DMG Assignment-2

Q1) Exploratory Data Analysis

1) finding frequently occurring values in categorical features

- In movies dataset

  - In the **genres** column, the frequently occurring value is **Drama** with a count of 1053.

```
Drama                                          1053
Comedy                                          946
Comedy|Drama                                    435
Comedy|Romance                                  363
Drama|Romance                                   349
                                               ...
Adventure|Animation|Children|Comedy|Drama|Fantasy    1
Adventure|Children|Fantasy|Sci-Fi|Thriller           1
Drama|Horror|Romance                                 1
Adventure|Comedy|Crime|Thriller                      1
Action|Comedy|Crime|Mystery                          1
Name: genres, Length: 951, dtype: int64
```

  - In the **title** column, the frequently occurring categorical value is **Saturn 3 (1980),Eros (2004),Emma (1996),Confessions of a Dangerous Mind (2002) and War of the Worlds (2005) with a count of 2.**

```
Saturn 3 (1980)                                          2
Eros (2004)                                              2
Emma (1996)                                              2
Confessions of a Dangerous Mind (2002)                   2
War of the Worlds (2005)                                 2
                                                        ..
Halloween 5: The Revenge of Michael Myers (1989)         1
King Kong (2005)                                         1
Name of the Rose, The (Name der Rose, Der) (1986)        1
Teahouse of the August Moon, The (1956)                  1
Expendables, The (2010)                                  1
Name: title, Length: 9737, dtype: int64


0    The most frequent one is Confessions of a Dang...
1                The most frequent one is Emma (1996)
2                The most frequent one is Eros (2004)
3            The most frequent one is Saturn 3 (1980)
4    The most frequent one is War of the Worlds (2005)
dtype: object
```

- In tags dataset
  - In the **tag** column, the frequently occurring categorical value is **Netflix queue.**

```
In Netflix queue     131
atmospheric           36
thought-provoking     24
superhero             24
funny                 23
                    ...
Scifi masterpiece      1
robbery                1
cool                   1
tension building       1
Sean Connery           1
Name: tag, Length: 1589, dtype: int64


0    The most frequent one is In Netflix queue
dtype: object
```

- In rating dataset
  - In the **rating** column, the frequently occurring value is **4.0**.

```
4.0     26818
3.0     20047
5.0     13211
3.5     13136
4.5      8551
2.0      7551
2.5      5550
1.0      2811
1.5      1791
0.5      1370
Name: rating, dtype: int64


The most frequent one in is 0    4.0
dtype: float64
```

➢

2) Counting of Nan values per column

- ❖ In movies dataset
    - ➢ Number of Nan values in the column movieId is 0.
    - ➢ Number of Nan values in the column title is 0.
    - ➢ Number of Nan values in the column genres is 0.

- ❖ In links dataset
    - ➢ Number of Nan values in the column movieId is 0.
    - ➢ Number of Nan values in the column imdbId is 0.
    - ➢ Number of Nan values in the column tmdbId is 8.

- ❖ In ratings dataset
    - ➢ Number of Nan values in the column userId is 0.
    - ➢ Number of Nan values in the column movieId is 0.
    - ➢ Number of Nan values in the column rating is 0.
    - ➢ Number of Nan values in the column timestamp is 0.

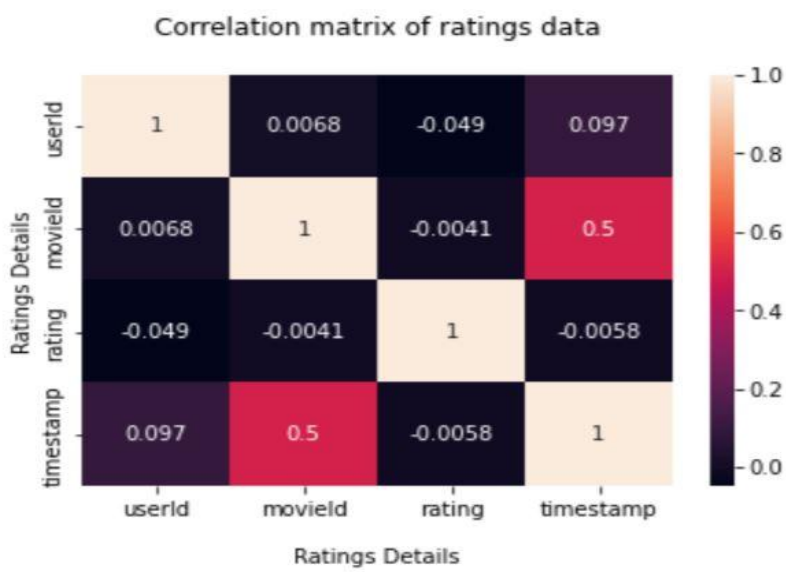
- ❖ In tags dataset
    - ➢ Number of Nan values in the column userId is 0.
    - ➢ Number of Nan values in the column movieId is 0.
    - ➢ Number of Nan values in the column tag is 0.
    - ➢ Number of Nan values in the column timestamp is 0.

