
atmospheric           36
superhero             24
thought-provoking     24
surreal               23
```

Correlation Heatmap:



Exploratory data analysis (EDA) of links.csv :

Introductory details:

```
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   imdbId   9742 non-null   int64
 2   tmdbId   9734 non-null   float64
```

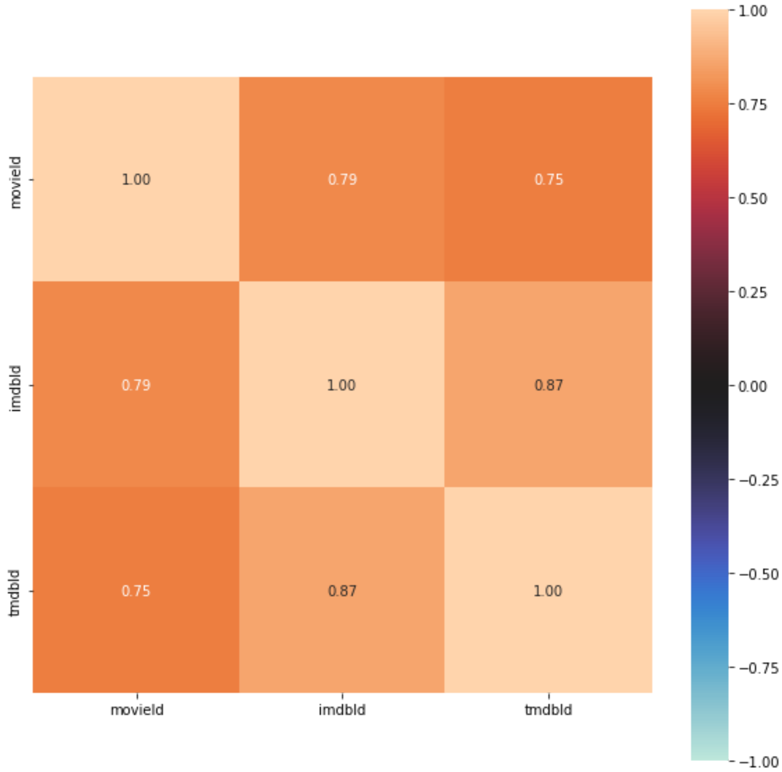
Statistical insight:

```
movieId         imdbId          tmdbId
count     9742.000000  9.742000e+03    9734.000000
mean     42200.353623  6.771839e+05   55162.123793
std      52160.494854  1.107228e+06   93653.481487
min          1.000000  4.170000e+02       2.000000
25%       3248.250000  9.518075e+04    9665.500000
50%       7300.000000  1.672605e+05   16529.000000
75%      76232.000000  8.055685e+05   44205.750000
max     193609.000000  8.391976e+06  525662.000000
```

Number of Nan values per column:

```
movieId     0
imdbId      0
tmdbId      8
```

Correlation Heatmap:



QUESTION 2:

Step 1 - merging data_ratings and data_movies
Reason - As Movies rating data contain more number of users as compared to tags data. Thus, it can be useful to be merged with the movie's title dataset to get more numbers of frequent items set implies with more number of rules and better recommendation.

Step 2 - dropping unnecessary columns like ratings, timestamp, genres to get a more clear view

Step 3 - preprocessing the dataset to create a transactional list in which every row represent a user and their selected movies (users count 610)

Step 4- encoding of data into binary using transactionEncoder

Step 5 - generating frequent itemsets using FPGrowth metric
Assumption- parameters min_support = 0.092 , max_len = None

Step 6 - applying association rule (number of rules generated around 96 lakhs)
Assumption- using Lift for making rules, threshold set = 1

Step 7 - recommending k movies based on the top lift values
Reason - the increasing value of lift is relatively proportional to other measures also such as confidence, support.

Assumption: For some movies, we don't have any recommendations as there are no rules generated for them because of less frequency, it is removed from association rules and frequent item formation. So for considering those cases we are using movies genres and ratings from the dataset to recommend movies.

