In [1]:

```
# Importing libraries
import pandas as pd
import numpy as np

from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors

import matplotlib.pyplot as plt
import seaborn as sns

from collections import OrderedDict
```

In [2]:

```
from mlxtend.frequent_patterns import apriori, association_rules
```

In [3]:

```
# Reading the dataset
movies = pd.read_csv("movies.csv")
ratings = pd.read_csv("ratings.csv")
```

In [4]:

```
movies.head()
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [5]:

```
ratings.head()
```

Out[5]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [6]:

```
final_dataset = ratings.pivot(index='userId',columns='movieId',values='rating')
```

In [7]:

```
final_dataset
```

Out[7]:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	19
userId														
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
606	2.5	NaN	NaN	NaN	NaN	NaN	2.5	NaN	NaN	NaN	...	NaN	NaN	
607	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
608	2.5	2.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	
609	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	
610	5.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	

610 rows × 9724 columns

In [8]:

```
final_dataset.fillna(0,inplace=True)
```

In [9]:

```
no_user_voted = ratings.groupby('movieId')['rating'].agg('count')
no_movies_voted = ratings.groupby('userId')['rating'].agg('count')
```

In [10]:

```
final_dataset = final_dataset.loc[:, no_user_voted[no_user_voted > 10].index]
```

In [11]:

```
final_dataset = final_dataset.loc[no_movies_voted[no_movies_voted > 50].index,
:]
```

In [12]:

```
final_dataset
```

Out[12]:

movieId	1	2	3	5	6	7	9	10	11	12	...	159093	164179	166528	168250
userId															
1	4.0	0.0	4.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
6	0.0	4.0	5.0	5.0	4.0	4.0	0.0	3.0	4.0	0.0	...	0.0	0.0	0.0	0.0
7	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
605	4.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
606	2.5	0.0	0.0	0.0	0.0	2.5	0.0	0.0	2.5	0.0	...	0.0	0.0	0.0	0.0
607	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	...	0.0	0.0	0.0	0.0
608	2.5	2.0	2.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	...	0.0	0.0	0.0	0.0
610	5.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	...	3.0	5.0	4.0	5.0

378 rows × 2121 columns

In [13]:

```
def hot_encode(x):
    if(x < 3.5):
        return 0
    else:
        return 1
```

In [14]:

```
final_dataset = final_dataset.applymap(hot_encode)
```

In [15]:

```
final_dataset
```

Out[15]:

movieId	1	2	3	5	6	7	9	10	11	12	...	159093	164179	166528	168250	168252
userId																
1	1	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
6	0	1	1	1	1	1	0	0	1	0	...	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
605	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0
606	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
607	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
608	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0
610	1	0	0	0	1	0	0	0	0	0	...	0	1	1	1	1

378 rows × 2121 columns

In [16]:

```
movieIdToName = dict()
for mid in final_dataset.columns:
    movieIdToName[mid] = movies[movies["movieId"] == mid]["title"].values[0]
```

In [17]:

```
cnt = 0
for movieId, movieName in movieIdToName.items():
    print(f"{movieId} -> {movieName}")
    cnt += 1

    if(cnt == 5):
        break
```

```
1 -> Toy Story (1995)
2 -> Jumanji (1995)
3 -> Grumpier Old Men (1995)
5 -> Father of the Bride Part II (1995)
6 -> Heat (1995)
```

In [18]:

```
finalLst = []
for i in final_dataset.index:
    lst = []
    for j in final_dataset.columns:
        if(final_dataset[j][i]):
            lst.append(j)
    finalLst.append(lst)
```

In [19]:

```
print(finalLst[0])
```

```
[1, 3, 6, 47, 50, 101, 110, 151, 157, 163, 216, 231, 235, 260, 333,
349, 356, 362, 367, 441, 457, 480, 527, 543, 552, 553, 590, 592, 59
3, 596, 608, 661, 733, 919, 923, 954, 1023, 1025, 1029, 1031, 1032,
1042, 1049, 1060, 1073, 1080, 1089, 1090, 1092, 1097, 1127, 1136, 11
96, 1197, 1198, 1206, 1208, 1210, 1213, 1214, 1220, 1222, 1224, 124
0, 1256, 1265, 1270, 1275, 1278, 1282, 1291, 1298, 1348, 1500, 1517,
1552, 1573, 1587, 1617, 1620, 1625, 1732, 1777, 1805, 1920, 1954, 19
67, 2000, 2005, 2012, 2018, 2028, 2046, 2054, 2058, 2078, 2090, 209
4, 2096, 2105, 2115, 2116, 2137, 2139, 2141, 2143, 2161, 2174, 2193,
2268, 2273, 2291, 2329, 2353, 2366, 2387, 2395, 2406, 2427, 2450, 24
59, 2470, 2478, 2502, 2529, 2542, 2571, 2580, 2596, 2616, 2628, 264
0, 2641, 2648, 2692, 2700, 2716, 2761, 2797, 2826, 2858, 2872, 2916,
2944, 2947, 2948, 2949, 2959, 2985, 2987, 2991, 2993, 2997, 3033, 30
34, 3052, 3053, 3062, 3147, 3168, 3253, 3273, 3386, 3439, 3440, 344
1, 3448, 3450, 3479, 3489, 3527, 3578, 3617, 3639, 3671, 3702, 3703,
3740, 3744, 3793, 3809, 5060]
```

In [20]:

