



# 关联分析 Apriori

## 如何找到啤酒和尿布的组合

# 关联规则

Association Rules

美国明尼苏达州一家Target被客户投诉，一位中年男子指控Target将婴儿产品优惠券寄给他的女儿（高中生）。但没多久他却来电道歉，因为女儿经他逼问后坦承自己真的怀孕了。

























# 关联规则

Association Rules

关联规则：Association Rules，或者是  
Basket Analysis

解释了：如果一个消费者购买了产品A，那么他有多大几率会购买产品B？

Transaction 1	   
Transaction 2	  
Transaction 3	 
Transaction 4	 
Transaction 5	   
Transaction 6	  
Transaction 7	 
Transaction 8	 

# 关联规则

Association Rules

## 啤酒和尿布：

沃尔玛在分析销售记录时，发现啤酒和尿布经常一起被购买，于是他们调整了货架，把两者放在一起，结果真的提升了啤酒的销量。

原因解释：爸爸在给宝宝买尿布的时候，会顺便给自己买点啤酒？

沃尔玛是最早通过大数据分析而受益的传统零售企业，对消费者购物行为进行跟踪和分析。



# 支持度、置信度和提升度

Association Rules

支持度：是个百分比，指的是某个商品组合出现的次数与总次数之间的比例。支持度越高，代表这个组合出现的频率越大。

“牛奶”的支持度= $4/5=0.8$

“牛奶+面包”的支持度= $3/5=0.6$ 。

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# 支持度、置信度和提升度

Association Rules

置信度：是个条件概念

指的是当你购买了商品A，会有多大的概率购买商品B

置信度（牛奶→啤酒）=2/4=0.5

置信度（啤酒→牛奶）=2/3=0.67

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# 支持度、置信度和提升度

Association Rules

提升度：商品A的出现，对商品B的出现概率提升的程度

如果我们单纯看置信度(可乐→尿布)=1，也就是说可乐出现的时候，用户都会购买尿布，那么当用户购买可乐的时候，就需要推荐尿布么？

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# 支持度、置信度和提升度

Association Rules

提升度：商品A的出现，对商品B的出现概率提升的程度

提升度( $A \rightarrow B$ ) = 置信度( $A \rightarrow B$ ) / 支持度(B)

提升度的三种可能：

- 提升度( $A \rightarrow B$ ) > 1：代表有提升；
- 提升度( $A \rightarrow B$ ) = 1：代表有没有提升，也没有下降；
- 提升度( $A \rightarrow B$ ) < 1：代表有下降。

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# Apriori算法原理

Association Rules

我们把上面案例中的商品用ID来代表，牛奶、面包、尿布、可乐、啤酒、鸡蛋的商品ID分别设置为1-6

Apriori算法就是查找频繁项集(frequent itemset)的过程

频繁项集：支持度大于等于最小支持度(Min Support)阈值的项集。

非频繁项集：支持度小于最小支持度的项集

订单编号	购买的商品
1	1、2、3
2	4、2、3、5
3	1、3、5、6
4	2、1、3、5
5	2、1、3、4

# Apriori算法原理

Association Rules

先计算K=1项的支持度

商品项集	支持度
1	4/5
2	4/5
3	5/5
4	2/5
5	3/5
6	1/5

假设最小支持度=0.5，那么Item4和6不符合最小支持度的，不属于频繁项集

商品项集	支持度
1	4/5
2	4/5
3	5/5
5	3/5

# Apriori算法原理

Association Rules

在这个基础上，我们将商品两两组合，得到k=2项的支持度

商品项集	支持度
1, 2	3/5
1, 3	4/5
1, 5	2/5
2, 3	4/5
2, 5	2/5
3, 5	3/5

筛选掉小于最小值支持度的商品组合

商品项集	支持度
1, 2	3/5
1, 3	4/5
2, 3	4/5
3, 5	3/5

# Apriori算法原理

Association Rules

将商品进行K=3项的商品组合，可以得到：筛选掉小于最小值支持度的商品组合

商品项集	支持度
1, 2, 3	3/5
2, 3, 5	2/5
1, 2, 5	1/5

商品项集	支持度
1, 2, 3	3/5



得到K=3项的频繁项集{1,2,3}，也就是{牛奶、面包、尿布}的组合。

# Apriori算法原理

Association Rules

Apriori算法的流程：

Step1 ,  $K=1$  , 计算 $K$ 项集的支持度；

Step2 , 筛选掉小于最小支持度的项集；

Step3 , 如果项集为空 , 则对应 $K-1$ 项集的结果为最终结果。

否则 $K=K+1$  , 重复1-3步。

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

使用工具包：

```
from efficient_apriori import apriori
```

或者：

```
from mlxtend.frequent_patterns import apriori
```

```
from mlxtend.frequent_patterns import association_rules
```

