**爬虫代码**

**zhihu.py文件**

import requests

import re

#模拟浏览器头

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134'}

#登录cookies

cookie = '\_zap=4eb27bd7-a400-4f7b-9d62-475c4872e9ee; d\_c0="AJDl2Y-wNA-PTi7x1hC1kaXbfFVyhJ8zSCE=|1554013983"; \_\_utmv=51854390.100--|2=registration\_date=20190331=1^3=entry\_date=20190331=1; \_\_gads=ID=08a5ae1d214f015d:T=1554014091:S=ALNI\_MZe304ilomnUE5bdyYgE6OTrYyODw; tst=r; \_xsrf=mK7upe5KkLcZ6cjDdlVDS0smadX2lfLQ; q\_c1=1cf6653dc4a345889fc86d2fef6eef0f|1559741884000|1554014047000; l\_cap\_id="MjgxZTIxZjQ1ZmVmNDlhODhmMjgwMjJkOThlZGUzNDg=|1560164722|aeffdb197610b7f86f17721445d41286cde8afb8"; r\_cap\_id="NWUwMWIyMzk0NDU4NDllYjk5YTI0ZDcwZmFhNTQwOWM=|1560164722|95b1f03510dcc75b16c38cd5fe132332b67eb1d5"; cap\_id="Mjc4ZThjYjVjYmZmNDgyZThiMzc0ZThjYTRjMDkzNzY=|1560164722|3df12a8c5b7bad2197694c4cc7c6740bce21c4de"; capsion\_ticket="2|1:0|10:1560165685|14:capsion\_ticket|44:ZmM3ZGVmNjcyNjZkNDI2NWEyMjcwN2UzODhlZDA3YWU=|970db0835d5c8163429ef84d1c7d6843b502a859e8d894f1419a30569e1ff123"; z\_c0="2|1:0|10:1560165691|4:z\_c0|92:Mi4xa2RrRkR3QUFBQUFBa09YWmo3QTBEeVlBQUFCZ0FsVk5PNHZyWFFCVU5mTEUxcXgyOUJqNzVLbHNncGNoeC1QVTdB|51b505972224e345db8521830a8d9f1333eafbf385a2a016f74862884a8eb84d"; \_\_utma=51854390.674597619.1554014082.1560694580.1560725848.38; \_\_utmb=51854390.0.10.1560725848; \_\_utmc=51854390; \_\_utmz=51854390.1560725848.38.19.utmcsr=zhihu.com|utmccn=(referral)|utmcmd=referral|utmcct=/people/madaye/activities; tgw\_l7\_route=73af20938a97f63d9b695ad561c4c10c'

def coo\_regular(cookie):

coo = {}

for k\_v in cookie.split(";"):

k,v = k\_v.split("=",1)

coo[k.strip()] = v.replace('"', '')

return(coo)

cookies = coo\_regular(cookie)

def to\_get\_html(url):

response = requests.get(url,headers = headers, cookies = cookies)

content = response.text.encode('utf-8').decode('utf-8').encode('utf-8').decode('utf-8')

return content

def spiders(url):

page\_content = to\_get\_html(url)

#优秀回答者

if re.findall('"type":"best\_answerer","description":"优秀回答者"',page\_content):

if\_best\_answer = '是'

else:

if\_best\_answer = '否'

#关注者数量

followerCount = re.search(re.compile(r'"followerCount":(\d+)'),page\_content).group(1)

#关注数量

followingCount = re.search(re.compile(r'"followingCount":(\d+)'),page\_content).group(1)

#回答数

answerCount = re.search(re.compile(r'"answerCount":(\d+)'),page\_content).group(1)

#提问数

questionCount = re.search(re.compile(r'"questionCount":(\d+)'),page\_content).group(1)

#文章数

articlesCount = re.search(re.compile(r'"articlesCount":(\d+)'),page\_content).group(1)

#专栏数

columnsCount = re.search(re.compile(r'"columnsCount":(\d+)'),page\_content).group(1)

#认证情况

if re.search(re.compile(r'href="/account/verification/intro'),page\_content):

if re.findall(re.compile(r'"userType":"(\w+)"'),page\_content)[1] == 'people':

verification = '已认证的个人账号'

else:

verification = '已认证的官方账号'

else:

verification = '未认证'

#知乎认证回答数

includedAnswersCount = re.search(re.compile(r'"includedAnswersCount":(\d+)'),page\_content).group(1)

#认证文章数

includedArticlesCount = re.search(re.compile(r'"includedArticlesCount":(\d+)'),page\_content).group(1)

#感谢数

thankedCount = re.search(re.compile(r'"thankedCount":(\d+)'),page\_content).group(1)

