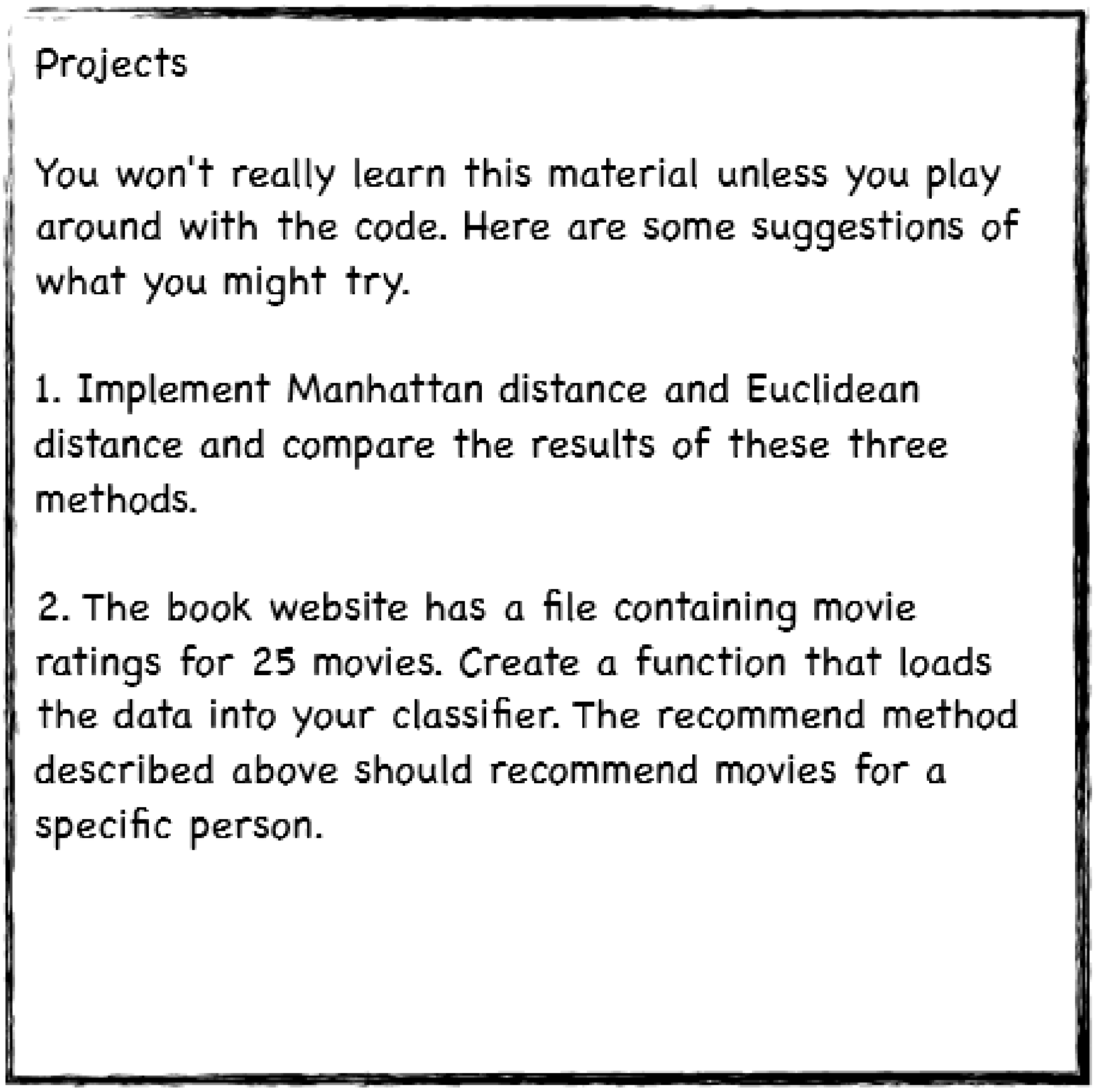
推荐系统作业一



Projects

You won't really learn this material unless you play around with the code. Here are some suggestions of what you might try.

1. Implement Manhattan distance and Euclidean distance and compare the results of these three methods.

2. The book website has a file containing movie ratings for 25 movies. Create a function that loads the data into your classifier. The recommend method described above should recommend movies for a specific person.

