TABLE OF CONTENTS

Title

Generating movie recommendations through the use of association rule mining.

Introduction

More and more streaming services spring up by the day, and with these services, comes an increased number of movies available to be seen.

As a result of the sheer number of movies available, a movie recommendation system comes in handy for both the viewers and the movie service provider.

For viewers, having to choose from a plethora of movies can usually be a daunting task, and in most instances, there are unknown movies that might be of interest. The movie recommendation system brings these previously unknown (or forgotten) movies to the attention of the viewers This also helps the service providers as the viewers tend to watch more movies of similar interests.

Association rule mining is what drives these recommendation systems. (Park, Kim, Choi, & Kim, 2012)

This report seeks to design a movie recommendation system through the use of Association Rules employing the Apriori Algorithm.

Datasets

This dataset (ml-latest-small) shows ratings (5-Star scale) and tag activity from MovieLens, a movie recommendation site. It is made up of 100836 ratings, and 3683 tag applications done across 9742 movies. These data were created by 610 users between 1996 and 2018. This dataset was generated on September 26, 2018.

The data are contained in the files links.csv, movies.csv, ratings.csv, and tags.csv.

This dataset was gotten from https://grouplens.org/datasets/movielens/ml-latest-small.zip

Explanation and preparation of datasets

The data was downloaded and unzipped, it contained a folder with 5 files inside.

- links.csv
- movies.csv
- ratings.csv
- README.txt
- tags.csv

The movies and ratings files were read into a variable called movies_data and ratings_data respectively. Then a check was done to see if the 'movieId' column had duplicates.

The shapes of the above-stated variables show that movies_data has 9742 columns and 3 rows, while the ratings_data has 100836 rows and 4 columns.

The columns' names for both variables can be seen in the images below:

```
In [5]: movies_data = pd.read_csv('C:\\Users\C\Downloads\movielens_data\movies.csv')
movies_data.head()
```

Out[5]:

	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 [8]: ratings_data = pd.read_csv('ratings.csv')
ratings_data.head()
```

Out[8]:

	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

This report follows the assumption that each move rated was watched by the user that rated it.

To get the users that watched the movies, the ratings_data table needs to be joined with the movies_data table. This was done as shown below:

```
In [10]: #to get the users that watch the movies, the ratings data needs to be joined with the movie data
movies_rate = movies_data.merge(ratings_data, on='movieId')
movies_rate.head()
```

Out[10]:

	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

The algorithm used in this report is the Apriori algorithm, and this works with a “list of lists”, the code below shows the use of list comprehension to create a list by checking through the unique values of the UserID column, and collating a list of the movie-titles that match the UserId. Then the list of lists was saved in a variable called ‘user_movies’.

It becomes clearer in the figure below:

```
In [16]: #Using list comprehension to get the movies each user has watched (rated)
         user_movies = [movies_rate.title[movies_rate.userId==i].tolist() for i in movies_rate.userId.unique()]
         user_movies[:5]
```

```
Out[16]: [['Toy Story (1995)',
           'Grumpier Old Men (1995)',
           'Heat (1995)',
           'Seven (a.k.a. Se7en) (1995)',
           'Usual Suspects, The (1995)',
           'From Dusk Till Dawn (1996)',
           'Bottle Rocket (1996)',
           'Braveheart (1995)',
           'Rob Roy (1995)',
           'Canadian Bacon (1995)',
           'Desperado (1995)',
           'Billy Madison (1995)',
           'Clerks (1994)',
           'Dumb & Dumber (Dumb and Dumber) (1994)',
           'Ed Wood (1994)',
           'Star Wars: Episode IV - A New Hope (1977)',
           'Pulp Fiction (1994)',
           'Stargate (1994)',
           'Tommy Boy (1995)',
```

Implementation in Python

Brief Description

Association rule mining is a data mining process designed to uncover the cross-relationships that exist between products. It helps to identify elements that are associated together, for example, the kinds of products that are usually bought together in a supermarket, or the kinds of movies a viewer might like to watch based on his viewing history.

This is done through calculations involving the following parameters:

Support: Gives a measure of the frequency at which a set of items appear together in all transactions.

Support (X)→(Y) = (Transactions containing both X and Y) / Total number of transactions

Confidence: This gives a measure of the probability that an item will be selected given that other one or other items have been selected.

Confidence (X)→(Y) = (Transactions containing both X and Y) / Transactions containing X

Lift: deviation of the support of the whole rule from the support expected under independence given the supports of both sides of the rule

Lift *(X)→(Y) =[support(X and Y)] / [Support(X) x Support (Y)]* (Garg, 2019)

Application of Data Mining Technique / Procedure Explanation

The TransactionEncoder from the imported ‘mlxtend’ package was initialized and used to encode, fit, and transform the data, yielding the result below:

Implementation ¶

