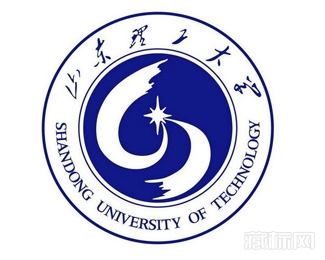
**数据科学导论课程设计报告**



**豆瓣电影推荐与数据可视化分析**

学 院：数学与统计学院

专业班级： 信科1701

姓 名： 李 雪

学 号： 17121202036

指导老师： 何 华

时 间： 2019-6-10

1. **项目简介**

随着社会与科技的发展，个性化推荐系统越来越具有研究价值。互联网把我们带进了信息爆炸的时代，海量信息出现在我们的面前，如何快速的从中找到我们需要的信息已经显得越来越重要，这不仅节约了我们时间，还是我们免去过量信息的困扰。去掉不符合我们兴趣爱好的数据成了当务之急，在这样的条件下，个性化推荐系统应该运而生。个性化推荐算法是推荐系统的基础，推荐系统根据用户的历史信息找到与用户兴趣相似的信息，过滤其它信息，只保留用户可能感兴趣的信息，这在很大程度上削减了信息，受到商家和用户的一直青睐。智能化的服务满足了人们日益增长的娱乐需求，为人们的生活带来便利。

1. **操作要求**
2. 分析项目需要的数据，是否需要分块爬取；
3. 查看网页源码，确定爬取方法；
4. 提取目的网页网址；
5. 解析网页得到网页源码；
6. json拿取电影数据；根据拿到的电影详情页的网址解析详情页，提取具体的电影信息；存入movie\_info数据表；
7. 登录网站获取cookie，
8. 选取部分大V网址；递归爬取用户信息；数据存储people数据表；
9. 清晰数据；
10. 根据给出的用户id计算相似度，找出评分较高的电影，推荐电影；
11. 分析用户兴趣爱好；分析电影信息。
12. **实验流程**
13. **分析所需数据**

爬取电影信息，进行电影数据分析；

爬取用户信息，推荐电影，分析用户爱好兴趣。

1. **确定爬取方式**

豆瓣电影以异步的方式渲染，找真正网址；电影信息可以通过json拿取；用户信息需要构造网址，递归爬取。

1. **电影信息爬取**

def params\_son\_html(son\_html):

print("分析个体电影")

soup = BeautifulSoup(son\_html)

#影视名称

name = soup.find('span',attrs={'property':"v:itemreviewed"}).text

#tag

tags = soup.find\_all('span',attrs={'property':"v:genre"})

data = soup.find('div',attrs={'class':"rating\_wrap clearbox",'rel':"v:rating"})

#得分

grade = data.find('strong',attrs={'class':"ll rating\_num",'property':"v:average"}).text

#评分人数

people = data.find('span',attrs={'property':"v:votes"})

#五星评分

star = data.find\_all('span',attrs={'class':"rating\_per"})

# 写入文件

with open("电影.txt",'a',encoding='utf-8') as f:

f.write(name + ';' + grade + ';' + people + ',')

for num in star:

f.write(num.text+',')

for tag in tags:

f.write(tag+',')

f.write("\n")

f.close()

1. **用户信息爬取**

def movie\_parser(html):

soup = BeautifulSoup(html, "html.parser")

movie\_list = soup.find\_all('a', attrs={'class':"nbg"})

for i in movie\_list:

url = i['href']

#爬取电影名称

son\_html = get\_url(url)

son\_soup = BeautifulSoup(son\_html, "html.parser")

name = son\_soup.find('span',attrs={'property':"v:itemreviewed"}).text

#爬取电影评分

rating = re.find(r'class="rating\d-t"', i)

grade = re.find(r'\d', rating)

print(name + ":" + grade)

save\_movie(name, grade)

# 写入存储

def save\_movie(name, grade):

pass

# with open("people\_movie.txt", 'a') as f:

# f.write(name + "," + grade + ";")

# f.close()

#进入电影主页

def html\_parser(name\_html):

