DATA MINING CSE 506
Assignment 2

Harshal Dev (2019306)
Sneh Suman (2019337)

Visualizations -
Sol 1- EDA
Exploratory Data Analysis is an approach in analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.
Here we will look at some of the figures which gives a clear vision of data we have been presented with.

```
[ ]  ratings.isnull().sum()

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

```
[ ]  movies.isnull().sum()

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

```
[ ]  tags.isnull().sum()

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

Q1

Double-click (or enter) to edit

[9]

```python
unique_user = ratings.userId.nunique(dropna = True)
unique_movie = ratings.movieId.nunique(dropna = True)
print("number of unique users in the dataset:")
print(unique_user)
print("number of unique movies in the dataset:")
print(unique_movie)
```

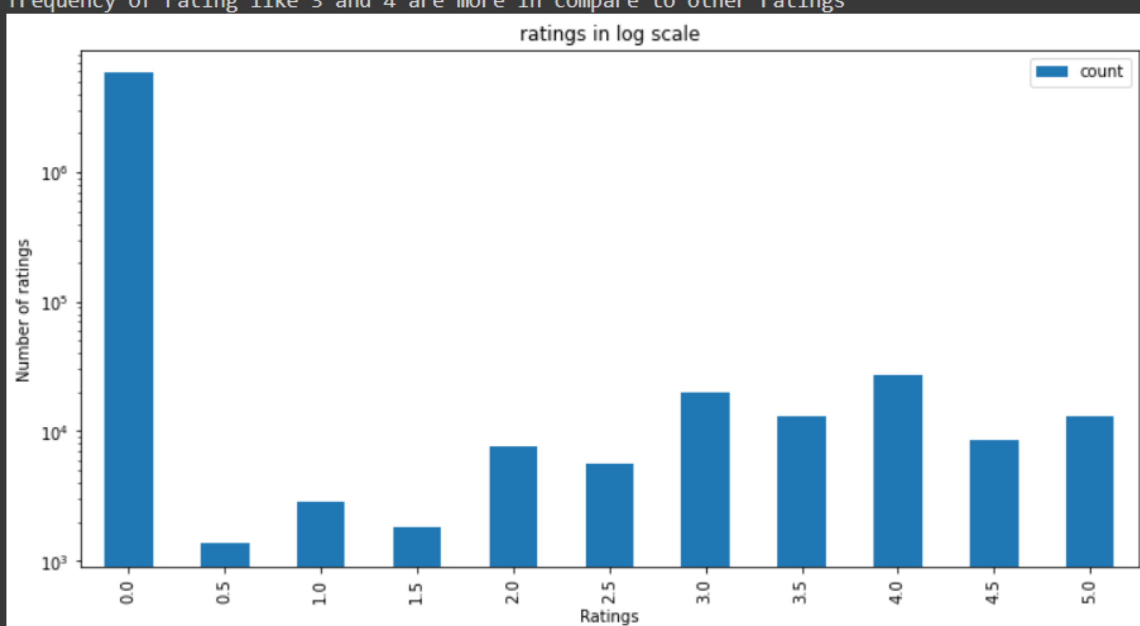
```
number of unique users in the dataset:
610
number of unique movies in the dataset:
9724
```

[10]

```python
ratings_total = unique_user*unique_movie
ratings_available = ratings.shape[0]
ratings_wedonthave = ratings_total - ratings_available
print("ratings not provided means some user have not watched some movies and its given by")
print(ratings_wedonthave)
```

```
ratings not provided means some user have not watched some movies and its given by
5830804
```