```
In [12]: te = TransactionEncoder() #Initialize the function
         te_1 = te.fit_transform(user_movies) #fits
         df = pd.DataFrame(te_1, columns=te.columns_)
         df.head()
```

Out[12]:

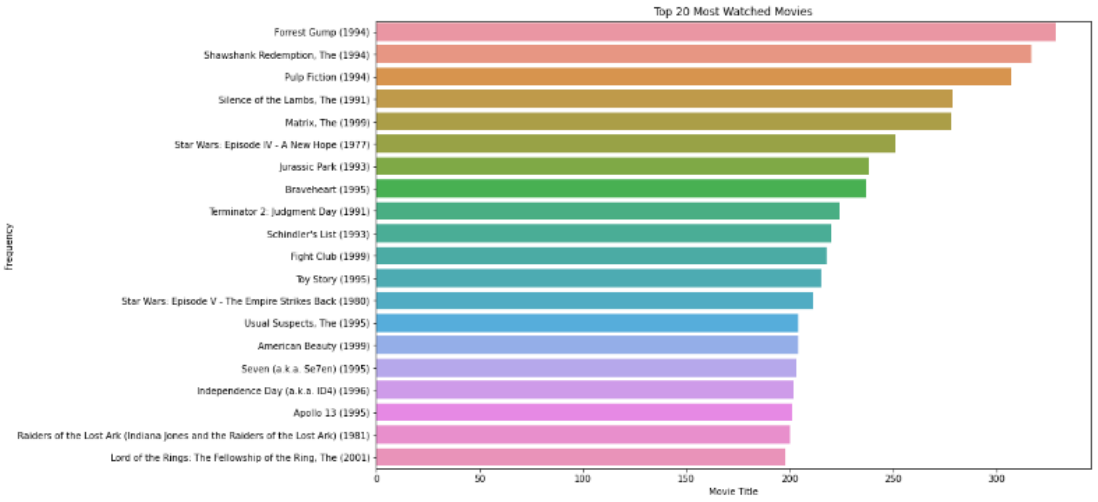
	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 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	True	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

5 rows × 9719 columns

To get a chart of the top 20 most-watched movies. The codes and results are as follows:

```
In [14]: top20 = df.sum().sort_values(ascending=False).iloc[:20] #Top 20 most watched movies
```

```
In [15]: plt.figure(figsize=(14,9))
         sns.barplot(y=top20.index, x=top20.values, orient = 'h')
         plt.title('Top 20 Most Watched Movies')
         plt.ylabel('Frequency')
         plt.xlabel('Movie Title')
         plt.show()
```



Next, the apriori function was used with the following parameters:

- Minimum confidence = 0.6
  With the dataset, this number means that 60% of the time the antecedent was watched, the consequent was also watched. This accounts for a better recommendation model and fewer rules.
- Minimum length = 2
- Minimum support = 0.25 This is a measure of the number of times the antecedent and consequent movies occur together in a 'transaction'.

The number of association rules gotten was 59. See below:

```
In [46]: rules = list(ap(user_movies, min_confidence=0.6, min_length=2, min_support=0.25))
```

```
In [31]: len(rules) #To get the number of rules
Out[31]: 59
```

```
In [34]: rules_no_null = [i for i in rules if len(i[0])>1] #To ensure there are no rules with empty antecendents and consequents
Out[34]: 59
```

Other higher numbers were gotten while tuning the hyperparameters to get a relatively manageable number of rules. As shown below:

```
rules = list(ap(user_movies, min_confidence=0.4, min_length=2, min_support=0.15))
```

```
len(rules) #To get the number of rules
```

```
2857
```

```
rules = list(ap(user_movies, min_confidence=0.5, min_length=2, min_support=0.2))
```

```
len(rules) #To get the number of rules
```

```
332
```

To ensure there were no rules with null antecedents or consequences, the first index of the rules list was looped through to select only the lengths greater than 1. See below:

```
In [34]: rules_no_null = [i for i in rules if len(i[0])>1] #To ensure there are no rules with empty antecendents and consequents
Out[34]: 59
```

A dictionary was made from the 'rules' outcome, to separate it into the various constituents and make it into a 'dataframe'. See the process below:

```
In [47]: rules_dict = [{
             'Antecedent': i[2][0][0],
             'Consequent': i[2][0][1],
             'Support': i[1],
             'Confidence': i[2][0][2],
             'Lift': i[2][0][3]}
             for i in rules_no_null
         ]
```

```
In [49]: ass_rulesDF = pd.DataFrame(rules_dict)
         ass_rulesDF
```

```
In [50]: ass_rulesDF.nlargest(10,'Lift') #Top 10 rules with the highest Lift
```

Out[50]:

	Antecedent	Consequent	Support	Confidence	Lift
55	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.252459	0.777778	2.946860
28	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.263934	0.870270	2.823749
25	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.270492	0.833333	2.747748
26	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.272131	0.838384	2.720288
50	(Star Wars: Episode V - The Empire Strikes Bac...	(Star Wars: Episode VI - Return of the Jedi (1...	0.265574	0.767773	2.389496
58	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.252459	0.613546	2.310265
56	(Star Wars: Episode IV - A New Hope (1977))	(Matrix, The (1999), Star Wars: Episode V - Th...	0.265574	0.645418	2.275752
48	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.311475	0.756972	2.188403
49	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.286885	0.697211	2.169892
42	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.260656	0.795000	1.932072

Afterward, the resulting 'dataframe' was sorted to reveal the 10 rules with the highest lift, support, and confidence respectively. As shown in the images below:

```
ass_rulesDF.nlargest(10,'Lift') #Top 10 rules with the highest Lift
```

	Antecedent	Consequent	Support	Confidence	Lift
55	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.252459	0.777778	2.946860
28	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.263934	0.870270	2.823749
25	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.270492	0.833333	2.747748
26	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.272131	0.838384	2.720288
50	(Star Wars: Episode V - The Empire Strikes Bac...	(Star Wars: Episode VI - Return of the Jedi (1...	0.265574	0.767773	2.389496
58	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.252459	0.613546	2.310265
56	(Star Wars: Episode IV - A New Hope (1977))	(Matrix, The (1999), Star Wars: Episode V - Th...	0.265574	0.645418	2.275752
48	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.311475	0.756972	2.188403
49	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.286885	0.697211	2.169892
42	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.260656	0.795000	1.932072

```
ass_rulesDF.nlargest(10,'Support') #Top 10 rules with the highest Support
```

	Antecedent	Consequent	Support	Confidence	Lift
16	(Forrest Gump (1994))	(Shawshank Redemption, The (1994))	0.378689	0.702128	1.351097
13	(Forrest Gump (1994))	(Pulp Fiction (1994))	0.377049	0.699088	1.389068
37	(Pulp Fiction (1994))	(Shawshank Redemption, The (1994))	0.363934	0.723127	1.391506
38	(Pulp Fiction (1994))	(Silence of the Lambs, The (1991))	0.339344	0.674267	1.474204
17	(Forrest Gump (1994))	(Silence of the Lambs, The (1991))	0.326230	0.604863	1.322461
44	(Shawshank Redemption, The (1994))	(Silence of the Lambs, The (1991))	0.326230	0.627760	1.372522
11	(Forrest Gump (1994))	(Jurassic Park (1993))	0.324590	0.601824	1.542489
12	(Matrix, The (1999))	(Forrest Gump (1994))	0.318033	0.697842	1.293871
48	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.311475	0.756972	2.188403
2	(Braveheart (1995))	(Forrest Gump (1994))	0.300000	0.772152	1.431649