3)Plotting correlations between features

- ❖ Correlation matrix of movies data:

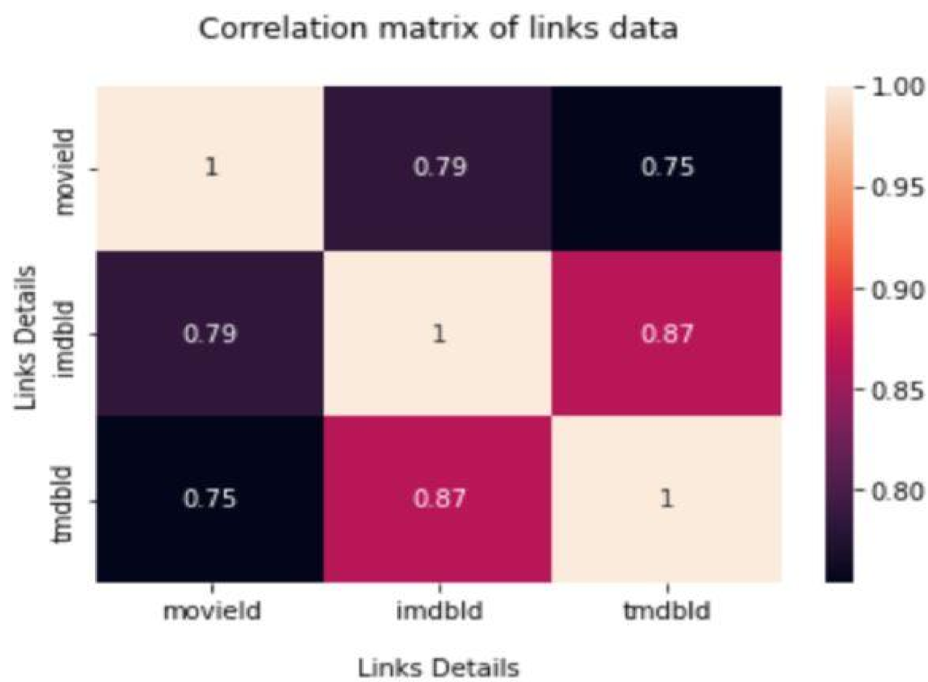
➢

❖ Correlation matrix of ratings data:



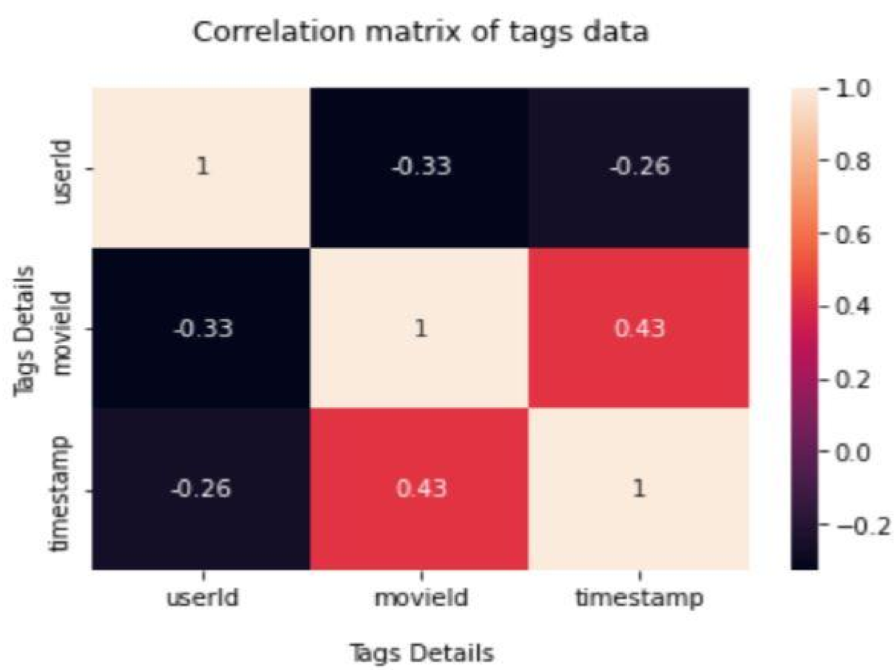
➢

❖ Correlation matrix of links data:



