商务智能第四次作业 关联分析apriori实战

数据集来源：https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv

2108080217 余睿

代码

```
import pandas as pd
import json
import gc
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
pd.options.display.max_columns=100
```

1.读取数据

```
# 读入元数据
movies_metadata = pd.read_csv("../data/movies_metadata.csv")
```

```
d:\OTHER\software\Anaconda3\envs\doog\lib\site-packages\IPython\core\interactiveshell.py:3258: DtypeWarning: Columns (10) have mixed
types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
# 只要 id 标题 题材（原始数据）
movies = movies_metadata[{'id', 'title', 'genres'}]

# 回收metadata
del movies_metadata
gc.collect()

movies
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	genres	id	title
0	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	862	Toy Story
1	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	8844	Jumanji
2	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	15602	Grumpier Old Men
3	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	31357	Waiting to Exhale
4	[{'id': 35, 'name': 'Comedy'}]	11862	Father of the Bride Part II
...	...	...	...
45461	[{'id': 18, 'name': 'Drama'}, {'id': 10751, 'n...	439050	Subdue
45462	[{'id': 18, 'name': 'Drama'}]	111109	Century of Birthing
45463	[{'id': 28, 'name': 'Action'}, {'id': 18, 'nam...	67758	Betrayal
45464	[]	227506	Satan Triumphant
45465	[]	461257	Queerama

45466 rows × 3 columns

制作数据集

```
# gpt-4编写的字符串处理函数
# 转换体裁

def genres2genre(str):
    # Since the input string uses single quotes, we need to replace them with double quotes for valid JSON format
    json_string = str.replace("'", '"')

    # Load the string as a JSON object (list of dictionaries)
    data = json.loads(json_string)

    # Extract the 'name' key from each dictionary and join them with '|'
    result = '|'.join(d['name'] for d in data)
    return result
```

```
# 将genres转换成容易处理的形式

movies['genre'] = movies['genres'].apply(genres2genre)
movies.drop(columns='genres', inplace=True)
movies
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	title	genre
0	862	Toy Story	Animation|Comedy|Family
1	8844	Jumanji	Adventure|Fantasy|Family
2	15602	Grumpier Old Men	Romance|Comedy
3	31357	Waiting to Exhale	Comedy|Drama|Romance
4	11862	Father of the Bride Part II	Comedy
...	...	...	...
45461	439050	Subdue	Drama|Family
45462	111109	Century of Birthing	Drama
45463	67758	Betrayal	Action|Drama|Thriller
45464	227506	Satan Triumphant	
45465	461257	Queerama	

45466 rows × 3 columns

```
# 队电影题材进行ont-hot编码
movies = movies.join(movies.genre.str.get_dummies())
movies.drop(columns='genre', inplace=True)
movies
```

```
C:\Users\64292\AppData\Roaming\Python\Python37\site-packages\pandas\compat\_optional.py:117: DeprecationWarning: distutils Version classes are
deprecated. Use packaging.version instead.
  if distutils.version.LooseVersion(version) < minimum_version:
d:\OTHER\software\Anaconda3\envs\doog\lib\site-packages\setuptools\_distutils\version.py:345: DeprecationWarning: distutils Version classes are
deprecated. Use packaging.version instead.
  other = LooseVersion(other)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	id	title	Action	Adventure	Animation	Aniplex	BROSTA TV	Carousel Productions	Comedy	Crime	Documentary	Drama	Far
0	862	Toy Story	0	0	1	0	0	0	1	0	0	0	1
1	8844	Jumanji	0	1	0	0	0	0	0	0	0	0	1
2	15602	Grumpier Old Men	0	0	0	0	0	0	1	0	0	0	0
3	31357	Waiting to Exhale	0	0	0	0	0	0	1	0	0	1	0
4	11862	Father of the Bride Part II	0	0	0	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
45461	439050	Subdue	0	0	0	0	0	0	0	0	0	1	1
45462	111109	Century of Birthing	0	0	0	0	0	0	0	0	0	1	0
45463	67758	Betrayal	1	0	0	0	0	0	0	0	0	1	0
45464	227506	Satan Triumphant	0	0	0	0	0	0	0	0	0	0	0
45465	461257	Queerama	0	0	0	0	0	0	0	0	0	0	0

45466 rows × 34 columns

关联分析

```
# 获取频繁项集
frequent_itemsets_movies = apriori(movies.drop(columns={'title', 'id'}), use_colnames=True, min_support=0.01)
```

```
d:\OTHER\software\Anaconda3\envs\doog\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:113: DeprecationWarning: DataFrames with non-bool
types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  DeprecationWarning,
```

```
frequent_itemsets_movies
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	support	itemsets
0	0.145075	(Action)
1	0.076893	(Adventure)
2	0.042559	(Animation)
3	0.289931	(Comedy)

	support	itemsets
4	0.094730	(Crime)
...	...	...
70	0.016870	(Crime, Action, Thriller)
71	0.019157	(Drama, Action, Thriller)
72	0.030836	(Comedy, Drama, Romance)
73	0.025821	(Crime, Drama, Thriller)
74	0.015594	(Mystery, Drama, Thriller)