```
ass_rulesDF.nlargest(10,'Confidence') #Top 10 rules with the highest Confidence
```

	Antecedent	Consequent	Support	Confidence	Lift
28	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.263934	0.870270	2.823749
36	(Seven (a.k.a. Se7en) (1995))	(Pulp Fiction (1994))	0.285246	0.857143	1.703118
26	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.272131	0.838384	2.720288
25	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.270492	0.833333	2.747748
31	(Saving Private Ryan (1998))	(Matrix, The (1999))	0.255738	0.829787	1.820756
8	(Fight Club (1999))	(Matrix, The (1999))	0.295082	0.825688	1.811762
1	(Apollo 13 (1995))	(Forrest Gump (1994))	0.270492	0.820896	1.522025
46	(Usual Suspects, The (1995))	(Shawshank Redemption, The (1994))	0.267213	0.799020	1.537546
42	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.260656	0.795000	1.932072
41	(Usual Suspects, The (1995))	(Pulp Fiction (1994))	0.265574	0.794118	1.577888

Visualisation of the results

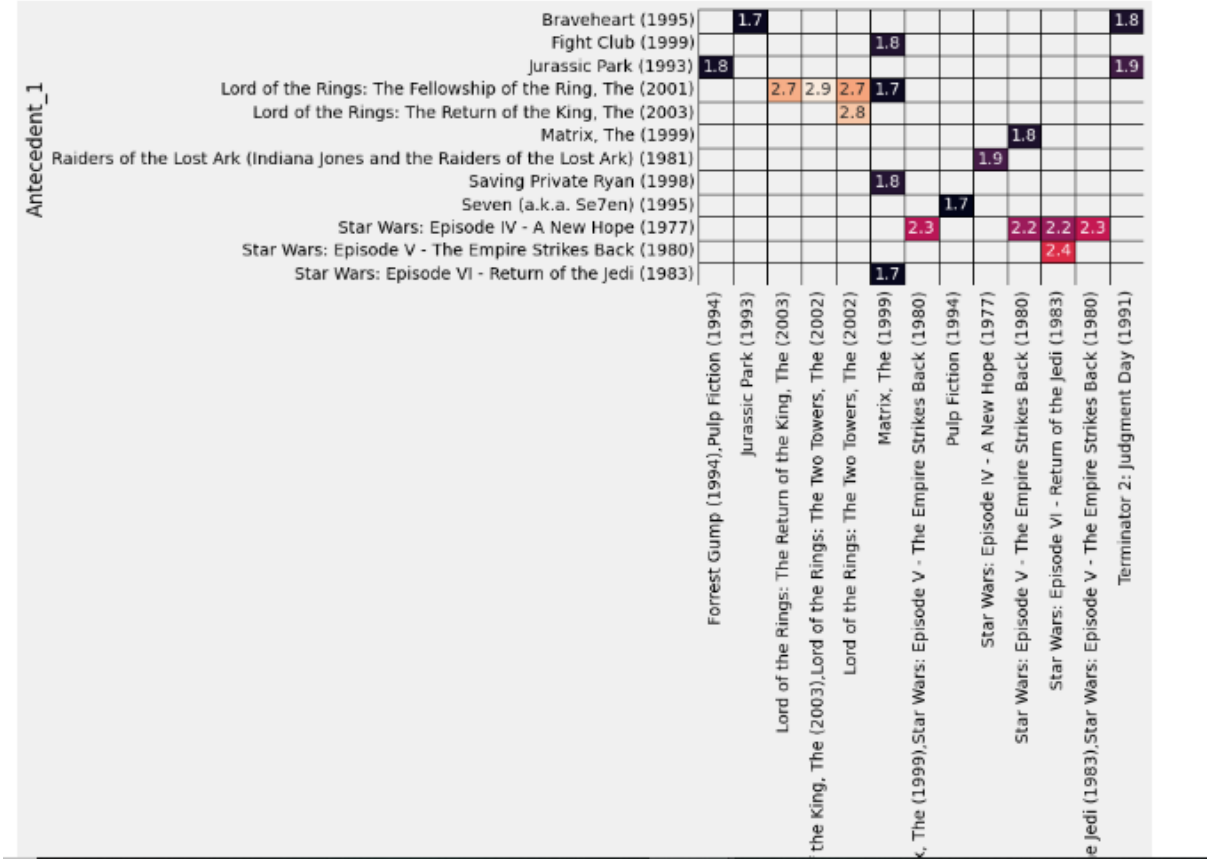
The association rules by lift, 'ass_rulesDF_lift' was sorted in descending order. Hitherto, each element in the antecedents and the consequent columns is a list of movies, but this list is made into a single string with all the elements joined and separated by a comma using a lambda function. The codes used are as shown:

```
In [75]: ass_rulesDF_lift = ass_rulesDF.sort_values('Lift', ascending=False).reset_index(drop=True)
         ass_rulesDF_lift['Antecedent_1']=ass_rulesDF_lift['Antecedent'].map(lambda x: ','.join(list(x)))
         ass_rulesDF_lift['Consequent_1']=ass_rulesDF_lift['Consequent'].map(lambda x: ','.join(list(x)))
```

Next, we get the lift relationship for each antecedent and the consequent(s), then a heatmap showing the relationship graphically was plotted. The codes and heatmap are shown below:

```
list_table = ass_rulesDF_lift.head(20).pivot(index='Antecedent_1', columns='Consequent_1', values='Lift')
```