➢

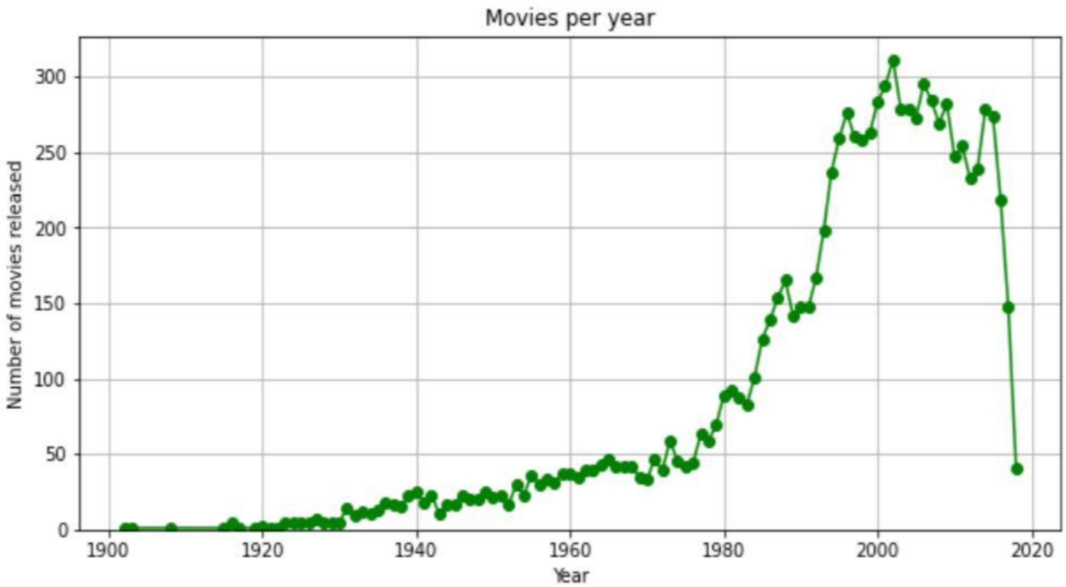
❖ Correlation matrix of tags data:



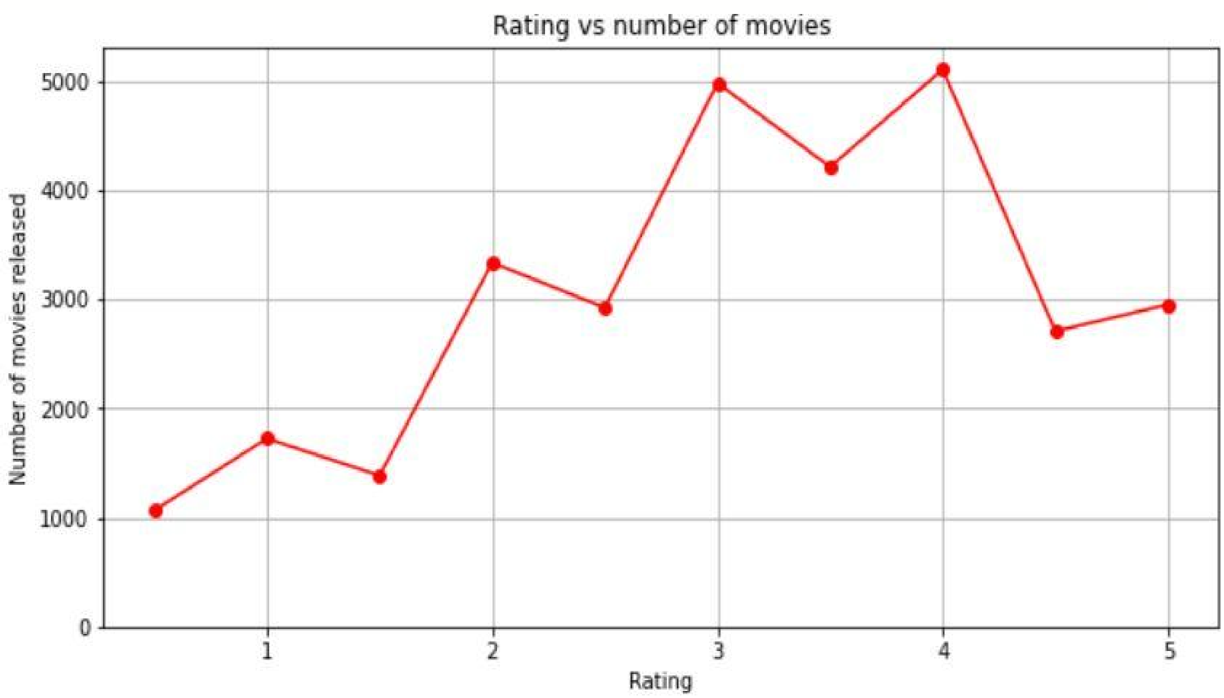
➢

4)Three insights about the dataset

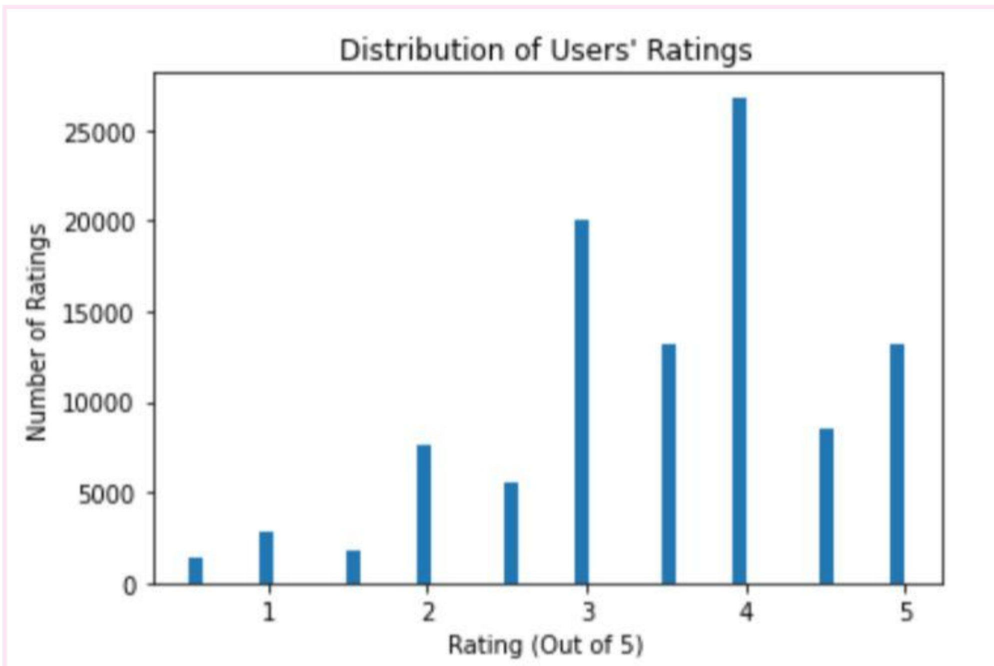
- We plotted a graph between the year and number of movies released on the movies dataset.



- From the above plot, we made an insight that **more number of movies were released in the year around 2002**.
- **Number of movies released per year increased exponentially around until 1995, then dropped significantly in 2014**.
- We plotted a graph between ratings and number of movies on the ratings dataset.



- From the above plot, we made an insight that the **rating 4 is given to most movies which indicates that most of the movies are good according to viewers**.
- We can also observe that as **a particular movie gets more ratings,then it is more likely to also have a higher rating since more people watch a movie if they hear that the movie is good and thus more number of ratings and more number of viewers for that movie and they give a good rating**.
- By the consequence of above point, we can also make an insight that **1 rating is low because not many people went to watch bad films**.



- From the above plot,we can observe that **most of the people watch the movie that is good and thus 4 ratings are more**.
- We can also observe that **ratings of 1 and 5 are low since they are rare cases ,i.e, outliers and such movies which are very bad or very good are low in number**.

