智能推荐系统 - 协同过滤算法

14.2 相似度计算三种常见方法

14.2.1 欧式距离

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
df = pd.DataFrame([[5, 1, 5], [4, 2, 2], [4, 2, 1]], columns=['用户1', '用户2', '用户3'], index=['物品A', '物品B', '物品C'])
df
```

Out[1]:

	用户1	用户2	用户3
物品A	5	1	5
物品B	4	2	2
物品C	4	2	1

In [2]:
```
import numpy as np
dist = np.linalg.norm(df.iloc[0] - df.iloc[1])
dist
```

Out[2]: 3.3166247903554

14.2.2 余弦内置函数

In [3]:
```
import pandas as pd
df = pd.DataFrame([[5, 1, 5], [4, 2, 2], [4, 2, 1]], columns=['用户1', '用户2', '用户3'], index=['物品A', '物品B', '物品C'])
df
```

Out[3]:

	用户1	用户2	用户3
物品A	5	1	5
物品B	4	2	2
物品C	4	2	1

In [4]:
```
from sklearn.metrics.pairwise import cosine_similarity
user_similarity = cosine_similarity(df)
pd.DataFrame(user_similarity, columns=['物品A', '物品B', '物品C'], index=['物品A', '物品B', '物品C'])
```

Out[4]:

	物品A	物品B	物品C
物品A	1.000000	0.914659	0.825029
物品B	0.914659	1.000000	0.979958
物品C	0.825029	0.979958	1.000000

14.2.3 皮尔逊相关系数简单版

In [5]:
```
from scipy.stats import pearsonr
X = [1, 3, 5, 7, 9]
Y = [9, 8, 6, 4, 2]
corr = pearsonr(X, Y)
print('相关系数r值为' + str(corr[0]) + '，显著性水平P值为' + str(corr[1])) # P<0.05 再看r
```

相关系数r值为-0.9938837346736191，显著性水平P值为0.0005736731093321747

皮尔逊相关系数小案例

In [6]:
```
import pandas as pd
df = pd.DataFrame([[5, 4, 4], [1, 2, 2], [5, 2, 1]], columns=['物品A', '物品B', '物品C'], index=['用户1', '用户2', '用户3'])
```

In [7]:
```
df
```

Out[7]:

	物品A	物品B	物品C
用户1	5	4	4
用户2	1	2	2
用户3	5	2	1

In [8]:
```
# 物品A与其他物品的皮尔逊相关系数
A = df['物品A']
corr_A = df.corrwith(A)
corr_A
```

Out[8]:
```
物品A    1.000000
物品B    0.500000
物品C    0.188982
dtype: float64
```

In [9]:
```
# 皮尔逊系数表，获取各物品相关性
df.corr()
```

Out[9]:

	物品A	物品B	物品C
物品A	1.000000	0.500000	0.188982
物品B	0.500000	1.000000	0.944911
物品C	0.188982	0.944911	1.000000

14.3 案例实战 - 电影智能推荐系统

1.读取数据

In [10]:
```
import pandas as pd
movies = pd.read_excel('电影.xlsx') #共9712部电影
movies.head()
```

Out[10]:

	电影编号	名称	类别
0	1	玩具总动员（1995）	冒险|动画|儿童|喜剧|幻想
1	2	勇敢者的游戏（1995）	冒险|儿童|幻想
2	3	斗气老顽童2（1995）	喜剧|爱情
3	4	待到梦醒时分（1995）	喜剧|剧情|爱情
4	5	新娘之父2（1995）	喜剧

In [11]:
```
score = pd.read_excel('评分.xlsx')
score.head()
```

Out[11]:

	用户编号	电影编号	评分
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

In [12]:
```
df = pd.merge(movies, score, on='电影编号')
df.head()
```

Out[12]:

	电影编号	名称	类别	用户编号	评分
0	1	玩具总动员 (1995)	冒险|动画|儿童|喜剧|幻想	1	4.0
1	1	玩具总动员 (1995)	冒险|动画|儿童|喜剧|幻想	5	4.0
2	1	玩具总动员 (1995)	冒险|动画|儿童|喜剧|幻想	7	4.5
3	1	玩具总动员 (1995)	冒险|动画|儿童|喜剧|幻想	15	2.5
4	1	玩具总动员 (1995)	冒险|动画|儿童|喜剧|幻想	17	4.5

In [13]:
```
df.to_excel('电影推荐系统.xlsx')
```

In [14]:
```
df['评分'].value_counts()  # 查看各个评分的出现的次数
```

Out[14]:
```
4.0    26794
3.0    20017
5.0    13180
3.5    13129
4.5     8544
2.0     7545
2.5     5544
1.0     2808
1.5     1791
0.5     1369
Name: 评分, dtype: int64
```

2.数据分析

In [15]:
```
ratings = pd.DataFrame(df.groupby('名称')['评分'].mean())
ratings.sort_values('评分', ascending=False).head() #递减排序
```

Out[15]:

名称	评分
假小子 (1997)	5.0
福尔摩斯和华生医生历险记：讹诈之王 (1980)	5.0
机器人 (2016)	5.0
奥斯卡 (1967)	5.0
人类状况III (1961)	5.0

In [16]:
```
ratings['评分次数'] = df.groupby('名称')['评分'].count()
ratings.sort_values('评分次数', ascending=False).head()
```

Out[16]:

名称	评分	评分次数
阿甘正传（1994）	4.164134	329
肖申克的救赎（1994）	4.429022	317
低俗小说（1994）	4.197068	307
沉默的羔羊（1991）	4.161290	279
黑客帝国（1999）	4.192446	278