```
sns.heatmap(list_table, annot=True, cbar=False, linewidths=1, linecolor='black')
plt.show()
```



Relevant Literature

(Lin, 2000) posits that too many rules are not necessary and will likely lead to problems in the model. It was also stated that even though confidence has a big influence on performance, 100% confidence is not ideal, finding the right balance is what leads to better model performance.

In designing this model, the statements above were factual as it was observed that lower confidence leads to a large number of rules (especially in a large dataset), so the confidence was adjusted until a more ideal number of rules was gotten.

Results analysis and discussion

Performance metric for evaluation

The 'lift' was chosen as the ideal metric of evaluation because the lift 'ties' other metrics together. It is responsible for the number of times the consequent occurs while calculating the probability of the consequent occurring, given the antecedent. So when an antecedent leads to a consequent, the lift value is greater than 1.

Presentation of results obtained

Top 10 rules when sorted based on 'Lift':

	Antecedent	Consequent	Support	Confidence	Lift
55	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.252459	0.777778	2.946860
28	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.263934	0.870270	2.823749
25	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.270492	0.833333	2.747748
26	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.272131	0.838384	2.720288
50	(Star Wars: Episode V - The Empire Strikes Bac...	(Star Wars: Episode VI - Return of the Jedi (1...	0.265574	0.767773	2.389496
58	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.252459	0.613546	2.310265
56	(Star Wars: Episode IV - A New Hope (1977))	(Matrix, The (1999), Star Wars: Episode V - Th...	0.265574	0.645418	2.275752
48	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.311475	0.756972	2.188403
49	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode VI - Return of the Jedi (1...	0.286885	0.697211	2.169892
42	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.260656	0.795000	1.932072

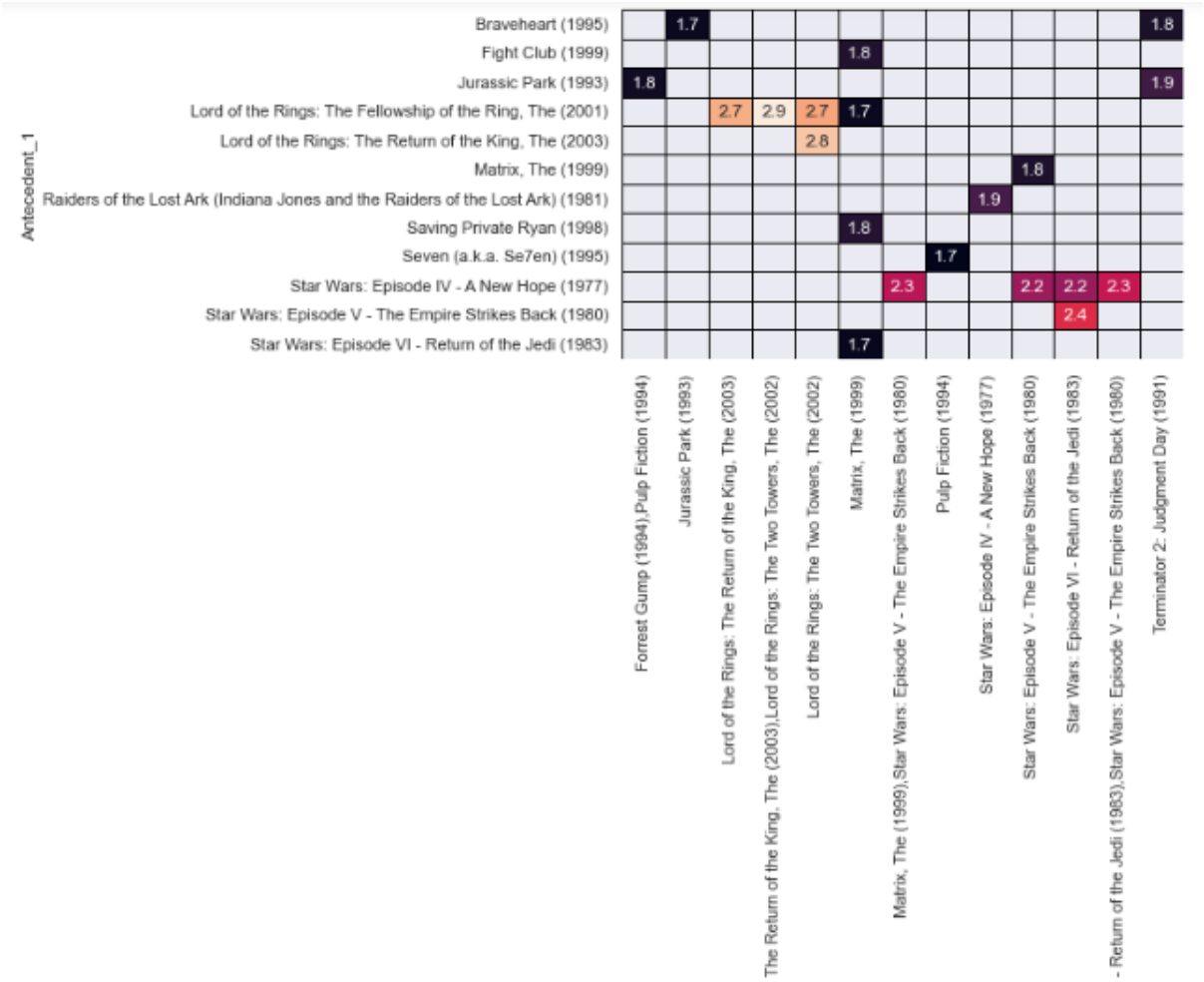
Top 10 rules when sorted based on 'Support:

	Antecedent	Consequent	Support	Confidence	Lift
16	(Forrest Gump (1994))	(Shawshank Redemption, The (1994))	0.378689	0.702128	1.351097
13	(Forrest Gump (1994))	(Pulp Fiction (1994))	0.377049	0.699088	1.389068
37	(Pulp Fiction (1994))	(Shawshank Redemption, The (1994))	0.363934	0.723127	1.391506
38	(Pulp Fiction (1994))	(Silence of the Lambs, The (1991))	0.339344	0.674267	1.474204
17	(Forrest Gump (1994))	(Silence of the Lambs, The (1991))	0.326230	0.604863	1.322461