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

```
from efficient_apriori import apriori

# 设置数据集

transactions = [('牛奶','面包','尿布'),

                ('可乐','面包','尿布','啤酒'),

                ('牛奶','尿布','啤酒','鸡蛋'),

                ('面包','牛奶','尿布','啤酒'),

                ('面包','牛奶','尿布','可乐')]

# 挖掘频繁项集和频繁规则

itemsets, rules = apriori(transactions, min_support=0.5, min_confidence=1)

print("频繁项集 :", itemsets)

print("关联规则 :", rules)
```

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

频繁项集：{1: {'啤酒': 3, ('尿布',): 5, ('牛奶',): 4, ('面包',): 4}, 2: {'啤酒', '尿布': 3, ('尿布', '牛奶'): 4, ('尿布', '面包'): 4, ('牛奶', '面包'): 3}, 3: {'尿布', '牛奶', '面包': 3}}

关联规则：[{啤酒} -> {尿布}, {牛奶} -> {尿布}, {面包} -> {尿布}, {牛奶, 面包} -> {尿布}]

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

# 关联分析的使用场景

Use cases for association analysis

万物皆Transaction：

- 超市购物小票 ( TransactionID => Item )
- 每部电影的分类 ( MovieID => 分类 )
- 每部电影的演员 ( MovieID => 演员 )

# 关联分析的使用场景

Use cases for association analysis

超市购物小票的关联关系

每笔订单的商品 ( TransactionID => Item )

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

BreadBasket数据集（ 21293笔订单 ）：

BreadBasket\_DMS.csv

字段：Date（ 日期 ） ， Time（ 时间 ） ， Transaction（ 交易ID ） Item（ 商品名称 ）

交易ID的范围是[1,9684]，存在交易ID为空的情况，同一笔交易中存在商品重复的情况。以外，有些交易没有购买商品（ 对应的Item为NONE ）

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

# 数据加载

```
data = pd.read_csv('./BreadBasket_DMS.csv')
```

# 统一小写

```
data['Item'] = data['Item'].str.lower()
```

# 去掉none项

```
data = data.drop(data[data.Item == 'none'].index)
```

Date	Time	Transaction	Item
2016/10/30	9:58:11	1	NONE
2016/10/30	10:05:34	2	Scandinavian
2016/10/30	10:05:34	2	Scandinavian
2016/10/30	10:07:57	3	Hot chocolate
2016/10/30	10:07:57	3	Jam
2016/10/30	10:07:57	3	Cookies
2016/10/30	10:08:41	4	Muffin
2016/10/30	10:13:03	5	Coffee
2016/10/30	10:13:03	5	Pastry
2016/10/30	10:13:03	5	Bread
2016/10/30	10:16:55	6	Medialuna
2016/10/30	10:16:55	6	Pastry
2016/10/30	10:16:55	6	Muffin
2016/10/30	10:19:12	7	Medialuna
2016/10/30	10:19:12	7	Pastry
2016/10/30	10:19:12	7	Coffee
2016/10/30	10:19:12	7	Tea
2016/10/30	10:20:51	8	Pastry
2016/10/30	10:20:51	8	Bread
2016/10/30	10:21:59	9	Bread
2016/10/30	10:21:59	9	Muffin

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

```
# 得到一维数组orders_series , 并且将Transaction作为
index, value为Item取值

orders_series = data.set_index('Transaction')['Item']

# 将数据集进行格式转换

transactions = []

temp_index = 0

for i, v in orders_series.items():

    if i != temp_index:

        temp_set = set()

        temp_index = i

        temp_set.add(v)

        transactions.append(temp_set)

    else:

        temp_set.add(v)
```



```
[{'scandinavian'}, {'hot chocolate', 'jam', 'cookies'},
{'muffin'}, {'bread', 'coffee', 'pastry'}, {'medialuna',
'pastry', 'muffin'}, {'tea', 'coffee', 'medialuna', 'pastry'},
{'bread', 'pastry'}, {'bread', 'muffin'}, {'scandinavian',
'medialuna'}].....
```

transaction : 订单数组

每笔订单为一个集合，去掉订单中的重复项

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

```
itemsets, rules = apriori(transactions, min_support=0.02,  
min_confidence=0.5)
```