Q2) Recommendation system using association rule mining

Preparing dataframe for association rule mining:

- Merge movies and tags on column movie id using inner join

	movieId	title	genres	userId	tag	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	336	pixar	1139045764
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	474	pixar	1137206825
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	567	fun	1525286013
3	2	Jumanji (1995)	Adventure|Children|Fantasy	62	fantasy	1528843929
4	2	Jumanji (1995)	Adventure|Children|Fantasy	62	magic board game	1528843932
...	...	...	...	...	...	...
3678	187595	Solo: A Star Wars Story (2018)	Action|Adventure|Children|Sci-Fi	62	star wars	1528934552
3679	193565	Gintama: The Movie (2010)	Action|Animation|Comedy|Sci-Fi	184	anime	1537098582
3680	193565	Gintama: The Movie (2010)	Action|Animation|Comedy|Sci-Fi	184	comedy	1537098587
3681	193565	Gintama: The Movie (2010)	Action|Animation|Comedy|Sci-Fi	184	gintama	1537098603
3682	193565	Gintama: The Movie (2010)	Action|Animation|Comedy|Sci-Fi	184	remaster	1537098592

3683 rows × 6 columns

- Dropping tags, timestamp and genres from the merged dataframe

	movieId	title	userId
0	1	Toy Story (1995)	336
1	1	Toy Story (1995)	474
2	1	Toy Story (1995)	567
3	2	Jumanji (1995)	62
4	2	Jumanji (1995)	62
...	...	...	...
3678	187595	Solo: A Star Wars Story (2018)	62
3679	193565	Gintama: The Movie (2010)	184
3680	193565	Gintama: The Movie (2010)	184
3681	193565	Gintama: The Movie (2010)	184
3682	193565	Gintama: The Movie (2010)	184

3683 rows × 3 columns

- Merge movies and ratings dataframe on movie id with inner join

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1	4.0	964982703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	847434962
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	4.5	1106635946
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.5	1510577970
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	17	4.5	1305696483
...	...	...	...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy	184	4.0	1537109082
100832	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy	184	3.5	1537109545
100833	193585	Flint (2017)	Drama	184	3.5	1537109805
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation	184	3.5	1537110021
100835	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy	331	4.0	1537157606

100836 rows × 6 columns

- From the above merged dataframe drop rating, timestamp and genres column

	movieId	title	userId
0	1	Toy Story (1995)	1
1	1	Toy Story (1995)	5
2	1	Toy Story (1995)	7
3	1	Toy Story (1995)	15
4	1	Toy Story (1995)	17
...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	184
100832	193583	No Game No Life: Zero (2017)	184
100833	193585	Flint (2017)	184
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	184
100835	193609	Andrew Dice Clay: Dice Rules (1991)	331

100836 rows × 3 columns

- Concatinating the above two merged dataframes and removing the duplicates

	movieId	title	userId
0	1	Toy Story (1995)	336
1	1	Toy Story (1995)	474
2	1	Toy Story (1995)	567
3	2	Jumanji (1995)	62
4	2	Jumanji (1995)	62
...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	184
100832	193583	No Game No Life: Zero (2017)	184
100833	193585	Flint (2017)	184
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	184
100835	193609	Andrew Dice Clay: Dice Rules (1991)	331

104519 rows × 3 columns

	movieId	title	userId
0	1	Toy Story (1995)	336
1	1	Toy Story (1995)	474
2	1	Toy Story (1995)	567
3	2	Jumanji (1995)	62
6	2	Jumanji (1995)	474
...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	184
100832	193583	No Game No Life: Zero (2017)	184
100833	193585	Flint (2017)	184
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	184
100835	193609	Andrew Dice Clay: Dice Rules (1991)	331

100972 rows × 3 columns

- Group by user id according to titles of movies watched

	userId	title
0	1	[Toy Story (1995), Grumpier Old Men (1995), He...
1	2	[Step Brothers (2008), Warrior (2011), Wolf of...
2	3	[Dangerous Minds (1995), Schindler's List (199...
3	4	[Get Shorty (1995), Twelve Monkeys (a.k.a. 12 ...
4	5	[Toy Story (1995), Get Shorty (1995), Babe (19...