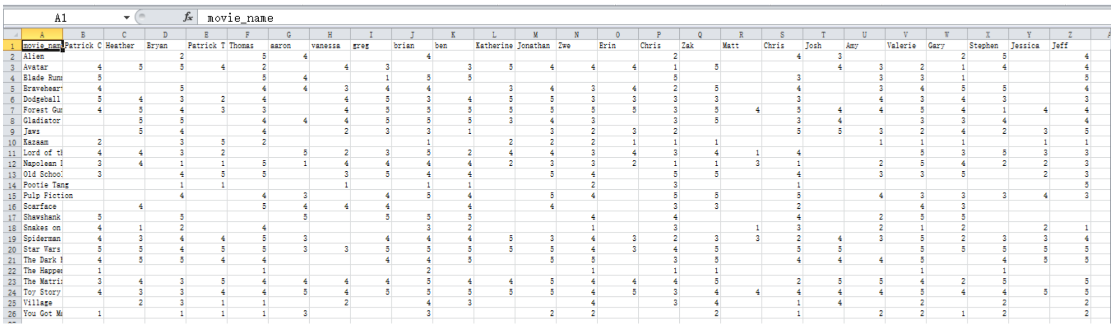
西安邮电大学

电子商务 1402 班

02142053　梁天华

1.数据集（save as Movie_Ratings.csv）

下载地址：http://guidetodatamining.com/assets/data/Movie_Ratings.csv



A1 movie_name

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	movie_nam	Patrick C	Heather	Bryan	Patrick T	Thomas	aaron	vanessa	greg	brian	ben	Katherine	Jonathan	Zwe	Erin	Chris	Zak	Matt	Chris	Josh	Amy	Valerie	Gary	Stephen	Jessica	Jeff
2	Alien			2		5	4			4						2			4	3			2	5		4
3	Avatar	4	5	5	4	2		4	3		3	5	4	4	4	1	5			4	3	2	1	4		4
4	Blade Run	5				5	4		1	5	5					5			3		3	3	1			5
5	Braveheart	4		5		4	4	3	4	4		3	4	3	4	2	5		4		3	4	5	5		4
6	Dodgeball	5	4	3	2	4		4	5	3	4	5	5	3	3	3	3		3		4	3	4	3		3
7	Forest Gu	4	5	4	3	3		4	5	5	5	5	5	5	5	3	5	4	5	4	4	5	4	1	4	4
8	Gladiator		5	5		4	4	4	5	5	5	3	4	3		3	5		3	4		3	3	4		4
9	Jaws		5	4		4		2	3	3	1		3	2	3	2			5	5	3	2	4	2	3	5
10	Kazaam	2		3	5	2				1		2	2	2	1	1	1				1	1	1		1	1
11	Lord of t	4	4	3	2		5	2	3	5	2	4	4	3	4	3	4	1	4			5	3	5	3	3
12	Napolean	3	4	1	1	5	1	4	4	4	4	2	3	3	2	1	1	3	1		2	5	4	2	2	3
13	Old Schoo	3		4	5	5		3	5	4	4		5	4		5	5		4		3	3	5		2	3
14	Pootie Tang			1	1			1		1	1			2		3			1							5
15	Pulp Fiction			4		4	3		4	5	4		5	4		5	5		5		4	3	3	3	4	3
16	Scarface		4			5	4	4	4		4		4			3	3		2			4	3			
17	Shawshank	5		5			5		5	5	5			4		4			4		2	5	5			
18	Snakes on	4	1	2		4				3	2			1		3		1	3		2	1	2		2	1
19	Spiderman	4	3	4	4	5	3		4	4	4	5	3	4	3	2	3	3	2	4	3	5	2	3	3	4
20	Star Wars	5	5	4	5	5	3	3	5	5	5	5	5	4	3	4	5		5	5		5	5	5	5	5
21	The Dark	4	5	5	4	4			4	4	5		5	5		3	5		4	4	4	5		4	5	5
22	The Happe	1				1				2				1		1	1					1		1		
23	The Matri	3	4	3	5	4	4	4	4	5	4	4	5	4	4	4	5		2	5	5	4	2	5		5
24	Toy Story	4	3	3	4	4	5	4	5	5	5	5	5	4	5	3	4	4	4	4	4	5	4	4	5	5
25	Village		2	3	1	1		2		4	3			4		3	4		1	4		2		2		2
26	You Got M	1		1	1	1	3			3			2	2			2		1		2	2	1	2		2