通过调整min\_support , min\_confidence可以得到不同的频繁项集和关联规则

min\_support=0.02 , min\_confidence=0.5时

一共有33个频繁项集 , 8种关联规则



使用efficient\_apriori工具包

效率较高, 但返回参数较少

频繁项集 : {1: {('alfajores',): 344, ('bread',): 3096, ('brownie',): 379, ('cake',): 983, ('coffee',): 4528, ('cookies',): 515, ('farm house',): 371, ('hot chocolate',): 552, ('juice',): 365, ('medialuna',): 585, ('muffin',): 364, ('pastry',): 815, ('sandwich',): 680, ('scandinavian',): 275, ('scone',): 327, ('soup',): 326, ('tea',): 1350, ('toast',): 318, ('truffles',): 192}, 2: {('bread', 'cake'): 221, ('bread', 'coffee'): 852, ('bread', 'pastry'): 276, ('bread', 'tea'): 266, ('cake', 'coffee'): 518, ('cake', 'tea'): 225, ('coffee', 'cookies'): 267, ('coffee', 'hot chocolate'): 280, ('coffee', 'juice'): 195, ('coffee', 'medialuna'): 333, ('coffee', 'pastry'): 450, ('coffee', 'sandwich'): 362, ('coffee', 'tea'): 472, ('coffee', 'toast'): 224}}

关联规则 : [{cake} -> {coffee}, {cookies} -> {coffee}, {hot chocolate} -> {coffee}, {juice} -> {coffee}, {medialuna} -> {coffee}, {pastry} -> {coffee}, {sandwich} -> {coffee}, {toast} -> {coffee}]

# 如何使用Apriori算法做关联分析

How to use Apriori algorithm for association analysis

```
from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules

hot_encoded_df=data.groupby(['Transaction','Item'])['Item'].count().unstack().r
eset_index().fillna(0).set_index('Transaction')

hot_encoded_df = hot_encoded_df.applymap(encode_units)

frequent_itemsets = apriori(hot_encoded_df, min_support=0.02,
use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=0.5)
```



使用mlxtend.frequent\_patterns工具包

效率较低，但返回参数较多

```
11 0.086116 (pastry)
12 0.071851 (sandwich)
13 0.029057 (scandinavian)
14 0.034552 (scone)
15 0.034446 (soup)
16 0.142646 (tea)
17 0.033601 (toast)
18 0.020287 (truffles)
19 0.023352 (cake, bread)
20 0.090025 (coffee, bread)
21 0.029163 (pastry, bread)
22 0.028107 (tea, bread)
23 0.054734 (cake, coffee)
24 0.023774 (cake, tea)
25 0.028212 (cookies, coffee)
26 0.029586 (hot chocolate, coffee)
27 0.020604 (juice, coffee)
28 0.035186 (medialuna, coffee)
29 0.047549 (pastry, coffee)
30 0.038250 (sandwich, coffee)
31 0.049873 (coffee, tea)
32 0.023669 (toast, coffee)
关联规则: antecedents consequents ... leverage conviction
8 (cake) (coffee) ... 0.005039 1.102557
12 (cookies) (coffee) ... 0.002177 1.083069
14 (hot chocolate) (coffee) ... 0.001680 1.058451
16 (juice) (coffee) ... 0.002152 1.119810
18 (medialuna) (coffee) ... 0.005612 1.210754
20 (pastry) (coffee) ... 0.006347 1.164569
22 (sandwich) (coffee) ... 0.003873 1.115276
26 (toast) (coffee) ... 0.007592 1.764411
```

# 关联分析的使用场景

Use cases for association analysis

## 电影分类中的关联关系

每部电影的分类 ( MovieID => 分类 )