# soup = BeautifulSoup(html,"html.parser")

# src = soup.find\_all('span', attrs={'class':"pl"}) #.find\_all('a',attrs={'target':"\_blank"})

# print(src)

# if src == None:

# return None

# for url in src:

# print(url)

# if "看过" in url.text:

# sum = int(re.compile(r'\d+', url.text))

# global\_url = url['href']

# print(global\_url)

# name\_html = get\_url(global\_url)

name\_soup = BeautifulSoup(name\_html, "html.parser")

name = name\_soup.find\_all('soan', attrs={'class':"verfy-name"}).text

with open("people\_movie.txt", 'a') as f:

f.write(name + ":")

f.close()

for num in range(0, sum+1, 15):

finall\_url = global\_url + "?start=" + str(num)

son\_html = get\_url(finall\_url)

movie\_parser(son\_html)

with open("people\_movie.txt", 'a') as f:

f.write("\n")

f.close()

break

1. **数据库操作**

# 打开数据库连接

con = pymysql.connect("localhost","root","lixue123","shujukexue" )

cur = con.cursor()

#创建数据表

sql = 'create table movie\_info(id int,name varchar(50),score float,five float,four float,three float,two float,one float,tag varchar(20),dirtor varchar(50),actor1 varchar(50),actor2 varchar(50),actor3 varchar(50),url varchar(50))'

cur.execute(sql)

#打开文件

wb = openpyxl.load\_workbook('data\_s\movie.xlsx')

#选择数据表

sheet = wb.get\_sheet\_by\_name('Sheet')

#获取每行的单元值

for row in sheet.rows:

try:

sql = "insert into movie\_info values ('%s', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d')" %\

(row[0].value,row[1].value,row[2].value,row[3].value,row[4].value,row[5].value,row[6].value,row[7].value,row[8].value,row[9].value,row[10].value,row[11].value,row[12].value,row[13].value)

print(sql)

cur.execute(sql)

con.commit()

except:

pass

con.commit()

sql = 'create table people(name varchar(30),movie0 int(7),movie1 int(7),movie2 int(7),movie3 int(7),movie4 int(7),movie5 int(7),movie6 int(7),movie7 int(7),movie8 int(7),movie9 int(7))'

cur.execute(sql)

con.commit()

con.close()

print('数据导入完成')

1. **格式化数据，存入数据库**

def \_\_init\_\_(self):

# 处理空数据

sql = 'select \* from movie\_info order by score desc,five desc,four desc,three desc,two desc,one desc limit 10'

cur.execute(sql)

rows = cur.fetchall()

row = [i for i in rows]

with open('data\_s\people\_movie.txt','r', encoding='UTF-8') as file:

for line in file.readlines():

give\_score = '0123456789'

movie\_score = {}

new = line.split(';')

#未看电影的客户存放在数据库，否接放在列表

if new[3] == '\n':#判断申请账号，但未观看任何一部电影的用户

self.empty\_people[new[1]] = row

else:

for i in new[3].split('，'):

if i == '\n' or i == '':

continue

if not i[:-2] in movie\_score.keys() and i[-1:] in give\_score:

movie\_score[i[0:-2]] = i[-1:]#更新个人观看字典

if not new[1] in self.data.keys():

self.data[new[1]] = movie\_score

file.close()

con.close()

1. **找出相似用户**

#工具函数计算欧氏距离

def Euclidean(self,userID, user\_two):

#取出两位用户评论过的电影和评分

user1\_data = self.data[userID]

user2\_data = self.data[user\_two]

distance = 0.0

#找到两位用户都评论过的电影，并计算欧式距离

for key in user1\_data.keys():

if key in user2\_data.keys():#注意，distance越大表示两者越相似

if user1\_data[key]=='\n' or user2\_data[key]=='\n':

continue

distance += (float(user1\_data[key])-float(user2\_data[key]))\*\*2

return 1/(1+sqrt(distance))#这里返回值越小，相似度越大

#计算某个用户与其他用户的相似度

def top\_simliar(self, userID):

res = []

for userid in self.data.keys():

if not userid == userID: #排除与自己计算相似度

simliar = self.Euclidean(userID,userid)