1.余弦相似度算法实现（save as movie_cos.py）————（针对类型：稀疏型数据）

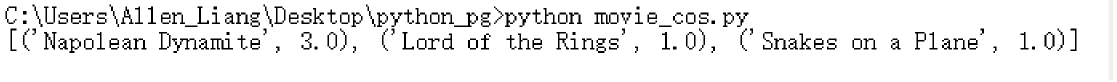
```
import pandas as pd
from math import sqrt
df = pd.read_csv(’Movie_Ratings.csv’)
df = df.fillna(value=0)
#处理缺失值，将缺失值全部设为 0
df.set_index(″movie_name″, drop=True, inplace=True)
users = df.to_dict(orient=″dict″)
#DataFrame 转化为 Dictionarys
#pandas.DataFrame.to_dict()用法 API 参考如下地址
#http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_dict
.html
def cos(rating1, rating2):
    '''计算两个用户之间的余弦相似度'''
    sum_xy = 0
    sum_x2 = 0
    sum_y2 = 0
    for key in rating1:
        if key in rating2 and rating1[key] != 0 and rating2[key] !=0:
        #处理判断为 0 的缺失值
            x = rating1[key]
            y = rating2[key]
            sum_x2 += pow(x, 2)
            sum_y2 += pow(y, 2)
            sum_xy += x * y
    denominator = sqrt(sum_x2) * sqrt(sum_y2)
    if denominator == 0:
        return 0
    else:
        return sum_xy / denominator

def computeNearestNeighbor(username, users):
    '''计算所有用户至 username 用户 A 的余弦相似度，降序排列并返回结果列表'''
    coefficients = []
    for user in users:
        if user != username:
            coefficient = cos(users[user], users[username])
            coefficients.append((coefficient, user))
    # 按余弦相似度降序排列
    coefficients.sort(reverse = True)
    return coefficients

def recommend(username, users):
    #返回余弦相似度最大的用户 B 推荐给用户 A(没评价过的)没看过的电影及评分
    nearest = computeNearestNeighbor(username, users)[0][1]
    recommendations = []
    neighborRatings = users[nearest]
    #最近用户 B 的电影评分列表
    userRatings = users[username]
    #用户 A 的电影评分列表
    for movies in neighborRatings:
        if movies in userRatings and neighborRatings[movies] != 0 and
userRatings[movies] ==0:
            #在电影字典中，找到最近用户 B 看过的（评分的）和用户 A 没看过的（没评
过分的）,推荐给用户 A
            recommendations.append((movies, neighborRatings[movies]))
    #将结果存储到列表中
    return sorted(recommendations, key=lambda artistTuple: artistTuple[1],
reverse = True)

#mytest
#print(cos(users['Bryan'], users['Zwe']))
#print(computeNearestNeighbor("Josh", users))
print( recommend('Josh', users))
```

eg:以 Josh 为对象推荐结果如下



```
C:\Users\Allen_Liang\Desktop\python_pg>python movie_cos.py
[('Napolean Dynamite', 3.0), ('Lord of the Rings', 1.0), ('Snakes on a Plane', 1.0)]
```