75 rows × 2 columns

```
# 获取规则
rules_movies = association_rules(frequent_itemsets_movies, metric='lift', min_threshold=1.25)
```

```
rules_movies
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(Adventure)	(Action)	0.076893	0.145075	0.038116	0.495709	3.416908	0.026961	1.695301	0.766257
1	(Action)	(Adventure)	0.145075	0.076893	0.038116	0.262735	3.416908	0.026961	1.252070	0.827369
2	(Action)	(Crime)	0.145075	0.094730	0.030088	0.207398	2.189361	0.016345	1.142150	0.635431
3	(Crime)	(Action)	0.094730	0.145075	0.030088	0.317622	2.189361	0.016345	1.252862	0.600093
4	(Fantasy)	(Action)	0.050873	0.145075	0.011019	0.216602	1.493029	0.003639	1.091303	0.347920
...	...	...	...	...	...	...	...	...	...	...
77	(Thriller)	(Drama, Crime)	0.167686	0.055536	0.025821	0.153987	2.772749	0.016509	1.116371	0.768156
78	(Mystery, Drama)	(Thriller)	0.025887	0.167686	0.015594	0.602379	3.592309	0.011253	2.093235	0.740805
79	(Drama, Thriller)	(Mystery)	0.075375	0.054260	0.015594	0.206886	3.812850	0.011504	1.192439	0.797868
80	(Mystery)	(Drama, Thriller)	0.054260	0.075375	0.015594	0.287394	3.812850	0.011504	1.297526	0.780055
81	(Thriller)	(Mystery, Drama)	0.167686	0.025887	0.015594	0.092996	3.592309	0.011253	1.073989	0.867013

82 rows × 10 columns

```
# 选取提升都大于3的电影
rules_movies_lift3 = rules_movies[rules_movies['lift'] > 3].sort_values('lift', ascending=False)
rules_movies_lift3
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
19	(Family)	(Animation)	0.060925	0.042559	0.018849	0.309386	7.269538	0.016256	1.386362	0.918392
18	(Animation)	(Family)	0.042559	0.060925	0.018849	0.442894	7.269538	0.016256	1.685632	0.900776
38	(Fantasy)	(Family)	0.050873	0.060925	0.013483	0.265024	4.350026	0.010383	1.277695	0.811395
39	(Family)	(Fantasy)	0.060925	0.050873	0.013483	0.221300	4.350026	0.010383	1.218860	0.820079
14	(Fantasy)	(Adventure)	0.050873	0.076893	0.015000	0.294855	3.834635	0.011088	1.309103	0.778841
15	(Adventure)	(Fantasy)	0.076893	0.050873	0.015000	0.195080	3.834635	0.011088	1.179157	0.800794
80	(Mystery)	(Drama, Thriller)	0.054260	0.075375	0.015594	0.287394	3.812850	0.011504	1.297526	0.780055
79	(Drama, Thriller)	(Mystery)	0.075375	0.054260	0.015594	0.206886	3.812850	0.011504	1.192439	0.797868
12	(Adventure)	(Family)	0.076893	0.060925	0.017244	0.224256	3.680880	0.012559	1.210548	0.788994
13	(Family)	(Adventure)	0.060925	0.076893	0.017244	0.283032	3.680880	0.012559	1.287516	0.775578
74	(Drama, Thriller)	(Crime)	0.075375	0.094730	0.025821	0.342574	3.616312	0.018681	1.376991	0.782453
75	(Crime)	(Drama, Thriller)	0.094730	0.075375	0.025821	0.272580	3.616312	0.018681	1.271101	0.799182
49	(Thriller)	(Mystery)	0.167686	0.054260	0.032882	0.196091	3.613898	0.023783	1.176427	0.869011
48	(Mystery)	(Thriller)	0.054260	0.167686	0.032882	0.605999	3.613898	0.023783	2.112468	0.764788
78	(Mystery, Drama)	(Thriller)	0.025887	0.167686	0.015594	0.602379	3.592309	0.011253	2.093235	0.740805
81	(Thriller)	(Mystery, Drama)	0.167686	0.025887	0.015594	0.092996	3.592309	0.011253	1.073989	0.867013
52	(Adventure, Drama)	(Action)	0.022940	0.145075	0.011481	0.500479	3.449787	0.008153	1.711490	0.726800
55	(Action)	(Adventure, Drama)	0.145075	0.022940	0.011481	0.079139	3.449787	0.008153	1.061028	0.830631
1	(Action)	(Adventure)	0.145075	0.076893	0.038116	0.262735	3.416908	0.026961	1.252070	0.827369
0	(Adventure)	(Action)	0.076893	0.145075	0.038116	0.495709	3.416908	0.026961	1.695301	0.766257
62	(Action, Thriller)	(Crime)	0.052127	0.094730	0.016870	0.323629	3.416323	0.011932	1.338421	0.746184
63	(Crime)	(Action, Thriller)	0.094730	0.052127	0.016870	0.178082	3.416323	0.011932	1.153246	0.781300
60	(Action, Crime)	(Thriller)	0.030088	0.167686	0.016870	0.560673	3.343591	0.011824	1.894519	0.722664
65	(Thriller)	(Action, Crime)	0.167686	0.030088	0.016870	0.100603	3.343591	0.011824	1.078402	0.842134
41	(Science Fiction)	(Fantasy)	0.067061	0.050873	0.011393	0.169892	3.339515	0.007982	1.143377	0.750912
40	(Fantasy)	(Science Fiction)	0.050873	0.067061	0.011393	0.223952	3.339515	0.007982	1.202166	0.738105
11	(Adventure)	(Animation)	0.076893	0.042559	0.010755	0.139874	3.286572	0.007483	1.113140	0.753684
10	(Animation)	(Adventure)	0.042559	0.076893	0.010755	0.252713	3.286572	0.007483	1.235279	0.726658

```
rules_movies_lift3.shape
```

```
(28, 10)
```

总共得到28条强关联的数据

保存数据

```
frequent_itemsets_movies.to_csv('../data/frequent_itemsets_movies.csv', index=False)
rules_movies_lift3.to_csv('../data/rules_movies_lift3.csv', index=False)
```