# 案例：电影分类中的关联分析

Case: Association Analysis in Movie Classification

数据集：MovieLens

下载地址：<https://www.kaggle.com/jneupane12/movielens/download>

主要使用的文件：movies.csv

格式：movieId      title      genres

记录了电影ID，标题和分类

我们可以分析下电影分类之间的频繁项集和关联规则

MovieLens 主要使用 Collaborative Filtering 和 Association Rules 相结合的技术，向用户推荐他们感兴趣的电影。

# 案例：电影分类中的关联分析

Case: Association Analysis in Movie Classification

```
# 将genres进行one-hot编码 ( 离散特征有多少取值, 就用多少维来表示这个特征 )

movies_hot_encoded =
movies.drop('genres',1).join(movies.genres.str.get_dummies())

# 将movieId, title设置为index

movies_hot_encoded.set_index(['movieId','title'],inplace=True)

# 挖掘频繁项集, 最小支持度为0.02

itemsets = apriori(movies_hot_encoded,use_colnames=True, min_support=0.02)

# 根据频繁项集计算关联规则, 设置最小提升度为2

rules = association_rules(itemsets, metric='lift', min_threshold=2)
```

```
----- 频繁项集 -----
      support      itemsets
7    0.489185      (Drama)
4    0.306987      (Comedy)
14   0.153164      (Thriller)
12   0.151294      (Romance)
0    0.129042      (Action)
5    0.107743      (Crime)
9    0.095718      (Horror)
31   0.094325      (Drama, Romance)
26   0.093335      (Drama, Comedy)
6    0.090586      (Documentary)
1    0.085380      (Adventure)
27   0.069470      (Comedy, Romance)
32   0.068480      (Drama, Thriller)
13   0.063898      (Sci-Fi)
28   0.062761      (Drama, Crime)
11   0.055503      (Mystery)
8    0.051763      (Fantasy)
29   0.045165      (Thriller, Crime)
20   0.044101      (Drama, Action)
15   0.043772      (War)
3    0.041755      (Children)
22   0.040655      (Thriller, Action)
34   0.039336      (Thriller, Horror)
```

# 案例：电影分类中的关联分析

Case: Association Analysis in Movie Classification

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
9	frozenset({'Mystery'})	frozenset({'Thriller'})	0.055502603	0.153163722	0.029144365	0.525099075	3.428351502	0.02064338	1.783185154
8	frozenset({'Thriller'})	frozenset({'Mystery'})	0.153163722	0.055502603	0.029144365	0.190282432	3.428351502	0.02064338	1.16645289
15	frozenset({'Crime'})	frozenset({'Drama', 'Thriller'})	0.107742503	0.068480094	0.024965173	0.231711466	3.383632432	0.017586957	1.212461029
12	frozenset({'Drama', 'Thriller'})	frozenset({'Crime'})	0.068480094	0.107742503	0.024965173	0.364561028	3.383632432	0.017586957	1.404159228
7	frozenset({'Action'})	frozenset({'Adventure'})	0.129041719	0.08538016	0.035633111	0.276136364	3.234198251	0.024615508	1.263525054
6	frozenset({'Adventure'})	frozenset({'Action'})	0.08538016	0.129041719	0.035633111	0.417346501	3.234198251	0.024615508	1.494813439
16	frozenset({'Action'})	frozenset({'Sci-Fi'})	0.129041719	0.063897646	0.02349879	0.182102273	2.849905792	0.015253328	1.144522502
17	frozenset({'Sci-Fi'})	frozenset({'Action'})	0.063897646	0.129041719	0.02349879	0.367756741	2.849905792	0.015253328	1.377568316
0	frozenset({'Thriller'})	frozenset({'Crime'})	0.153163722	0.107742503	0.045164602	0.294877932	2.736876567	0.028662359	1.265394373
1	frozenset({'Crime'})	frozenset({'Thriller'})	0.107742503	0.153163722	0.045164602	0.419190201	2.736876567	0.028662359	1.458026844
5	frozenset({'Horror'})	frozenset({'Thriller'})	0.095718161	0.153163722	0.039335728	0.410953658	2.683100496	0.024675179	1.437639482
4	frozenset({'Thriller'})	frozenset({'Horror'})	0.153163722	0.095718161	0.039335728	0.256821446	2.683100496	0.024675179	1.216776014

# 关联分析的使用场景

Use cases for association analysis

## 电影演员中的关联关系

每部电影的演员列表 ( MovieID => 演员 )