#赞数

voteupCount = re.search(re.compile(r'"voteupCount":(\d+)'),page\_content).group(1)

#收藏数

favoritedCount = re.search(re.compile(r'"favoritedCount":(\d+)'),page\_content).group(1)

#专业认证数

recognizedCount = re.search(re.compile(r'"recognizedCount":(\d+)'),page\_content).group(1)

i\_list = [followerCount,followingCount,answerCount,questionCount,articlesCount,columnsCount,verification,if\_best\_answer,includedAnswersCount,includedArticlesCount,voteupCount,thankedCount,favoritedCount,recognizedCount]

return i\_list

**follow.py文件**

#encoding utf-8

import requests

import re

import os

import csv

import time

import random

import threading

import copy

from ip import ip

from update\_ip import update\_ip

from zhihu import spiders

all = []

'''

all这个数组是计划者存放着一个用户所有的url\_list、name\_list和url\_token\_list

形式如下：

all.append((name\_list[i],url\_list[i],url\_token\_list[i]))

用户名|个人主页|唯一ID|

一下是all的一个例子

all = [[(下厨房,https://www.zhihu.com/org/xia-chu-fang/activities,xia-chu-fang),(何明科,https://www.zhihu.com/org/he-ming-ke/activities,he-ming-ke)……],[下厨房,https://www.zhihu.com/org/xia-chu-fang/activities,xia-chu-fang),(何明科,https://www.zhihu.com/org/he-ming-ke/activities,he-ming-ke)……]","……]

'''

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36','cookies':'\_zap=7c7470ca-1e7c-440a-9796-eecf4d38b516; \_\_DAYU\_PP=BiiaBj7Zen2UNfmRzbyq20d231854a42; d\_c0="AJAgybI8iA2PTiF1-pqTqkmiu70lSkjAvFM=|1525261016"; \_xsrf=z5eshtUP7uwxOUs2ygmAN3eU5BwFqN95; q\_c1=7928b2b4c1c04b6a89d80f727fc50b9b|1555937282000|1512121450000; r\_cap\_id="MDQyNGNmNWIxNmMxNDliY2E2NGM5ODE3NjA2NTNkMWI=|1555937282|965bb1a73087c52ebf3256a999cbdb8b14240fb9"; cap\_id="NDc5MWRiZjYwMmM2NGE0OTkzMDk0NjgxYTk1NmUxZDI=|1555937282|2eb41b60c5d5d65fbcf020be83c953d47212c328"; l\_cap\_id="YTNkMGE0NmM0YWUxNDlmNjg1ZDZjNWIxM2ZhNGQ5YjQ=|1555937282|19d65843614ed8c3265a4d1932f1c2c683f9c871"; \_\_utma=51854390.670307953.1555937283.1555937283.1555937283.1; \_\_utmz=51854390.1555937283.1.1.utmcsr=zhihu.com|utmccn=(referral)|utmcmd=referral|utmcct=/question/34554321; \_\_utmv=51854390.000--|3=entry\_date=20171201=1; capsion\_ticket="2|1:0|10:1556807997|14:capsion\_ticket|44:NTdkMjlhODlkNzA0NDBmYzlhMWZjMTNiMjRkZDQ1NmY=|ddf4fe297c4d0bceed3a73853d02cace23b5956b5aa9fb21acfa07d1372d38be"; z\_c0="2|1:0|10:1556807999|4:z\_c0|92:Mi4xUno2ckJRQUFBQUFBa0NESnNqeUlEU1lBQUFCZ0FsVk5QMC00WFFBMjFlTTFzYXpBTHFsVDNnU2FBWnBqY2l5Y1NR|2b6fe5d64e7841185a993f0860cafbe43539201588f33e36a7b29136cb9d8e06"; tst=r; tgw\_l7\_route=7bacb9af7224ed68945ce419f4dea76d'}

proxies = {'http':ip[random.randint(0,len(ip) - 1 )]}

cookie = '\_zap=7c7470ca-1e7c-440a-9796-eecf4d38b516; \_\_DAYU\_PP=BiiaBj7Zen2UNfmRzbyq20d231854a42; d\_c0="AJAgybI8iA2PTiF1-pqTqkmiu70lSkjAvFM=|1525261016"; \_xsrf=z5eshtUP7uwxOUs2ygmAN3eU5BwFqN95; z\_c0="2|1:0|10:1556807999|4:z\_c0|92:Mi4xUno2ckJRQUFBQUFBa0NESnNqeUlEU1lBQUFCZ0FsVk5QMC00WFFBMjFlTTFzYXpBTHFsVDNnU2FBWnBqY2l5Y1NR|2b6fe5d64e7841185a993f0860cafbe43539201588f33e36a7b29136cb9d8e06"; tst=r; q\_c1=7928b2b4c1c04b6a89d80f727fc50b9b|1558661520000|1512121450000; \_\_utmv=51854390.100-1|2=registration\_date=20170811=1^3=entry\_date=20170811=1; \_\_utmc=51854390; \_\_utma=51854390.1240898352.1558936844.1559727712.1559733446.6; \_\_utmz=51854390.1559733446.6.4.utmcsr=zhihu.com|utmccn=(referral)|utmcmd=referral|utmcct=/topic/19776749/organize/entire; tgw\_l7\_route=060f637cd101836814f6c53316f73463'