```
# storing data to file
with open("dataset.txt", "w") as fp:
    for lst in finalLst:
        for x in lst:
            fp.write(str(x))
            fp.write(" ")
        fp.write("\n")
```

In [21]:

```
# encoding the movie id length to fixed size
movieIdSize = 6

# encoding value
encoder = 100000

# Total users
userCnt = 378
```

In [22]:

```
minSupport = 70
```

In [23]:

```
# Too generate new (k+1)-itemsets
def generateKPlus1thSet(itemSet):
    length = len(itemSet)
    candidates = []   # all (k + 1) candidates

    # for each candidate
    for (i, candidate) in enumerate(itemSet):
        # for next all candidates in itemSet
        for j in range(i + 1, length):
            nextCandidate = itemSet[j]
            # matching first (k - 1) elements
            if(candidate[:-movieIdSize] == nextCandidate[:-movieIdSize]):
                newItem = candidate[:-movieIdSize] + candidate[-movieIdSize:] +
nextCandidate[-movieIdSize:]
                candidates.append(newItem)

    return candidates
```

In [24]:

```
# Prune step
def prune(Ck):
    Lk = []

    for item in Ck:
        if(Ck[item] >= minSupport):
            Lk.append(item)

    return Lk
```

In [25]:

```
# calculating support for new itemset
def calculateSupport(candidates):

    Ck = dict()

    for line in finalLst:
        line = list(map(lambda x: str(x + encoder), line))

        for candidate in candidates:

            if(candidate not in Ck):
                Ck[candidate] = 0

            present = True

            for k in range(0, len(candidate), movieIdSize):
                item = candidate[k: k + movieIdSize]

                if(item not in line):
                    present = False
                    break

            if(present):
                Ck[candidate] += 1

    return Ck
```

In [26]:

```
C1 = dict()

for line in finalLst:
    for item in line:
        item = str(item + encoder)
        C1[item] = C1.get(item, 0) + 1

L1 = prune(C1)

print('====================================')
print('     Generating 1 itemset')
print('====================================')

L = generateKPlus1thSet(L1)

k = 2
while(L != []):

    C = calculateSupport(L)

    frequentItemset = prune(C)

    print('     Generating', k, 'itemset')
    print('====================================')

    L = generateKPlus1thSet(frequentItemset)

    k += 1
```

```
====================================
     Generating 1 itemset
====================================
     Generating 2 itemset
====================================
     Generating 3 itemset
====================================
     Generating 4 itemset
====================================
     Generating 5 itemset
====================================
```

In [27]:

```
def decoder(frequentItemset):

    y = [[itemSet[x : x + movieIdSize] for x in range(0, len(itemSet), movieIdSi
ze)] for itemSet in frequentItemset]

    x1 = [list(map(lambda x: str(int(x) - 100000), z)) for z in y]

    movieItemSet = []

    # for each itemset
    for itemSet in x1:
        tempSet = []
        for movieId in itemSet:
            tempSet.append(movieIdToName[int(movieId)])

        movieItemSet.append(tempSet)

    return movieItemSet
```

In [28]:

```
frequentItems = decoder(frequentItemset)

print("Final Frequent ItemSets\n\n")

for itemSet in frequentItems:
    for movie in itemSet:
        print(movie)

    print("\n")
```

```
Final Frequent ItemSets


Star Wars: Episode IV - A New Hope (1977)
Star Wars: Episode V - The Empire Strikes Back (1980)
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost A
rk) (1981)
Star Wars: Episode VI - Return of the Jedi (1983)
Indiana Jones and the Last Crusade (1989)


Matrix, The (1999)
Fight Club (1999)
Lord of the Rings: The Fellowship of the Ring, The (2001)
Lord of the Rings: The Two Towers, The (2002)
Lord of the Rings: The Return of the King, The (2003)
```

In [29]:

```
# Formating frequent itemset to generate association rules
freqItems = []

items = "".join(frequentItemset)

for k in range(0, len(items), movieIdSize):
    item = items[k: k + movieIdSize]
    support = (C1[item] / userCnt)
    movieName = frozenset([movieIdToName[int(item) - encoder]])
    freqItems.append([support, movieName])

freqDf = pd.DataFrame(freqItems, columns=["support", "itemsets"])
print(freqDf)
```

```
    support                                           itemsets
0  0.481481         (Star Wars: Episode IV - A New Hope (1977))
1  0.415344  (Star Wars: Episode V - The Empire Strikes Bac...
2  0.407407  (Raiders of the Lost Ark (Indiana Jones and th...
3  0.380952  (Star Wars: Episode VI - Return of the Jedi (1...
4  0.285714         (Indiana Jones and the Last Crusade (1989))
5  0.507937                                 (Matrix, The (1999))
6  0.431217                                  (Fight Club (1999))
7  0.370370  (Lord of the Rings: The Fellowship of the Ring...
8  0.351852    (Lord of the Rings: The Two Towers, The (2002))
9  0.351852  (Lord of the Rings: The Return of the King, Th...
```

In [30]:

```
rules = association_rules(freqDf, metric ="confidence", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])

rules
```

Out[30]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	co

Generating rules from frequent itemsSet

In [31]:

```
final_dataset.columns = [movieIdToName[mid] for mid in final_dataset.columns]
```

In [32]:

```
final_dataset
```

Out[32]:

userId	Toy Story (1995)	Jumanji (1995)	Grumpier Old Men (1995)	Father of the Bride Part II (1995)	Heat (1995)	Sabrina (1995)	Sudden Death (1995)	GoldenEye (1995)	American President, The (1995)
1	1	0	1	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0
6	0	1	1	1	1	1	0	0	1
7	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
605	1	1	0	0	0	0	0	0	0
606	0	0	0	0	0	0	0	0	0
607	1	0	0	0	0	0	0	0	0
608	0	0	0	0	0	0	0	1	0
610	1	0	0	0	1	0	0	0	0

378 rows × 2121 columns

In [33]:

```
# Building the model
frq_items = apriori(final_dataset, min_support = 0.3, use_colnames = True)
print(frq_items)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
```

	support	itemsets
0	0.357143	(Toy Story (1995))
1	0.301587	(Twelve Monkeys (a.k.a. 12 Monkeys) (1995))
2	0.351852	(Seven (a.k.a. Se7en) (1995))
3	0.375661	(Usual Suspects, The (1995))
4	0.380952	(Braveheart (1995))
5	0.481481	(Star Wars: Episode IV - A New Hope (1977))
6	0.539683	(Pulp Fiction (1994))
7	0.555556	(Shawshank Redemption, The (1994))
8	0.568783	(Forrest Gump (1994))
9	0.317460	(Fugitive, The (1993))
10	0.380952	(Jurassic Park (1993))
11	0.378307	(Schindler's List (1993))
12	0.386243	(Terminator 2: Judgment Day (1991))
13	0.494709	(Silence of the Lambs, The (1991))
14	0.338624	(Fargo (1996))
15	0.378307	(Godfather, The (1972))
16	0.415344	(Star Wars: Episode V - The Empire Strikes Bac...
17	0.407407	(Raiders of the Lost Ark (Indiana Jones and th...
18	0.380952	(Star Wars: Episode VI - Return of the Jedi (1...
19	0.346561	(Back to the Future (1985))
20	0.365079	(Saving Private Ryan (1998))
21	0.507937	(Matrix, The (1999))
22	0.330688	(Sixth Sense, The (1999))
23	0.412698	(American Beauty (1999))
24	0.431217	(Fight Club (1999))
25	0.304233	(Gladiator (2000))
26	0.335979	(Memento (2000))
27	0.325397	(Shrek (2001))
28	0.370370	(Lord of the Rings: The Fellowship of the Ring...
29	0.351852	(Lord of the Rings: The Two Towers, The (2002))
30	0.351852	(Lord of the Rings: The Return of the King, Th...
31	0.309524	(Pulp Fiction (1994), Usual Suspects, The (1995))
32	0.373016	(Star Wars: Episode V - The Empire Strikes Bac...
33	0.314815	(Star Wars: Episode IV - A New Hope (1977), Ra...
34	0.335979	(Star Wars: Episode IV - A New Hope (1977), St...
35	0.333333	(Matrix, The (1999), Star Wars: Episode IV - A...
36	0.386243	(Pulp Fiction (1994), Shawshank Redemption, Th...
37	0.346561	(Pulp Fiction (1994), Forrest Gump (1994))
38	0.359788	(Pulp Fiction (1994), Silence of the Lambs, Th...
39	0.338624	(Pulp Fiction (1994), Matrix, The (1999))
40	0.328042	(Pulp Fiction (1994), Fight Club (1999))
41	0.399471	(Forrest Gump (1994), Shawshank Redemption, Th...
42	0.335979	(Silence of the Lambs, The (1991), Shawshank R...
43	0.317460	(Matrix, The (1999), Shawshank Redemption, The...
44	0.304233	(Fight Club (1999), Shawshank Redemption, The ...
45	0.312169	(Silence of the Lambs, The (1991), Forrest Gum...
46	0.341270	(Matrix, The (1999), Forrest Gump (1994))
47	0.312169	(Star Wars: Episode V - The Empire Strikes Bac...
48	0.309524	(Star Wars: Episode V - The Empire Strikes Bac...
49	0.335979	(Fight Club (1999), Matrix, The (1999))
50	0.320106	(Lord of the Rings: The Fellowship of the Ring...
51	0.314815	(Lord of the Rings: The Fellowship of the Ring...
52	0.314815	(Lord of the Rings: The Return of the King, Th...
53	0.301587	(Lord of the Rings: The Fellowship of the Ring...

In [34]:

```
rules
```

Out[34]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
44	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.314815	0.351852	0.301587	0.957983	2.722689	0.1
46	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Fellowship of the Ring...	0.314815	0.370370	0.301587	0.957983	2.586555	0.1
45	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.320106	0.351852	0.301587	0.942149	2.677686	0.1
39	(Lord of the Rings: The Two Towers, The (2002))	(Lord of the Rings: The Fellowship of the Ring...	0.351852	0.370370	0.320106	0.909774	2.456391	0.1
2	(Star Wars: Episode V - The Empire Strikes Bac...	(Star Wars: Episode IV - A New Hope (1977))	0.415344	0.481481	0.373016	0.898089	1.865262	0.1
42	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.351852	0.351852	0.314815	0.894737	2.542936	0.1
43	(Lord of the Rings: The Two Towers, The (2002))	(Lord of the Rings: The Return of the King, Th...	0.351852	0.351852	0.314815	0.894737	2.542936	0.1
41	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Fellowship of the Ring...	0.351852	0.370370	0.314815	0.894737	2.415789	0.1
7	(Star Wars: Episode VI - Return of the Jedi (1...	(Star Wars: Episode IV - A New Hope (1977))	0.380952	0.481481	0.335979	0.881944	1.831731	0.1
38	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.370370	0.351852	0.320106	0.864286	2.456391	0.1
49	(Lord of the Rings: The Two Towers, The (2002))	(Lord of the Rings: The Fellowship of the Ring...	0.351852	0.314815	0.301587	0.857143	2.722689	0.1
48	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Fellowship of the Ring...	0.351852	0.320106	0.301587	0.857143	2.677686	0.1
40	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.370370	0.351852	0.314815	0.850000	2.415789	0.1
1	(Usual Suspects, The (1995))	(Pulp Fiction (1994))	0.375661	0.539683	0.309524	0.823944	1.526719	0.1

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
33	(Star Wars: Episode VI - Return of the Jedi (1...	(Star Wars: Episode V - The Empire Strikes Bac...	0.380952	0.415344	0.312169	0.819444	1.972930	0.1
47	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.370370	0.314815	0.301587	0.814286	2.586555	0.1
36	(Fight Club (1999))	(Matrix, The (1999))	0.431217	0.507937	0.335979	0.779141	1.533934	0.1
3	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.481481	0.415344	0.373016	0.774725	1.865262	0.1
5	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.407407	0.481481	0.314815	0.772727	1.604895	0.1
19	(Fight Club (1999))	(Pulp Fiction (1994))	0.431217	0.539683	0.328042	0.760736	1.409599	0.0
32	(Star Wars: Episode V - The Empire Strikes Bac...	(Star Wars: Episode VI - Return of the Jedi (1...	0.415344	0.380952	0.312169	0.751592	1.972930	0.1
34	(Star Wars: Episode V - The Empire Strikes Bac...	(Matrix, The (1999))	0.415344	0.507937	0.309524	0.745223	1.467158	0.0
15	(Silence of the Lambs, The (1991))	(Pulp Fiction (1994))	0.494709	0.539683	0.359788	0.727273	1.347594	0.0
21	(Shawshank Redemption, The (1994))	(Forrest Gump (1994))	0.555556	0.568783	0.399471	0.719048	1.264186	0.0
10	(Pulp Fiction (1994))	(Shawshank Redemption, The (1994))	0.539683	0.555556	0.386243	0.715686	1.288235	0.0
26	(Fight Club (1999))	(Shawshank Redemption, The (1994))	0.431217	0.555556	0.304233	0.705521	1.269939	0.0