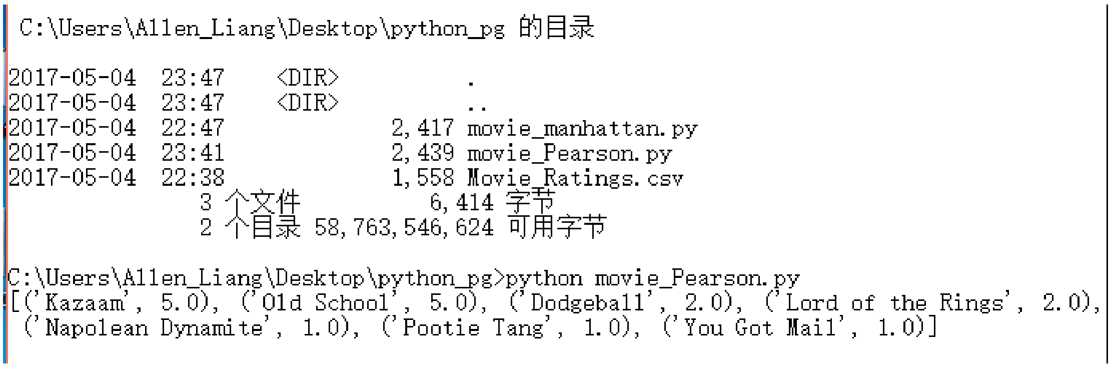
3.威尔逊相关系数算法实现（save as movie_Pearson.py）————（针对类型：分数膨胀）

```
import pandas as pd
from math import sqrt
df = pd.read_csv(’Movie_Ratings.csv’)
df = df.fillna(value=0)
#处理缺失值，将缺失值全部设为0
df.set_index(″movie_name″, drop=True, inplace=True)
users = df.to_dict(orient=″dict″)
#DataFrame 转化为 Dictionarys
#pandas.DataFrame.to_dict()用法 API 参考如下地址
#http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_dict.html
def pearson(rating1, rating2):
    sum_xy = 0
    sum_x = 0
    sum_y = 0
    sum_x2 = 0
    sum_y2 = 0
    n = 0
    for key in rating1:
        if key in rating2 and rating1[key] != 0 and rating2[key] !=0:
            n += 1
            x = rating1[key]
            y = rating2[key]
            sum_xy += x * y
            sum_x += x
            sum_y += y
            sum_x2 += pow(x, 2)
            sum_y2 += pow(y, 2)
    denominator = sqrt(sum_x2 - pow(sum_x, 2) / n) * sqrt(sum_y2 - pow(sum_y, 2) / n)
    if denominator == 0:
        return 0
    else:
        return (sum_xy - (sum_x * sum_y) / n) / denominator
def computeNearestNeighbor(username, users):
    '''计算所有用户至username用户A的pearson相关系数,降序排列并返回结果列表’''
    coefficients = []
    for user in users:
        if user != username:
            coefficient = pearson(users[user], users[username])
            coefficients.append((coefficient, user))
    # 按 pearson 系数降序排列
    coefficients.sort(reverse = True)
    return coefficients
```

```
def recommend(username, users):
    #返回相关系数最大的用户 B 推荐给用户 A(没评价过的)没看过的
    nearest = computeNearestNeighbor(username, users)[0][1]
    recommendations = []
    neighborRatings = users[nearest]
    #最近用户 B 的电影评分列表
    userRatings = users[username]
    #用户 A 的电影评分列表
    for movies in neighborRatings:
        if movies in userRatings and neighborRatings[movies] != 0 and userRatings[movies] ==0:
            #在电影字典中，找到最近用户 B 看过的（评分的）和用户 A 没看过的（没评过分的）,推荐给用户 A
            recommendations.append((movies, neighborRatings[movies]))
    #将结果存储到列表中
    return sorted(recommendations, key=lambda artistTuple: artistTuple[1], reverse = True)

#mytest
#print(pearson(users['Bryan'], users['Zwe']))
#print(computeNearestNeighbor("Josh", users))
print( recommend('Josh', users))
```

eg:以 Josh 为对象推荐结果如下



```
 C:\Users\Allen_Liang\Desktop\python_pg 的目录

2017-05-04  23:47    <DIR>          .
2017-05-04  23:47    <DIR>          ..
2017-05-04  22:47             2,417 movie_manhattan.py
2017-05-04  23:41             2,439 movie_Pearson.py
2017-05-04  22:38             1,558 Movie_Ratings.csv
               3 个文件          6,414 字节
               2 个目录 58,763,546,624 可用字节

C:\Users\Allen_Liang\Desktop\python_pg>python movie_Pearson.py
[('Kazaam', 5.0), ('Old School', 5.0), ('Dodgeball', 2.0), ('Lord of the Rings', 2.0),
 ('Napolean Dynamite', 1.0), ('Pootie Tang', 1.0), ('You Got Mail', 1.0)]
```