3.数据处理

In [17]:
```
user_movie = df.pivot_table(index='用户编号', columns='名称', values='评分')
user_movie.tail()
```

Out[17]:

名称	007之黄金眼 (1995)	100个女孩 (2000)	100条街道 (2016)	101忠狗续集:伦敦大冒险 (2003)	101忠狗 (1961)	101雷克雅未克 (2000)	102只斑点狗 (2000)	10件或更少 (2006)	10 (1979)	11:14 (2003)	...	龙珠：神秘冒险 (1988)	龙珠：血红宝石的诅咒 (1986)	龙珠：魔鬼城堡中的睡公主 (1987)	九 (1
用户编号															
606	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
608	4.0	NaN	NaN	NaN	NaN	NaN	NaN	3.5	NaN	NaN	...	NaN	NaN	NaN	
609	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
610	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	

5 rows × 9687 columns

In [18]:
```
user_movie.describe()  # 因为数据量较大，这个耗时可能会有1分钟左右
```

Out[18]:

名称	007之黄金眼（1995）	100个女孩 (2000)	100条街道 (2016)	101忠狗续集:伦敦大冒险 (2003)	101忠狗 (1961)	101雷克雅未克 (2000)	102只斑点狗 (2000)	10件或更少 (2006)	10 (1979)	11:14 (2003)	...	龙珠：神秘冒险 (1988)	龙珠：血红宝石的诅咒 (1986)	龙珠：魔鬼城堡中的睡公主 (1987)
count	132.000000	4.00	1.0	1.0	44.000000	1.0	9.000000	3.000000	4.000000	4.00	...	1.0	1.0	2.000000
mean	3.496212	3.25	2.5	2.5	3.431818	3.5	2.777778	2.666667	3.375000	3.75	...	3.5	3.5	3.250000
std	0.859381	0.50	NaN	NaN	0.751672	NaN	0.833333	1.040833	1.030776	0.50	...	NaN	NaN	0.353553
min	0.500000	2.50	2.5	2.5	1.500000	3.5	2.000000	1.500000	2.000000	3.00	...	3.5	3.5	3.000000
25%	3.000000	3.25	2.5	2.5	3.000000	3.5	2.000000	2.250000	3.125000	3.75	...	3.5	3.5	3.125000
50%	3.500000	3.50	2.5	2.5	3.500000	3.5	2.500000	3.000000	3.500000	4.00	...	3.5	3.5	3.250000
75%	4.000000	3.50	2.5	2.5	4.000000	3.5	3.000000	3.250000	3.750000	4.00	...	3.5	3.5	3.375000
max	5.000000	3.50	2.5	2.5	5.000000	3.5	4.500000	3.500000	4.500000	4.00	...	3.5	3.5	3.500000

8 rows × 9687 columns

4.智能推荐

In [19]:
```
FG = user_movie['阿甘正传（1994）']  # FG是Forrest Gump（），阿甘英文名称的缩写
pd.DataFrame(FG).head()
```

Out[19]:

	阿甘正传（1994）
用户编号	
1	4.0
2	NaN
3	NaN
4	NaN
5	NaN

In [20]:
```
# axis默认为0，计算user_movie各列与FG的相关系数
corr_FG = user_movie.corrwith(FG)
similarity = pd.DataFrame(corr_FG, columns=['相关系数']) # 至少有两位用户都对‘阿甘正传（1994）’和对比的电影评价，否则出现警告
similarity.head()
```

Out[20]:

	相关系数
名称	
007之黄金眼（1995）	0.217441
100个女孩（2000）	NaN
100条街道（2016）	NaN
101忠狗续集:伦敦大冒险（2003）	NaN
101忠狗（1961）	0.141023

In [21]:
```
similarity.dropna(inplace=True)  # 或写成similarity=similarity.dropna(), 剔除NaN
similarity.head()
```

Out[21]:

名称	相关系数
007之黄金眼 (1995)	0.217441
101忠狗 (1961)	0.141023
102只斑点狗 (2000)	-0.857589
10件或更少 (2006)	-1.000000
11:14 (2003)	0.500000

In [22]:
```
similarity_new = pd.merge(similarity, ratings['评分次数'], left_index=True, right_index=True)
similarity_new.head()
```

Out[22]:

名称	相关系数	评分次数
007之黄金眼 (1995)	0.217441	132
101忠狗 (1961)	0.141023	44
102只斑点狗 (2000)	-0.857589	9
10件或更少 (2006)	-1.000000	3
11:14 (2003)	0.500000	4

In [23]:
```
# 第二种合并方式
similarity_new = similarity.join(ratings['评分次数'])
similarity_new.head()
```

Out[23]:

名称	相关系数	评分次数
007之黄金眼（1995）	0.217441	132
101忠狗（1961）	0.141023	44
102只斑点狗（2000）	-0.857589	9
10件或更少（2006）	-1.000000	3
11:14（2003）	0.500000	4

In [24]:
```
similarity_new[similarity_new['评分次数'] > 20].sort_values(by='相关系数', ascending=False).head()  # 选取阈值,评分次数大于20 降
```

Out[24]:

名称	相关系数	评分次数
阿甘正传（1994）	1.000000	329
抓狂双宝（1996）	0.723238	31
雷神：黑暗世界（2013）	0.715809	21
致命吸引力（1987）	0.701856	36
X战警：未来的日子（2014）	0.682284	30

In [ ]:
```

```