44	(Shawshank Redemption, The (1994))	(Silence of the Lambs, The (1991))	0.326230	0.627760	1.372522
11	(Forrest Gump (1994))	(Jurassic Park (1993))	0.324590	0.601824	1.542489
12	(Matrix, The (1999))	(Forrest Gump (1994))	0.318033	0.697842	1.293871
48	(Star Wars: Episode IV - A New Hope (1977))	(Star Wars: Episode V - The Empire Strikes Bac...	0.311475	0.756972	2.188403
2	(Braveheart (1995))	(Forrest Gump (1994))	0.300000	0.772152	1.431649

Top 10 rules when sorted based on 'Confidence':

	Antecedent	Consequent	Support	Confidence	Lift
28	(Lord of the Rings: The Return of the King, Th...	(Lord of the Rings: The Two Towers, The (2002))	0.263934	0.870270	2.823749
36	(Seven (a.k.a. Se7en) (1995))	(Pulp Fiction (1994))	0.285246	0.857143	1.703118
26	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Two Towers, The (2002))	0.272131	0.838384	2.720288
25	(Lord of the Rings: The Fellowship of the Ring...	(Lord of the Rings: The Return of the King, Th...	0.270492	0.833333	2.747748
31	(Saving Private Ryan (1998))	(Matrix, The (1999))	0.255738	0.829787	1.820756
8	(Fight Club (1999))	(Matrix, The (1999))	0.295082	0.825688	1.811762
1	(Apollo 13 (1995))	(Forrest Gump (1994))	0.270492	0.820896	1.522025
46	(Usual Suspects, The (1995))	(Shawshank Redemption, The (1994))	0.267213	0.799020	1.537546
42	(Raiders of the Lost Ark (Indiana Jones and th...	(Star Wars: Episode IV - A New Hope (1977))	0.260656	0.795000	1.932072
41	(Usual Suspects, The (1995))	(Pulp Fiction (1994))	0.265574	0.794118	1.577888

Heatmap showing the relationships based on the lift between antecedents and consequents:



Results discussion

From the experiments and the results observed, some points were observed:

1. As the size of the dataset increases, a slight increase in the minimum support leads to a dramatic change in the number of association rules.
2. The lift seems to give the best results because a look at the table showing the top 10 association rules by list shows that the antecedents and the consequents are sequels and prequels. This makes sense as logically, people tend to watch a sequel to a movie after watching that movie.

Ethical, legal, or professional considerations

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

Conclusion

The purpose of the experiment was met, as the recommendation systems successfully showed the movies that people would be interested in after watching a certain movie. Though the system has been shown to work, it is a simplified approach to what is being used by industry giants like Netflix, and Amazon, but the basic concept still follows the same principles..