Data Transformation with transaction encoder:

- Data is transformed to binary input to get accepted by the algorithm using Transaction Encoder

	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]^2 (2009)	[REC]^3 3 Génesis (2012)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)	xXx (2002)
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
605	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
606	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
607	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	True	True
608	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
609	True	False	False	False	False	False	False	False	True	False	...	False	True	True	True	False	False	True

610 rows × 9737 columns

Generating frequent itemsets using fp growth:

- Now, generate itemsets according to given parameters min support = 0.09. The results of frequent itemsets are python frozensets

	support	itemsets
0	0.539344	(Forrest Gump (1994))
1	0.503279	(Pulp Fiction (1994))
2	0.457377	(Silence of the Lambs, The (1991))
3	0.455738	(Matrix, The (1999))
4	0.416393	(Star Wars: Episode IV - A New Hope (1977))

Association rules:

- Using mlxtend module find rules. Here we are using lift as the score evaluation metric.

```
[46] rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(Pulp Fiction (1994))	(Forrest Gump (1994))	0.503279	0.539344	0.377049	0.749186	1.389068
1	(Forrest Gump (1994))	(Pulp Fiction (1994))	0.539344	0.503279	0.377049	0.699088	1.389068
2	(Pulp Fiction (1994))	(Shawshank Redemption, The (1994))	0.503279	0.519672	0.363934	0.723127	1.391506
3	(Shawshank Redemption, The (1994))	(Pulp Fiction (1994))	0.519672	0.503279	0.363934	0.700315	1.391506
4	(Pulp Fiction (1994), Shawshank Redemption, Th...	(Forrest Gump (1994))	0.363934	0.539344	0.293443	0.806306	1.494975
...	...	...	...	...	...	...	...
7453739	(Star Wars: Episode VI - Return of the Jedi (1...	(Jerry Maguire (1996))	0.286885	0.139344	0.095082	0.331429	2.378487
7453740	(Jerry Maguire (1996), Star Wars: Episode IV -...	(Star Wars: Episode VI - Return of the Jedi (1...	0.106557	0.321311	0.095082	0.892308	2.777080
7453741	(Star Wars: Episode VI - Return of the Jedi (1...	(Jerry Maguire (1996), Star Wars: Episode IV -...	0.321311	0.106557	0.095082	0.295918	2.777080
7453742	(Jerry Maguire (1996))	(Star Wars: Episode VI - Return of the Jedi (1...	0.139344	0.286885	0.095082	0.682353	2.378487
7453743	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.416393	0.103279	0.095082	0.228346	2.210974

7453744 rows × 9 columns

- Sort the rules in descending order of lift value.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
7173622	(Star Wars: Episode VI - Return of the Jedi (1...	(Forrest Gump (1994), Matrix, The (1999), Star...	0.108197	0.103279	0.095082	0.878788	8.508899
7173611	(Forrest Gump (1994), Matrix, The (1999), Star...	(Star Wars: Episode VI - Return of the Jedi (1...	0.103279	0.108197	0.095082	0.920635	8.508899
7172272	(Matrix, The (1999), Star Wars: Episode V - Th...	(Star Wars: Episode VI - Return of the Jedi (1...	0.111475	0.104918	0.098361	0.882353	8.409926
7172225	(Star Wars: Episode VI - Return of the Jedi (1...	(Matrix, The (1999), Star Wars: Episode V - Th...	0.104918	0.111475	0.098361	0.937500	8.409926
7173470	(Forrest Gump (1994), Star Wars: Episode V - T...	(Star Wars: Episode VI - Return of the Jedi (1...	0.104918	0.108197	0.095082	0.906250	8.375947
...	...	...	...	...	...	...	...
5068371	(Matrix, The (1999))	(Fugitive, The (1993), True Lies (1994))	0.455738	0.211475	0.096721	0.212230	1.003569
802574	(Fight Club (1999))	(Fugitive, The (1993))	0.357377	0.311475	0.111475	0.311927	1.001449
802575	(Fugitive, The (1993))	(Fight Club (1999))	0.311475	0.357377	0.111475	0.357895	1.001449
2068865	(Matrix, The (1999))	(Pulp Fiction (1994), Dances with Wolves (1990))	0.455738	0.222951	0.101639	0.223022	1.000317
2068860	(Pulp Fiction (1994), Dances with Wolves (1990))	(Matrix, The (1999))	0.222951	0.455738	0.101639	0.455882	1.000317

7453744 rows × 9 columns

- Save rules to file1.csv for future reference.
- Recommending movies based on user inputs

```
Input = 'Dances with Wolves (1990)'
```

```
[76] #film={'Dances with Wolves (1990)', 'Aladdin (1992)', 'Batman (1989)'}
     film={'Dances with Wolves (1990)'}
```