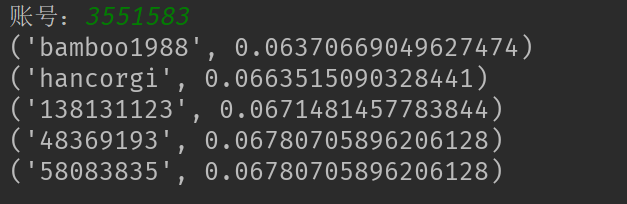
res.append((userid,simliar))

get\_people = sorted(res, key = lambda val:val[1])[:5]

# 返回相似用户的ID以及相似度

return get\_people

例如，得出3551583所有用户中最高相似用户



1. **推荐电影**

#找出推荐电影存入数据库

def get\_movie(self,userID):

# 打开数据库连接

con = pymysql.connect("localhost", "root", "lixue123", "shujukexue")

cur = con.cursor()

simliar\_people = self.top\_simliar(userID)

r\_film = {}

for man in simliar\_people:

#相似用户的电影字典 man = (用户id,相似率)

man\_movie = self.data[man[0]]

for movie in man\_movie.keys():

if movie not in self.data[userID].keys():

sql = 'select \* from movie\_info where name like ' + str('"%' + movie + '%"')

cur.execute(sql)

cols = cur.fetchall()

if len(cols) == 0:

continue

else:

for g\_score in cols:

if len(r\_film) < 10:

r\_film[movie] = (g\_score[2],g\_score)

else:

temp = ['', 10]

for r\_name in r\_film.keys():

#选择高评分电影

if r\_film[r\_name][0] < temp[1]:

temp[0] = r\_name

if g\_score[2] > temp[1]:

r\_film[temp[0]] = (g\_score[2],g\_score)

con.close()

return r\_film

得出推荐电影信息：



1. **分析用户爱好**

#用户分析

def analyze\_cus(self,userID):

# 打开数据库连接

con = pymysql.connect("localhost", "root", "lixue123", "shujukexue")

cur = con.cursor()

#取出该用户所看电影

if userID not in self.data.keys():

return None

saw\_movie = self.data[userID]

sum = 0

save\_score = {'0':0,'1':0,'2':0,'3':0,'4':0,'5':0}

save\_tag = {'其它':0}

for n\_movie in saw\_movie.keys():

sql = 'select tag,score from movie\_info where name like ' + str('"%' + n\_movie + '%"')

cur.execute(sql)

cols = cur.fetchall()

if len(cols) == 0:

sum += 1

save\_tag['其它'] += 1

save\_score[saw\_movie[n\_movie]] += 1

else:

for i in cols:

sum += 1

if i[0] in save\_tag.keys():

save\_tag[i[0]] += 1

else:

save\_tag[i[0]] = 0

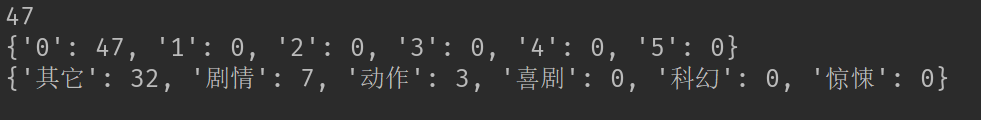
save\_score[saw\_movie[n\_movie]] += 1

con.close()

return sum,save\_score,save\_tag

#找出用户好评情况，所看电影类型趋向

得出用兴趣爱好，所看电影总数，评分趋向，爱看类型：



1. **绘图分析**

#设置字体

mpl.rcParams['font.sans-serif'] = ['SimHei']

movie = pd.DataFrame(pd.read\_excel(r'data\_s/movie.xlsx'))

matplotlib.style.use('ggplot')

#计算平均分

score = movie.groupby(movie['类型']).mean()

#统计

kind\_movie = {}#字典形式

sum = movie.groupby(movie['类型']).size()

for i in sum.index:

kind\_movie[i] = sum[i]

#每种类型电影五星好评相对比

for i in score.keys()[2:]:

pd.DataFrame(score[i],index=kind\_movie.keys()).plot.pie(subplots=True,autopct='%.2f')

plot.show()

#散点图 探索评分与评星比例的关系

for i in score.keys()[2:]:

plot.scatter(score['评分'],score[i])