```
frequency of rating like 3 and 4 are more in compare to other ratings
```

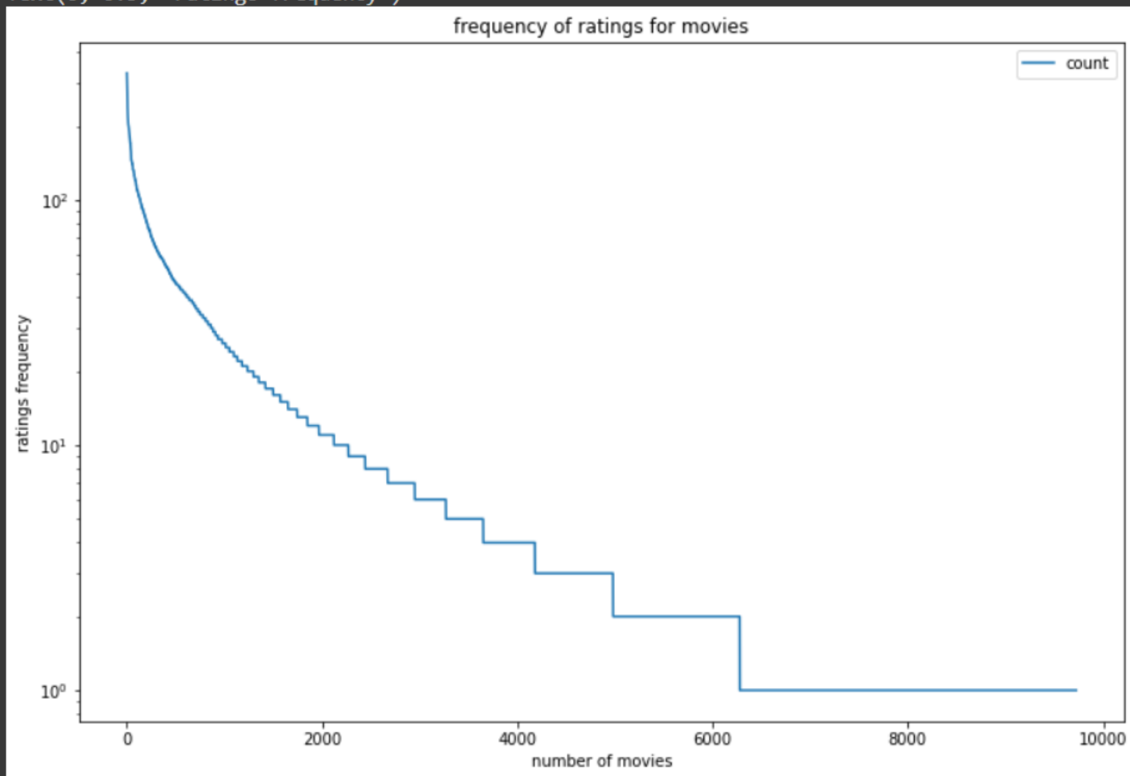


[12]

```
# Number of ratings for each movie
f=(12,8)
gh=(12,6)
movie_freq = pd.DataFrame(ratings.groupby('movieId').size(),columns=['count'])
movie_freq.head(15)
```

movieId	count
1	215
2	110
3	52
4	7
5	49
6	102
7	54
8	8
9	16
10	132
11	70
12	19
13	8
14	18
15	13

```
Text(0, 0.5, 'ratings frequency')
```



[14]

```
g=9
import math
#Movies with less than a cent reviews are ignored
popular_movies_id = list(set(movie_freq.query('count>=@threshold_rating_freq').index))
ratings_excep=math.log(15,10)
# ratings df after dropping non popular movies
ratings_with_popular_movies = ratings[ratings.movieId.isin(popular_movies_id)]
# ratings details
print('shape of ratings:')
print(ratings.shape)
ratings_excep=math.log(15,10)
print('shape of ratings_with_popular_movies:')
print(ratings_with_popular_movies.shape)
ratings_excep=math.log(15,10)
print("no of movies which are rated more than 50 times:")
print(len(popular_movies_id))
print("no of unique movies present in dataset:")
print(unique_movie)
```

```
shape of ratings:
(100836, 4)
shape of ratings_with_popular_movies:
(81116, 4)
no of movies which are rated more than 50 times:
2269
no of unique movies present in dataset:
9724
```

Sol 2 - Association Rule Mining

Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness.

Here we have a file input.tsv file having a list of movies which we are required to give a recommendation for.

In the figure shown below we can see that we have a movie name and corresponding four values are the recommended movies . We have also saved the data in the output.csv file.