# 案例：电影演员中的关联分析

Case: Association Analysis among Movie Actors

数据集：MovieActors

来源：movie\_actors.csv

爬虫抓取 movie\_actors\_download.py

格式：title actors

记录了电影标题和演员列表

我们可以分析下电影演员之间的频繁  
项集和关联规则

title	actors
我和我的祖国（2019）	陈凯歌/张一白/管虎/薛晓路/徐峥/宁浩/文牧野/黄渤/张译/吴京/马伊琍/杜江/葛优/刘昊然/陈飞宇
我不是药神（2018）	文牧野/徐峥/王传君/周一围/谭卓/章宇/杨新鸣/王佳佳/王砚辉
疯狂的外星人（2019）	宁浩/黄渤/沈腾/汤姆·派福瑞/马修·莫里森/徐峥/于和伟/雷佳音/刘桦
一出好戏（2018）	黄渤/舒淇/王宝强/张艺兴/于和伟/王迅/李勤勤/李又麟
疯狂的石头（2006）	宁浩/郭涛/刘桦/连晋/黄渤/徐峥/优惠/罗兰/王迅
江湖儿女（2018）	贾樟柯/赵涛/廖凡/徐峥/梁嘉艳/刁亦男/张一白/丁嘉丽/张译
心花路放（2014）	宁浩/黄渤/徐峥/袁泉/周冬雨/陶慧/岳小军/沈腾/张俪
人再囧途之泰囧（2012）	徐峥/王宝强/黄渤/陶虹/谢楠/范冰冰
无人区（2013）	宁浩/徐峥/黄渤/余男/多布杰/王双宝/巴多/杨新鸣/郭虹
港囧（2015）	徐峥/赵薇/包贝尔/杜鹃/葛民辉/李灿森/潘虹/赵有亮
脱口秀大会 第二季（2019）	谭晓虹/于谦/李诞/吴昕/池子/马东/王建国/程璐/庞博
超时空同居（2018）	苏伦/雷佳音/佟丽娅/张衣/于和伟/王正佳/陶虹/李念/李光洁
建党伟业（2011）	韩三平/黄建新/刘烨/冯远征/张嘉译/陈坤/马少骅/李沁/周润发/赵本山
人在囧途（2010）	叶伟民/徐峥/王宝强/李曼/李小璐/左小青/张歆艺/黄小蕾/马健
疯狂的赛车（2009）	宁浩/黄渤/戎祥/九孔/徐峥/王双宝/巴多/董立范/高捷
春娇与志明 春嬌與志明（2017）	彭浩翔/杨千嬅/余文乐/杨冪/徐峥/陈逸宁/林兆霞/谷德昭/詹瑞文
幕后玩家（2018）	任鹏远/徐峥/王丽坤/王砚辉/段博文/任达华/于和伟/朱珠/赵达
催眠大师（2014）	陈正道/徐峥/莫文蔚/胡静/吕中/王耀庆/杨凯迪
囧妈（2020）	徐峥/王祖蓝/彭昱畅/潘虹
爱情呼叫转移（2007）	张建亚/徐峥/刘仪伟/宁静/白冰/伊能静/车永莉/沈星/宋佳
春光灿烂猪八戒（2000）	范小天/叶崇铭/梦继/徐峥/陶虹/孙兴/李立群/陈红/翁虹/屈中恒/钮承泽
夜·店（2009）	杨庆/徐峥/李小璐/乔任梁/杨青/张嘉译/赵英俊
一夜惊喜（2013）	金依萌/范冰冰/李治廷/吴佩慈/蒋劲夫/丹尼尔·海尼/刘维/倪虹洁/黎明
穿越时空的爱恋（2003）	孙太泉/杨军/张庭/徐峥/万弘杰/刘莉莉/孙宝光/史可/李倩/杨俊毅
爱情呼叫转移 II：爱情左右	张建亚/林嘉欣/邓超/黄渤/范伟/古巨基/黄磊/苏有朋/佟大为

# 案例：电影演员中的关联分析

Case: Association Analysis among Movie Actors

```
from selenium import webdriver

# 设置想要下载的导演 数据集

director = u'徐峥'

base_url = 'https://movie.douban.com/subject_search?search_text='+director+'&cat=1002&start='

# 下载指定页面的数据

def download(request_url):

# 将字典类型转化为DataFrame

movie_actors = pd.DataFrame(movie_actors, index=[0])

# DataFrame 行列转换

movie_actors = pd.DataFrame(movie_actors.values.T, index=movie_actors.columns,
columns=movie_actors.index)

movie_actors.index.name = 'title'

movie_actors.set_axis(['actors'], axis='columns', inplace=True)

movie_actors.to_csv('./movie_actors.csv')
```



- 当没有现成的数据源时，可以通过爬虫进行数据抓取，保存在csv文件
- 爬虫抓取属于提升部分，不是本次课重点
- 重点理解：万物皆Transaction
- 掌握方法：挖掘数据集中的频繁项集和关联规则