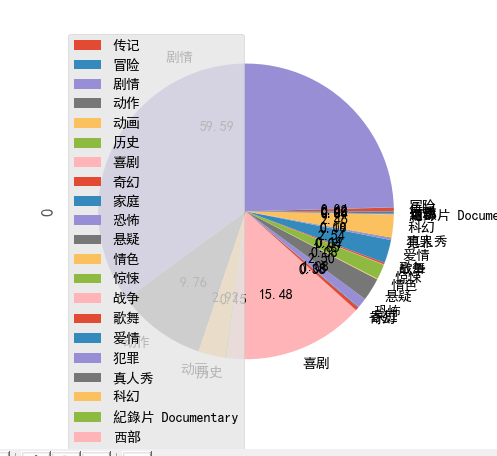
plot.xlim(0, 10)

plot.ylim(0, 1)

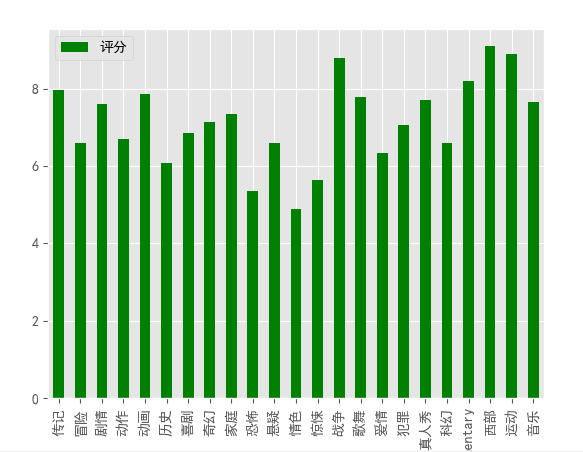
plot.title(i)

plot.show()

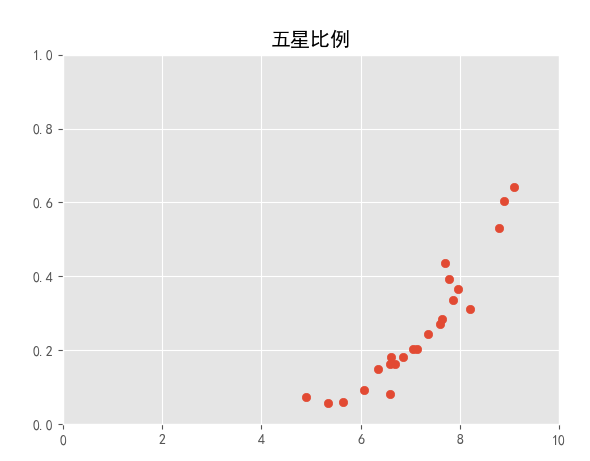
电影类型分布



各类型电影平均得分



电影得分与五星比率



1. **分析单部电影**

sql = 'select name,tag,score,five,four,three,two,one from movie\_info where name like ' + '"%' + name + '%"'

cur.execute(sql)

rows = cur.fetchall()

if len(rows) == 0:

print("查无此电影！")

return

else:

#分析电影相对于与所有电影的平均分

#相对于此标签中所有电影的平均分

for row in rows:

sql = 'select score,tag from movie\_info'

cur.execute(sql)

cols = cur.fetchall()

per\_score = 0.0

tag\_score = 0.0

tag\_num = 0

for i in cols:

per\_score += i[0]

if i[1] == row[1]:

tag\_score += i[0]

tag\_num += 1

print('电影名称：' + str(row[0]))

print('电影评分：' + str(row[2]))

if per\_score/len(cols) > row[2]:

print('低于平均分，电影平均分为' + str(per\_score/len(cols)))

else:

print('高于平均分，电影平均分为' + str(per\_score / len(cols)))

if tag\_score/tag\_num > row[2]:

print('低于同类型电影平均分，同类型电影平均分为' + str(tag\_score / tag\_num))

else:

print('高于同类型电影平均分，同类型电影平均分为' + str(tag\_score / tag\_num))

print('电影类型：' + str(row[1]))

print('电影五星率：'+ str(row[3]) + ',' + str(row[4]) + ',' + str(row[5]) + ',' + str(row[6]) + ',' + str(row[7]))