```
                                   rating_relative
title
Aladdin (1992)                            1.000000
Beauty and the Beast (1991)               0.651768
Lion King, The (1994)                     0.598575
Die Hard: With a Vengeance (1995)         0.423469
True Lies (1994)                          0.413527
<class 'pandas.core.frame.DataFrame'>
----------------------------
                                   rating_relative
title
True Lies (1994)                          1.000000
Batman (1989)                             0.534785
Speed (1994)                              0.525890
Die Hard: With a Vengeance (1995)         0.524531
Cliffhanger (1993)                        0.518631
<class 'pandas.core.frame.DataFrame'>
----------------------------
                             rating_relative
title
Lion King, The (1994)                1.000000
Aladdin (1992)                       0.598575
Beauty and the Beast (1991)          0.593200
Mrs. Doubtfire (1993)                0.485212
Mask, The (1994)                     0.467683
<class 'pandas.core.frame.DataFrame'>
----------------------------
                                   rating_relative
title
Die Hard: With a Vengeance (1995)         1.000000
True Lies (1994)                          0.524531
Cliffhanger (1993)                        0.500704
Speed (1994)                              0.491706
GoldenEye (1995)                          0.470082
<class 'pandas.core.frame.DataFrame'>
----------------------------
```

Sol 3 - A maximal frequent itemset is a frequent itemset for which none of its immediate supersets are frequent.

For frequent pattern growth tree visualization we have first taken the list of movies and then set the support as 2 . Now when we input a movie. We see a fp-growth tree from the fp-growth table for the movie which we have created for support =2 and ignore less than 2 support valued movies.

We can see an example in the following image-

[30]
```
dd=(50,30)
te = TransactionEncoder()
te_ary = te.fit(merge_list).transform(merge_list)
df = pd.DataFrame(te_ary, columns=te.columns_)
df.head()
```

	(500) Days of Summer (2009)	...And Justice for All (1979)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	11'09"01 - September 11 (2002)	12 Angry Men (1957)	127 Hours (2010)	13 Going on 30 (2004)	2001: A Space Odyssey (1968)	21 Grams (2003)	25th Hour (2002)	28 Days Later (2002)	39 Steps, The (1935)	3:10 to Yuma (2007)	40-Year-Old Virgin, The (2005)	400 Blows, The (Les quatre cents coups) (1959)	42 Up (1998)	84 Charing Cross Road (1987)	(19
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F

5 rows × 1572 columns

[31]
```
from mlxtend.frequent_patterns import fpgrowth
fpgrowth_frequent_itemsets = fpgrowth(df, min_support=lk, use_colnames=cb,max_len=ll)
fpgrowth_frequent_itemsets.head()
```

	support	itemsets
0	0.051724	(Step Brothers (2008))
1	0.034483	(Wolf of Wall Street, The (2013))
2	0.017241	(Warrior (2011))
3	0.051724	(Departed, The (2006))
4	0.034483	(Godfather: Part II, The (1974))

[32]
```
fpgrowth_frequent_itemsets['itemsets'].apply(lambda x: len(x)).value_counts()
```

```
2    774986
1      1572
Name: itemsets, dtype: int64
```

[33]

```
fpgrowth_frequent_itemsets['length'] = fpgrowth_frequent_itemsets['itemsets'].apply(lambda x: len(x))
fpgrowth_frequent_itemsets
```

	support	itemsets	length
0	0.051724	(Step Brothers (2008))	1
1	0.034483	(Wolf of Wall Street, The (2013))	1
2	0.017241	(Warrior (2011))	1
3	0.051724	(Departed, The (2006))	1
4	0.034483	(Godfather: Part II, The (1974))	1
...	...	...	...
776553	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776554	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776555	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776556	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776557	0.017241	(Hard-Boiled (Lat sau san taam) (1992), John W...	2

776558 rows × 3 columns

[34]

```
fpgrowth_frequent_itemsets[(fpgrowth_frequent_itemsets['length'] > dx)
                           & (fpgrowth_frequent_itemsets['support'] > dzs)].head()
```

support	itemsets	length

[35]