```
[77] get_movie(film)
```

```
['Pulp Fiction (1994)',
 'Lion King, The (1994)',
 'Braveheart (1995)',
 'True Lies (1994)']
```

- Testing the recommendation by using sample test file with some inputs

```
[129] test=pd.read_csv("test.tsv", sep='\t')
      test
```

	movies	recommendation
0	Dances with Wolves (1990)	NaN
1	Fight Club (1999)	NaN
2	Forrest Gump (1994)	NaN

- Save the inferences in output dataframe

```
Output=test.drop("recommendation", axis=1)
Output
```

	movies	Inference
0	Dances with Wolves (1990)	Pulp Fiction (1994)\nLion King, The (1994)\nBr...
1	Fight Club (1999)	Terminator 2: Judgment Day (1991)\nMatrix, The...
2	Forrest Gump (1994)	Ace Ventura: Pet Detective (1994)\nBraveheart ...

- Save the results in output.csv

```
[134] Output.to_csv("Output.csv")
```

Q3) Visualizing maximal frequent pattern set

- First, finding maximal frequent itemsets denoted by fpmax

Using mlxtend we can find the maximal frequent itemsets.
We will use networkx for visualizing the patterns of maximal frequent itemsets.
Take the value of min_support = 0.095

```
[78] import networkx as nx
     from mlxtend.frequent_patterns import fpmax
     fpmax = fpmax(rec_df1, min_support=0.095, use_colnames=True)
```

```
[79] fpmax
```

	support	itemsets
0	0.095082	(Raising Arizona (1987))
1	0.095082	(28 Days Later (2002))
2	0.095082	(Truth About Cats & Dogs, The (1996))
3	0.095082	(Harry Potter and the Order of the Phoenix (20...
4	0.095082	(Naked Gun 33 1/3: The Final Insult (1994))
...	...	...
51651	0.095082	(Shawshank Redemption, The (1994), Jurassic Pa...
51652	0.098361	(Shawshank Redemption, The (1994), Jurassic Pa...
51653	0.095082	(Shawshank Redemption, The (1994), Jurassic Pa...
51654	0.096721	(Jurassic Park (1993), Matrix, The (1999), Sil...
51655	0.095082	(Shawshank Redemption, The (1994), Jurassic Pa...

51656 rows × 2 columns

- Considering an instance of maximal frequent itemsets. Let's consider 7 maximal frequent itemset.

- Find all the subsets of maximal frequent itemsets and store with its corresponding maximal frequent itemset in a dataframe visual

```
visual=pd.DataFrame({'Subset': s1,'Maximal frequent set': d1,})
visual.head()
```

	Subset	Maximal frequent set
0	(Shawshank Redemption, The (1994), Jurassic Pa...	(Jurassic Park (1993), Matrix, The (1999), Sil...
1	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Matrix, The...
2	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...
3	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...
4	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...

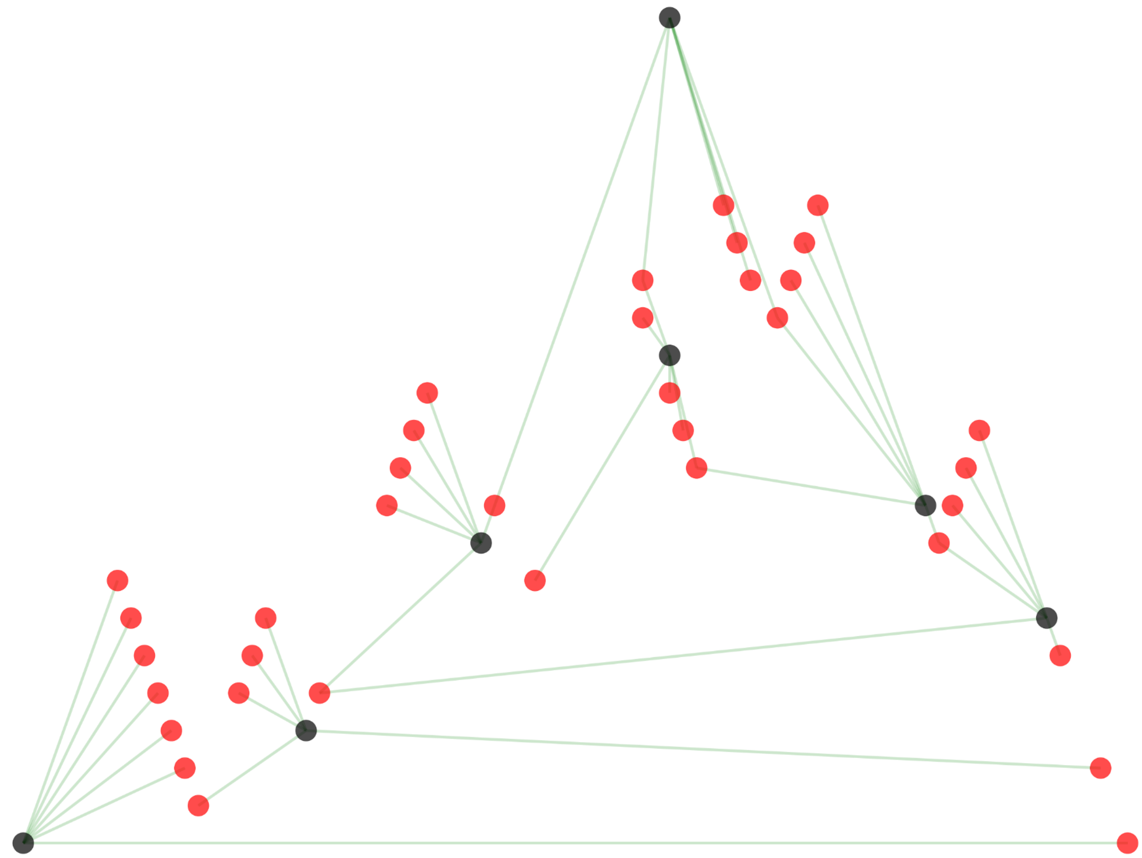
- Convert the subsets and maximal frequent set to frozenset format as for visualizing the maximal frequent itemsets the networkx library uses frozenset.