```
Input Movie : 'American Movie (1999)', 'Cocaine Cowboys (2006)'
```

```
4 recommended movies are :  ['Little Dieter Needs to Fly (1997)', '9/11
(2002)', 'Koyaanisqatsi (a.k.a. Koyaanisqatsi: Life Out of Balance)
(1983)', 'Scratch (2001)']
```

```
Input Movie: 'Godfather, The (1999)'
```

```
4 recommended movies are :  ['Toy Story (1995)', 'Godfather: Part II, The
(1974)', 'Star Wars: Episode IV - A New Hope (1977)', 'Fugitive, The
(1993)']
```

```
Input Movie: "Forrest Gump (1994)", "Shawshank Redemption"
['Pulp Fiction (1994)', 'Lord of the Rings: The Return of the King, The (2003)', 'Godfather, The (1972)', 'Matrix, The (1999)']
```

QUESTION 3:

First type of Visualization Implemented:

Step 1 - fpmax() to find all maximal frequent pattern sets:

	support	itemsets	length
0	0.093443	(12 Angry Men (1957))	1
1	0.093443	(North by Northwest (1959))	1
2	0.093443	(Congo (1995))	1
3	0.093443	(Anchorman: The Legend of Ron Burgundy (2004))	1
4	0.093443	(Social Network, The (2010))	1
...	...	...	...
59503	0.098361	(Pulp Fiction (1994), Shawshank Redemption, Th...	6
59504	0.101639	(Pulp Fiction (1994), Shawshank Redemption, Th...	6
59505	0.095082	(Pulp Fiction (1994), Shawshank Redemption, Th...	6
59506	0.093443	(Pulp Fiction (1994), Jurassic Park (1993), Ma...	6
59507	0.095082	(Pulp Fiction (1994), Shawshank Redemption, Th...	7

59508 rows × 3 columns

Step 2 - implemented a function to create FP-Tree based on a given set of movies (This function will create FP-Tree visualization of the maximal frequent pattern set only for the movies given as input.)

Assumption -

- We are visualizing maximal frequent sets for some sets of the movies. This is because there are 2^(no of movies) sets and a large number of maximal frequent sets and visualizing all sets is not possible.