# 设置字体

mpl.rcParams['font.sans-serif'] = ['SimHei']

movie = pd.DataFrame(pd.read\_excel(r'data\_s\movie.xlsx'))

matplotlib.style.use('ggplot')

# 五星比

index\_name= ['五星','四星','三星','二星','一星']

pd.DataFrame(row[3:8], index=index\_name).plot(color='green', kind='bar')

plot.title('五星评价分布')

plot.show()

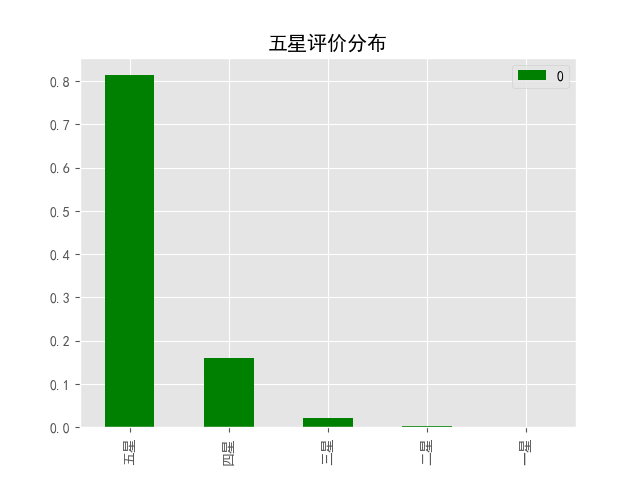
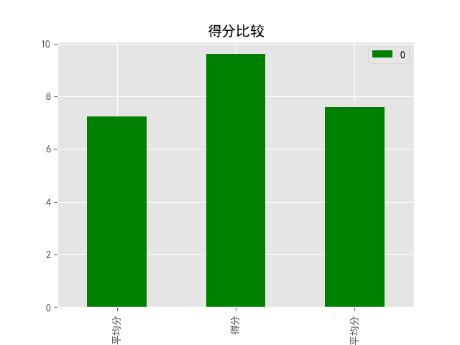
#评分

gride = ['平均分','得分','类型平均分']

pd.DataFrame([per\_score/len(cols), row[2], tag\_score/tag\_num],index=gride).plot(color='green', kind='bar')

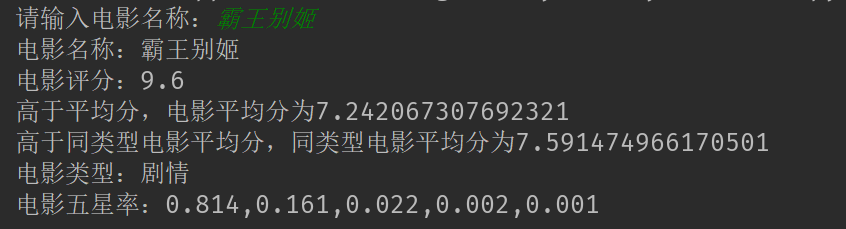
plot.title('得分比较')

plot.show()

****

五星评价分布：根据电影评分，分析五星评价那个区间人数较多，星级越高、电影质量越好，五星占比例越高电影质量越高。

得分比较：电影得分高于一般电影平均分，同类电影平均分，越值得观看。



1. **总结**

根据用户观看过的电影推荐较多同类评分较高的电影，用户的评分不能作为推荐的一项指标，因为很多用户并不对电影打分；单部电影评分高于同类型电影得分，例如喜剧，喜欢看喜剧的人建议观看，一般同类型平均分与所有电影平均分相差无几，五星、四星好评较高者，一般得分较高，建议观看。

1. **学习心得**

现在互联网的快速发展带来了数据快速增加，如何利用海量数据成为了人们关注的问题。此次项目通过分析电影数据，推荐电影信息，对于如何在网页上爬取数据，对数据分析和可视化处理有了更深一步的认识。为以后接触到实际的大数据项目打了一些有效的基础。

1. **附录（整体代码）**

**https://github.com/lixue506/douban-projects**