```
visual['Subset']=visual["Subset"].apply(lambda x: frozenset(x) )
visual['Maximal frequent set']=visual["Maximal frequent set"].apply(lambda x: frozenset(x))
visual.head()
```

	Subset	Maximal frequent set
0	(Shawshank Redemption, The (1994), Jurassic Pa...	(Jurassic Park (1993), Matrix, The (1999), Sil...
1	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Matrix, The...
2	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...
3	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...
4	(Shawshank Redemption, The (1994), Jurassic Pa...	(Shawshank Redemption, The (1994), Jurassic Pa...

- Highlight the maximal frequent itemset node by black color. Remaining subsets are shown by red node color. The edges represent the relation and patterns formed using fp tree. Since, we only took 7 maximal frequent set so the highlighted nodes with black color are only 7 in number. We can further increase it and check the pattern.

```
edges = nx.from_pandas_edgelist(visual,source='Subset',target='Maximal frequent set',edge_attr=None)
l=list(visual["Maximal frequent set"])
color_map = []
for g in edges.nodes():
    if g in l:
        color_map.append('red')
    else:
        color_map.append('black')
plt.subplots(figsize=(40,30))
pos = nx.planar_layout(edges)
nx.draw_networkx_nodes(edges, pos, node_size = 2000,alpha= 0.7,node_color = color_map)
nx.draw_networkx_edges(edges, pos, width = 6, alpha = 0.2, edge_color = 'green')
#nx.draw_networkx_labels(edges, pos, font_size = 25, font_family = 'FreeMono')
plt.grid()
plt.axis('off')
plt.tight_layout()
plt.show()
```



Learnings:

- Firstly we learned how to perform exploratory data analysis for analyzing datasets to summarize their characteristics.
  Using EDA, we got better understanding of data.
  We identified various pattern sets.
  We got a better understanding of datasets given to use.
- We learned that FP growth works faster than Apriori algorithm.
- FP growth works on tree data structure and allows faster scanning.
- We learned how to use different metric rules like support, confidence and lift. We used lift as the score evaluation criteria.
- Using mlxtend, we also generated the maximal frequent itemsets.
- Later, we also visualized the maximal itemsets using the small instance from the itemsets.

References:

- https://towardsdatascience.com/the-fp-growth-algorithm-1ffa20e839b8
- https://towardsdatascience.com/how-to-find-closed-and-maximal-frequent-itemsets-from-fp-growth-861a1ef13e21
- https://www.analyticsvidhya.com/blog/2021/04/mastering-exploratory-data-analysiseda-for-data-science-enthusiasts/