def coo\_regular(cookie):

coo = {}

for k\_v in cookie.split(";"):

k,v = k\_v.split("=",1)

coo[k.strip()] = v.replace('"', '')

return(coo)

cookies = coo\_regular(cookie)

def to\_get\_html(url):

response = requests.get(url,headers = headers, proxies=proxies, cookies = cookies)

content = str(response.content).encode("utf-8").decode("unicode\_escape").encode("utf-8").decode("unicode\_escape")

return str(content)

def get\_all(content):

'''

得到需要的东西

url\_list是抓的那个用户关注的人的主页链接的集合（因为一页有二十人）

name\_list关注的人的用户名的列表

url\_token\_list关注的人的唯一ID

followr\_count\_list关注它的人有多少（这个没有用了","不用管它）

'''

# with open("1.txt","w") as f:

# f.write(str(content))

url\_list = re.findall(re.compile('"url": "(https://www.zhihu.com/people[\w\W]+?)"'),content)

# name\_list = re.findall(re.compile('"articles\_count": \d+?, "type": "[\w\W]+?", "name": "([\w\W]+?)", "url": "https://www.zhihu.com/people'),content)

name\_list = re.findall(re.compile('"articles\_count": \d+?, "name": "([\w\W]+?)", "url": "https://www.zhihu.com/people'),content)

url\_token\_list = re.findall(re.compile('"url\_token": "([\w\W]+?)"'),content)

follower\_count\_list = re.findall(re.compile('"follower\_count": ([\w\W]+?),'),content)

return url\_list,name\_list,url\_token\_list,follower\_count\_list

#废？

def init\_file(length,name):

# if not os.path.isfile('start\_from\_' + name + '.csv'):

# with open('start\_from\_' + name + '.csv','w',newline='') as csvfile:

# spamwriter = csv.writer(csvfile,dialect='excel')

# spamwriter.writerow(["name","url","url\_token","follower\_count"])

if os.path.isfile('process\_follow\_'+name):

with open('process\_follow\_'+name,"r") as f:

data = f.readlines()

L = data[0].strip().split(" ")

return L

else:

with open('process\_follow\_'+name,"w") as f:

L = [0] \* length

f.write("".join(str(i) for i in L))

return L

#废

def write\_to\_file(url\_list,name\_list,url\_token\_list,follower\_count\_list):

with open('start\_from\_' + name + '.csv','a',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

for i in range(len(url\_list)):

spamwriter.writerow([name\_list[i],url\_list[i],url\_token\_list[i],follower\_count\_list[i]])

return True

def get\_length(url):

'''

获得这个用户关注的人总共有多少页

'''

response = requests.get(url,headers = headers)

content = str(response.content).encode("utf-8").decode("unicode\_escape").encode("utf-8").decode("unicode\_escape")

try:

return int(re.findall(re.compile('"totals": ([\w\W]+?),'),content)[0])//20+1

except:

return 0

# if (re.findall(re.compile('"totals": ([\w\W]+?),'),content) !=[]):

# n = int(re.findall(re.compile('"totals": ([\w\W]+?),'),content)[0])//20

# if(n != 0 ):

# return n+1

# else:

# return 0

def do\_it(url,pos):

'''

线程中做的事情","输入一个url之后抓取它关注的用户

'''

content = to\_get\_html(url)

# print(url)

url\_list, name\_list, url\_token\_list, follower\_count\_list = get\_all(content)

# print(name\_list)

# print(url\_list)

# print(url\_token\_list)

# print("-"\*100)

# print(len(name\_list))

# print(len(url\_list))

# print(len(url\_token\_list))

# print(follower\_count\_list)

for i in range(len(name\_list)):

all.append((name\_list[i], url\_list[i], url\_token\_list[i]))

# print(all)

# if(write\_to\_file(url\_list,name\_list,url\_token\_list,follower\_count\_list)):

# finall\_L[pos-1] = 1

def write\_back(L,name):

'''