```
fpgrowth_frequent_itemsets[(fpgrowth_frequent_itemsets['length'] != dx)]
```

	support	itemsets	length
1572	0.034483	(Step Brothers (2008), Anchorman: The Legend o...	2
1573	0.017241	(Step Brothers (2008), Corpse Bride (2005))	2
1574	0.017241	(City of God (Cidade de Deus) (2002), Step Bro...	2
1575	0.017241	(Departed, The (2006), Step Brothers (2008))	2
1576	0.017241	(Terminator 2: Judgment Day (1991), Step Broth...	2
...	...	...	...
776553	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776554	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776555	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776556	0.017241	(Night of the Shooting Stars (Notte di San Lor...	2
776557	0.017241	(Hard-Boiled (Lat sau san taam) (1992), John W...	2

774986 rows × 3 columns

[36]
```python
fpgrowth_frequent_itemsets[fpgrowth_frequent_itemsets['itemsets'].apply(lambda x: 'Aladdin (1992)' in str(x))]
```

	support	itemsets	length
616	0.017241	(Aladdin (1992))	1
100423	0.017241	(Airplane! (1980), Aladdin (1992))	2
100424	0.017241	(Aladdin (1992), Air Force One (1997))	2
100425	0.017241	(Aladdin (1992), Age of Innocence, The (1993))	2
100426	0.017241	(After the Thin Man (1936), Aladdin (1992))	2
...	...	...	...
763526	0.017241	(Aladdin (1992), Paper Clips (2004))	2
764757	0.017241	(Paper Moon (1973), Aladdin (1992))	2
765989	0.017241	(Aladdin (1992), Paper, The (1994))	2
767222	0.017241	(Aladdin (1992), Paradise Lost: The Child Murd...	2
768456	0.017241	(Aladdin (1992), Parallax View, The (1974))	2

1235 rows × 3 columns

[37]
```python
xcz='antecedents'
rules = association_rules(fpgrowth_frequent_itemsets,metric="lift",min_threshold=(1k*3))
```

[38]
```python
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Step Brothers (2008))	(Anchorman: The Legend of Ron Burgundy (2004))	0.051724	0.068966	0.034483	0.666667	9.666667	0.030916	2.793103
1	(Anchorman: The Legend of Ron Burgundy (2004))	(Step Brothers (2008))	0.068966	0.051724	0.034483	0.500000	9.666667	0.030916	1.896552
2	(Step Brothers (2008))	(Corpse Bride (2005))	0.051724	0.051724	0.017241	0.333333	6.444444	0.014566	1.422414
3	(Corpse Bride (2005))	(Step Brothers (2008))	0.051724	0.051724	0.017241	0.333333	6.444444	0.014566	1.422414
4	(City of God (Cidade de Deus) (2002))	(Step Brothers (2008))	0.051724	0.051724	0.017241	0.333333	6.444444	0.014566	1.422414
...	...	...	...	...	...	...	...	...	...
1549967	(Gladiator (2000))	(Night of the Shooting Stars (Notte di San Lor...	0.034483	0.017241	0.017241	0.500000	29.000000	0.016647	1.965517
1549968	(Night of the Shooting Stars (Notte di San Lor...	(I'm Not Scared (Io non ho paura) (2003))	0.017241	0.034483	0.017241	1.000000	29.000000	0.016647	inf
1549969	(I'm Not Scared (Io non ho paura) (2003))	(Night of the Shooting Stars (Notte di San Lor...	0.034483	0.017241	0.017241	0.500000	29.000000	0.016647	1.965517
1549970	(Hard-Boiled (Lat sau san taam) (1992))	(John Wick: Chapter Two (2017))	0.017241	0.034483	0.017241	1.000000	29.000000	0.016647	inf
1549971	(John Wick: Chapter Two (2017))	(Hard-Boiled (Lat sau san taam) (1992))	0.034483	0.017241	0.017241	0.500000	29.000000	0.016647	1.965517

1549972 rows × 9 columns

[39]
```python
rules[rules[xcz].apply(lambda x: "Aladdin (1992)" in str(x))].sort_values(ascending=False,by='lift')
```

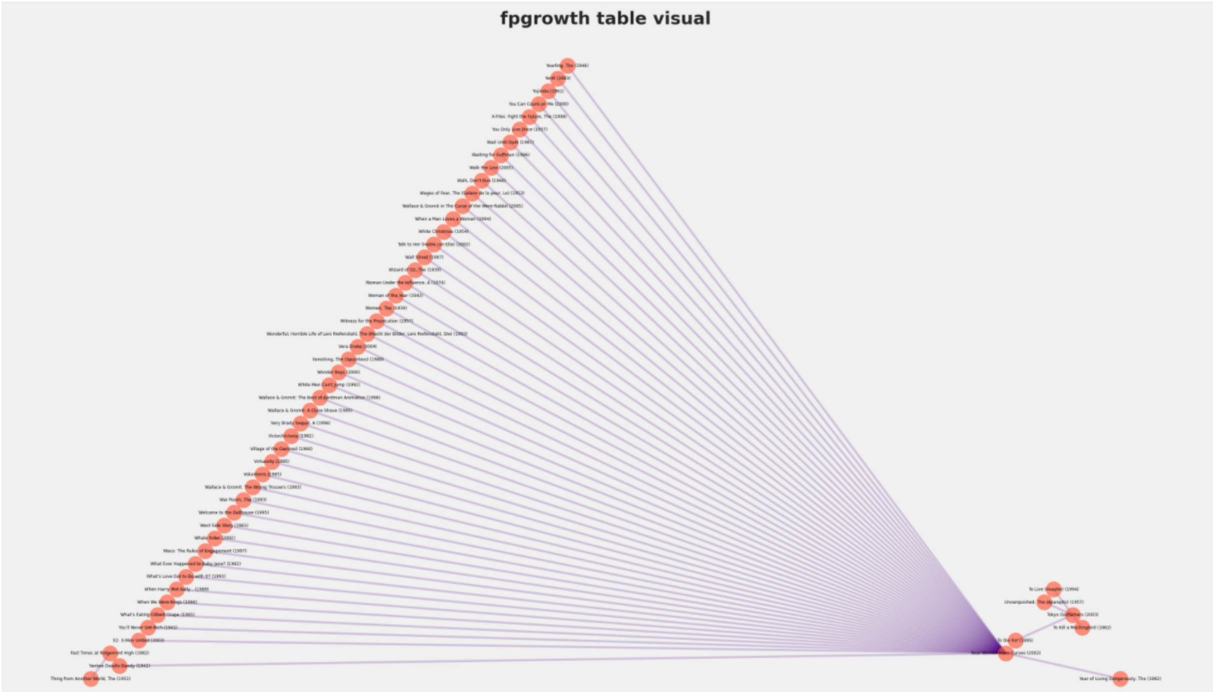
```python
rules[rules[xcz].apply(lambda x: "Aladdin (1992)" in str(x))].sort_values(ascending=False,by='lift')