4.曼哈顿距离算法实现（save as movie_manhattan.py）————（针对类型：密集型数据）

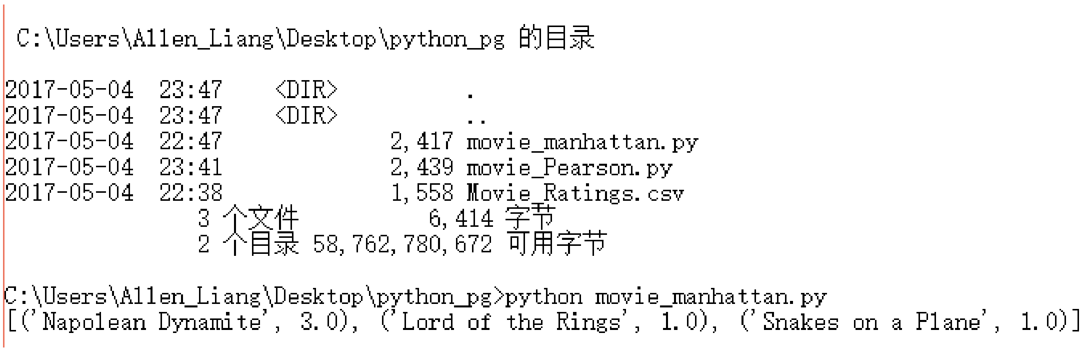
```
import pandas as pd
df = pd.read_csv('Movie_Ratings.csv')
df = df.fillna(value=0)
#处理缺失值，将缺失值全部设为0
df.set_index("movie_name", drop=True, inplace=True)
users = df.to_dict(orient="dict")
#DataFrame 转化为 Dictionarys
#pandas.DataFrame.to_dict()用法 API 参考如下地址
#http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_dict.html
def manhattan(rating1, rating2):
    distance = 0
    commonRatings = False
    for key in rating1:
        if key in rating2 and rating1[key] != 0 and rating2[key] !=0:
            #如果缺失，则不计算两者的曼哈顿距离
            distance += abs(rating1[key] - rating2[key])
            commonRatings = True
    if commonRatings:
        return distance
    else:
        return -1

def computeNearestNeighbor(username, users):
    '''计算所有用户至 username 用户的距离，倒序排列并返回结果列表'''
    distances = []
    for user in users:
        if user != username:
            distance = manhattan(users[user], users[username])
            distances.append((distance, user))
    # 按距离排序——距离近的排在前面
    distances.sort()
    return distances

def recommend(username, users):
    #返回距离最近的用户 B 推荐给用户 A(没评价过的)没看过的
    nearest = computeNearestNeighbor(username, users)[0][1]
    recommendations = []
    neighborRatings = users[nearest]
    #最近用户 B 的电影评分列表
    userRatings = users[username]
    #用户 A 的电影评分列表
    for movies in neighborRatings:
        if  movies in userRatings and neighborRatings[movies] != 0 and userRatings[movies] ==0:
            #在电影字典中，找到最近用户 B 看过的（评分的）和用户 A 没看过的（没评过分的）,推荐给用户 A
            recommendations.append((movies, neighborRatings[movies]))
    #将结果存储到列表中
    return sorted(recommendations, key=lambda artistTuple: artistTuple[1], reverse = True)

#mytest
#print(df.fillna(value=0))
#print(users)
#print(users['Bryan'])
#print(manhattan(users['Josh'], users['Zwe']))
#print(computeNearestNeighbor("Josh", users)))
print( recommend('Josh', users))
```

eg:以 Josh 为对象推荐结果如下



```
 C:\Users\Allen_Liang\Desktop\python_pg 的目录

2017-05-04  23:47    <DIR>          .
2017-05-04  23:47    <DIR>          ..
2017-05-04  22:47             2,417 movie_manhattan.py
2017-05-04  23:41             2,439 movie_Pearson.py
2017-05-04  22:38             1,558 Movie_Ratings.csv
               3 个文件          6,414 字节
               2 个目录 58,762,780,672 可用字节

C:\Users\Allen_Liang\Desktop\python_pg>python movie_manhattan.py
[('Napolean Dynamite', 3.0), ('Lord of the Rings', 1.0), ('Snakes on a Plane', 1.0)]
```