20	(Forrest Gump (1994))	(Shawshank Redemption, The (1994))	0.568783	0.555556	0.399471	0.702326	1.264186	0.0
6	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.481481	0.380952	0.335979	0.697802	1.831731	0.1
11	(Shawshank Redemption, The (1994))	(Pulp Fiction (1994))	0.555556	0.539683	0.386243	0.695238	1.288235	0.0
9	(Star Wars: Episode IV - A New Hope (1977))	(Matrix, The (1999))	0.481481	0.507937	0.333333	0.692308	1.362981	0.0

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
22	(Silence of the Lambs, The (1991))	(Shawshank Redemption, The (1994))	0.494709	0.555556	0.335979	0.679144	1.222460	0.0
30	(Matrix, The (1999))	(Forrest Gump (1994))	0.507937	0.568783	0.341270	0.671875	1.181250	0.0
14	(Pulp Fiction (1994))	(Silence of the Lambs, The (1991))	0.539683	0.494709	0.359788	0.666667	1.347594	0.0
17	(Matrix, The (1999))	(Pulp Fiction (1994))	0.507937	0.539683	0.338624	0.666667	1.235294	0.0
37	(Matrix, The (1999))	(Fight Club (1999))	0.507937	0.431217	0.335979	0.661458	1.533934	0.1
8	(Matrix, The (1999))	(Star Wars: Episode IV - A New Hope (1977))	0.507937	0.481481	0.333333	0.656250	1.362981	0.0
4	(Star Wars: Episode IV - A New Hope (1977))	(Raiders of the Lost Ark (Indiana Jones and th...	0.481481	0.407407	0.314815	0.653846	1.604895	0.1
12	(Pulp Fiction (1994))	(Forrest Gump (1994))	0.539683	0.568783	0.346561	0.642157	1.129001	0.0
28	(Silence of the Lambs, The (1991))	(Forrest Gump (1994))	0.494709	0.568783	0.312169	0.631016	1.109414	0.0
16	(Pulp Fiction (1994))	(Matrix, The (1999))	0.539683	0.507937	0.338624	0.627451	1.235294	0.0
24	(Matrix, The (1999))	(Shawshank Redemption, The (1994))	0.507937	0.555556	0.317460	0.625000	1.125000	0.0
35	(Matrix, The (1999))	(Star Wars: Episode V - The Empire Strikes Bac...	0.507937	0.415344	0.309524	0.609375	1.467158	0.0
13	(Forrest Gump (1994))	(Pulp Fiction (1994))	0.568783	0.539683	0.346561	0.609302	1.129001	0.0
18	(Pulp Fiction (1994))	(Fight Club (1999))	0.539683	0.431217	0.328042	0.607843	1.409599	0.0
23	(Shawshank Redemption, The (1994))	(Silence of the Lambs, The (1991))	0.555556	0.494709	0.335979	0.604762	1.222460	0.0
31	(Forrest Gump (1994))	(Matrix, The (1999))	0.568783	0.507937	0.341270	0.600000	1.181250	0.0
0	(Pulp Fiction (1994))	(Usual Suspects, The (1995))	0.539683	0.375661	0.309524	0.573529	1.526719	0.1
25	(Shawshank Redemption, The (1994))	(Matrix, The (1999))	0.555556	0.507937	0.317460	0.571429	1.125000	0.0
29	(Forrest Gump (1994))	(Silence of the Lambs, The (1991))	0.568783	0.494709	0.312169	0.548837	1.109414	0.0

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
27	(Shawshank Redemption, The (1994))	(Fight Club (1999))	0.555556	0.431217	0.304233	0.547619	1.269939	0.0

In [35]:

```
# Recommendation for a particular movie
def getRecommendation(movie):
    similarMovies = []
    for movies in frequentItemset:
        if movie in movies:
            similarMovies.extend(movies)
    return similarMovies
```

In [36]:

```
movie = 'Star Wars: Episode IV - A New Hope (1977)'
print("The Recommended Movies are\n")
recommended_movies = getRecommendation(movie)
for movies in recommended_movies:
    if(movies != movie):
        print(movies)
```

```
The Recommended Movies are
```

The End