```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
197703	(Aladdin (1992))	(Airplane! (1980))	0.017241	0.017241	0.017241	1.0	58.0	0.016944	inf
684640	(Aladdin (1992))	(Waiting for Guffman (1996))	0.017241	0.017241	0.017241	1.0	58.0	0.016944	inf
696191	(Aladdin (1992))	(Talk to Her (Hable con Ella) (2002))	0.017241	0.017241	0.017241	1.0	58.0	0.016944	inf
694534	(Aladdin (1992))	(White Christmas (1954))	0.017241	0.017241	0.017241	1.0	58.0	0.016944	inf
692880	(Aladdin (1992))	(When a Man Loves a Woman (1994))	0.017241	0.017241	0.017241	1.0	58.0	0.016944	inf
...	...	...	...	...	...	...	...	...	...
198558	(Aladdin (1992))	(Donnie Darko (2001))	0.017241	0.086207	0.017241	1.0	11.6	0.015755	inf
198561	(Aladdin (1992))	(Dr. Strangelove or: How I Learned to Stop Wor...	0.017241	0.086207	0.017241	1.0	11.6	0.015755	inf
198562	(Aladdin (1992))	(Memento (2000))	0.017241	0.086207	0.017241	1.0	11.6	0.015755	inf
198565	(Aladdin (1992))	(Eternal Sunshine of the Spotless Mind (2004))	0.017241	0.086207	0.017241	1.0	11.6	0.015755	inf
198567	(Aladdin (1992))	(Star Wars: Episode IV - A New Hope (1977))	0.017241	0.172414	0.017241	1.0	5.8	0.014269	inf

1234 rows × 9 columns

Microsoft Edge



Assumptions -

1. There are a lot of movies which were not rated by the users. Since, we have to use ratings to do the recommendation . We will make all those empty ratings as 0. This will help in calculating the movies in a better manner.
2. There are a lot of movies which are rated rarely by the users. For our example we have ignored movies with less than 50 ratings from users.
3. After getting ratings we have tried to keep the recommendation till rating 3 and ignored less than 3 rating movies.
4. There are tags for each movie and also timestamps . Since, we are not using them and they are not very much required for the recommendation process. We have dropped them from our dataset.
5. The recommendations are based on relative ratings. It means when we input one movie . The system checks for other movies which are rated by the common users. Then a relative value is calculated . And we get the best 4 commonly rated movies as our solution.
6. When we do the fp-growth. We have kept a support of 2 for movies to be considered for frequent patterns . These patterns are also made with the help of ratings given to the movies.
7. When we have a table showing fp-growth . We have used it to visualize an image of the fp-tree will look like.

Learnings -

From 1-

i) How EDA works for a dataset.

ii) How we can conclude important details from datasets for future use.
From 2-
i) Learned to use association rules in real life examples.
ii) Learnt about how a recommendation system works.
iii) Learned about using apriori algorithm.
From 3-
i) Using frequent pattern growth for real life examples.
ii) Visualizing a fp-growth tree in python.

References -

1.https://github.com/MCoffey1129/Recommender_Systems/blob/master/Recommendation%20models.py
2.https://levelup.gitconnected.com/create-a-recommendation-system-in-python-d7a95b2837ab
3.https://www.kaggle.com/ahm6644/movies-recommendations-by-association-rules
4.https://github.com/himeshmehta/movie-recommender-system-using-KNN/blob/master/movie_recommender_system_using_KNN.ipynb
5.https://github.com/govegito/Mrec-system/blob/master/movie%20recommendation%20using%20apriori.py