# 案例：电影演员中的关联分析

Case: Association Analysis among Movie Actors

```
from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules

# 数据加载

movies = pd.read_csv('./movie_actors.csv')

# 将genres进行one-hot编码（离散特征有多少取值，就用多少维来表示这个特征）

movies_hot_encoded = movies.drop('actors',1).join(movies.actors.str.get_dummies('/'))

# 将movieid, title设置为index

movies_hot_encoded.set_index(['title'],inplace=True)

# 挖掘频繁项集，最小支持度为0.05

itemsets = apriori(movies_hot_encoded,use_colnames=True, min_support=0.05)

# 按照支持度从大到小进行时候粗

itemsets = itemsets.sort_values(by="support" , ascending=False)
```

----- 频繁项集 -----

	support	itemsets
2	0.771739	(徐峥)
4	0.141304	(黄渤)
7	0.086957	(黄渤, 徐峥)
0	0.076087	(于和伟)
1	0.076087	(宁浩)
5	0.065217	(宁浩, 徐峥)
6	0.065217	(黄渤, 宁浩)
9	0.065217	(黄渤, 宁浩, 徐峥)
3	0.054348	(王迅)
8	0.054348	(黄渤, 王迅)

# 案例：电影演员中的关联分析

Case: Association Analysis among Movie Actors

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

```
# 根据频繁项集计算关联规则，设置最小提升度为2
```

```
rules = association_rules(itemsets, metric='lift',  
min_threshold=2)
```

```
# 按照提升度从大到小进行排序
```

```
rules = rules.sort_values(by="lift", ascending=False)
```

```
#rules.to_csv('./rules.csv')
```

关联规则						
	antecedents	consequents	antecedent support	consequent support	support	\
5	(宁浩)	(黄渤, 徐峥)	0.076087	0.086957	0.065217	
2	(黄渤, 徐峥)	(宁浩)	0.086957	0.076087	0.065217	
3	(宁浩, 徐峥)	(黄渤)	0.065217	0.141304	0.065217	
4	(黄渤)	(宁浩, 徐峥)	0.141304	0.065217	0.065217	
6	(黄渤)	(王迅)	0.141304	0.054348	0.054348	
7	(王迅)	(黄渤)	0.054348	0.141304	0.054348	
0	(黄渤)	(宁浩)	0.141304	0.076087	0.065217	
1	(宁浩)	(黄渤)	0.076087	0.141304	0.065217	
	confidence	lift	leverage	conviction		
5	0.857143	9.857143	0.058601	6.391304		
2	0.750000	9.857143	0.058601	3.695652		
3	1.000000	7.076923	0.056002	inf		
4	0.461538	7.076923	0.056002	1.736025		
6	0.384615	7.076923	0.046668	1.536685		
7	1.000000	7.076923	0.046668	inf		
0	0.461538	6.065934	0.054466	1.715839		
1	0.857143	6.065934	0.054466	6.010870		

# 关联规则的视角

Association rules perspective

不需要考虑用户一定时期内的偏好，而是基于Transaction

只要能将数据转换成Transaction，就可以做购物篮分析：

Step1、把数据整理成id=>item形式，转换成transaction

Step2、设定关联规则的参数（support、confident）挖掘关联规则

Step3、按某个指标（lift、support等）对以关联规则排序

# 关联规则中的最小支持度、最小置信度该如何确定

How to determine the minimum support and minimum confidence in association rules

- 最小支持度，最小置信度是实验出来的
- 最小支持度：

不同的数据集，最小值支持度差别较大。可能是0.01到0.5之间

可以从高到低输出前20个项集的支持度作为参考

最小置信度：可能是0.5到1之间

提升度：表示使用关联规则可以提升的倍数，是置信度与期望置信度的比值

提升度至少要大于1

# 总结

Summary

- 支持度、置信度、提升度
- 最小支持度，最小置信度是实验出来的
- 基于关联规则的推荐算法：

Apriori算法

FPGrowth算法

PrefixSpan算法

- 万物皆Transaction：

超市购物小票 ( TransactionID => Item )

每部电影的分类 ( MovieID => 分类 )

每部电影的演员 ( MovieID => 演员 )

A blue-tinted image of the Chicago skyline, featuring prominent skyscrapers like the Willis Tower, reflected in the water of Lake Michigan.

# Thank You Using data to solve problems