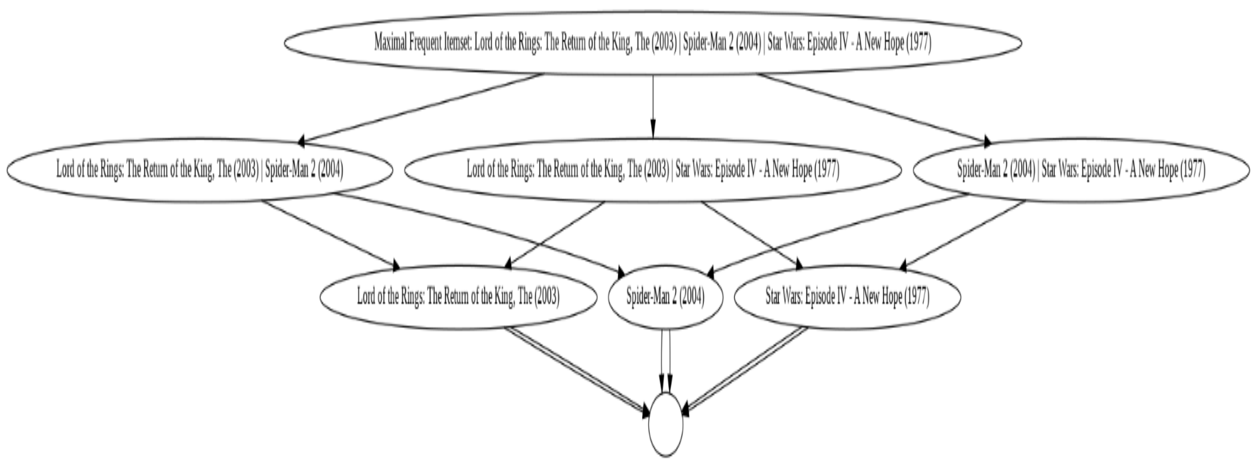
- Maximal Frequent Itemsets will be marked by “Maximal Frequent Itemsets:” before their name.

Visualization using FP-Tree:

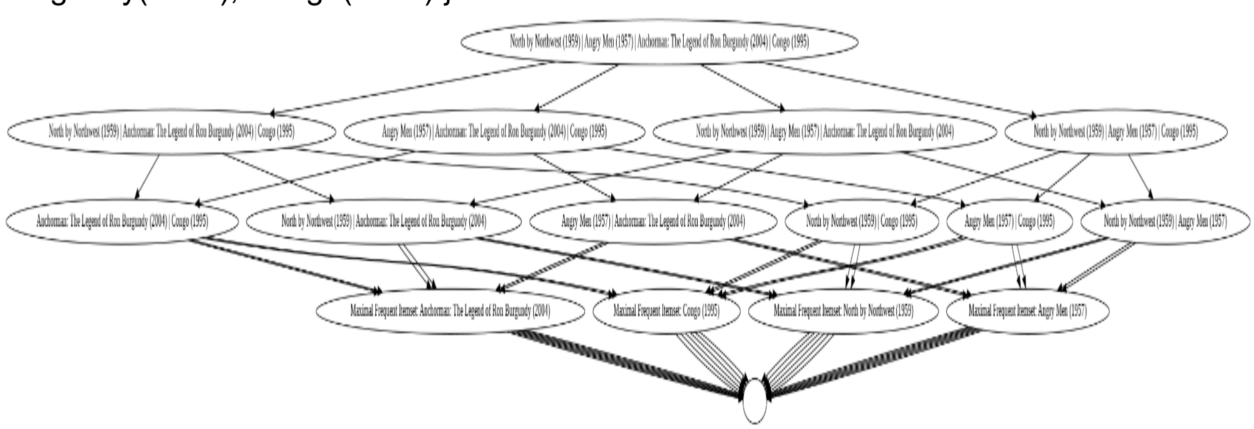
- Taken a set of movies as input (whose maximal frequent itemsets will be visualized).
- Visualization of the tree is done using anytree()
- Maximal Frequent Itemsets will be marked by “Maximal Frequent Itemsets:” before their name.

Some Visualizations:

Input: {'Lord of the Rings: The Return of the King, The (2003)',
'Spider-Man 2 (2004)',
'Star Wars: Episode IV - A New Hope (1977)'}



Input: {North by Northwest(1959), Angry Men(1957), Anchorman: The Legend of Ron Burgundy(2004), Congo(1995) }



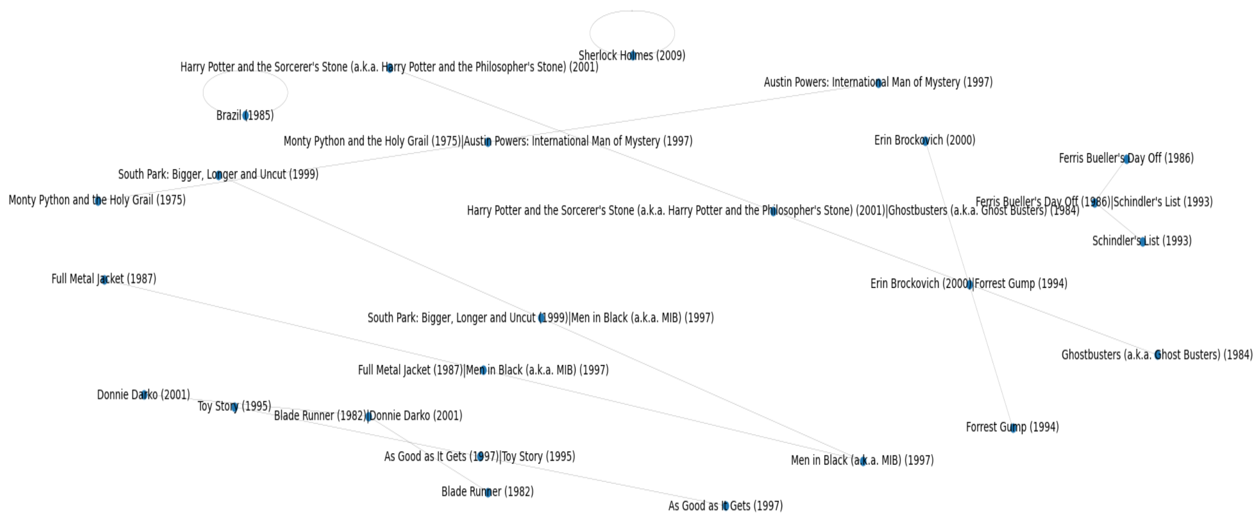
Second type of Visualization Implemented:

Step 1 - fpmax() to find all maximal frequent pattern sets:

Visualization using Networx:

- Selected random 10 maximal frequent itemsets.
- Visualized them along with their subsets.

Visualization Eg:



Learning:

1. Learned different EDA analysis techniques.
2. Learned to visualize correlation heat map of a dataset having categorical values using associations() function from python.nominal library.
3. Learned about Apriori, and FP growth metrics to make rules for association rule algorithm
4. Learned about association rule measure techniques such as confidence, support, and lift.
5. Learned to visualize FP growth trees.

References:

https://medium.com/@jwu2/content-based-recommender-systems-and-association-rules-599843cb2fd9

https://www.youtube.com/watch?v=CIoMnbD6Bc4&t=0s&ab_channel=PonshankarPalanivel

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