这个其实也没有用了","因为现在无法保存进展了","所以最好一次解决问题。

'''

with open('process\_follow\_'+name,"w") as f:

f.write(" ".join(str(i) for i in L))

return True

def get\_follower(name):

'''

输入用户唯一的ID获得它关注的人

因为用户名有重复的","所以这里的name是唯一的键ID。

'''

url1 ="https://www.zhihu.com/api/v4/members/"+ name +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset=0&limit=20"

length = get\_length(url1)

# L = init\_file(length,name)

threads = []

# finall\_L = copy.deepcopy(L)

start = 1

end = 10

flag = 0

if (length and length!= 0):

for i in range(0,length):

url = "https://www.zhihu.com/api/v4/members/"+ name +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset="+ str(i \* 20) +"&limit=20"

t = threading.Thread( target = do\_it, args = (url,i) )

threads.append(t)

if(i % 20 == 0):

for th in threads:

th.setDaemon(True)

th.start()

for th in threads:

th.join()

threads = []

time.sleep(random.randint(5,15))

for th in threads:

th.setDaemon(True)

th.start()

for th in threads:

th.join()

if \_\_name\_\_ == '\_\_main\_\_':

'''

通过从下厨房或者何明科开始以一个广度优先的形式进行抓取它们关注的用户

这里主要是做一个稀疏矩阵出来

以下是稀疏矩阵的例子","注意这是一个非对称的矩阵","关注了为1","没关注为0","（自己和自己无所谓关注不关注）

比如

在这里下厨房关于了何明科","

而何明科没有关注下厨房","何明科关注了小明

小明关注了下厨房和何明科

|name |下厨房|何明科| 小明|

|下厨房| x | 1 | 0 |

|何明科| 0 | x | 1 |

|小明 | 1 | 1 | x |

上面这一个稀疏矩阵也是file这一个变量存的东西file[0] = [name,下厨房","何明科","小明","……]

然后这里的page是打算记录某个用户的关注的人的一个矩阵","因为是按照顺序来的","所以直接取几个用户就可以了","和all其实是一样的","但是page是作为一个全局的表可以随时进行查询的。也就是说file是它的一个子集。

'''

if not os.path.isdir("file"):

os.mkdir("file")

os.chdir("file")

names = ['he-ming-ke', 'xia-chu-fang']

s = 0

#name = "he-ming-ke"

# name = "he-ming-ke"

first\_part = "https://www.zhihu.com/people/"

third\_part = "/activities"

while(s<2):

name = names[s]

get\_follower(name)

page = []

file = []

i\_list = []

file.append(['name'])

n = 0

tmp = copy.deepcopy(all)

all = []

page.append(tmp)

with open('start\_from\_' + name + '\_user.csv','w',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

spamwriter.writerow(["name","id","关注者数","关注数","回答数","提问数","文章数","专栏数","认证情况","是否优秀回答者","知乎认证回答数","认证文章数","赞数","感谢数","收藏数","专业认证数"])

# print()

# spamwriter.writerow([page[0][0][0] , page[0][0][2]] + spiders(first\_part + page[0][0][2] + third\_part))

m = 0

while(True):

print(len(file))

print(len(page))

if(len(file) == 1000):

break

for i in page[n]:

with open('start\_from\_' + name + '\_user.csv','a+',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

# t = spiders(first\_part + i[2] + third\_part)

# print([i[0] , i[2] , first\_part + i[2] + third\_part])

spamwriter.writerow([i[0] , i[2] , first\_part + i[2] + third\_part])

get\_follower(i[2])

if (all not in page and all is not None):

page.append(all)

if(len(file) < 1000):

file[0].append(i[2])#把名字添加到第一列

file.append([i[2]])

i\_list.append(i)

else:

break

all = []

n += 1

for i in i\_list:

m+=1

for k in range(1,len(file)):

if(i not in page[k]):

file[m].append(0)

else:

file[m].append(1)

file[0].append(name)

file.append([name])

for i in range(len(file)):

if(i < len(tmp)):

file[m].append(1)

else:

file[m].append(0)

#get\_follower(i[2])

#为了防止中间抓到一半就断掉了","所以每次都存储起来。

#其实如果为了后面还有些数据的需要","最好也把page给保存起来

with open('start\_from\_' + name + '.csv','a+',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

for i in file:

spamwriter.writerow(i)

s+=1

**ip.py文件**

ip = ['111.177.190.110:9999',

'163.204.241.118:9999',

'111.177.167.87:9999',

'110.52.235.234:9999',

'119.102.25.100:9999',

'119.102.25.242:9999',

'111.177.182.50:9999',

'112.87.71.69:9999',

'111.177.185.57:9999',

'58.55.206.25:9999',

'111.177.189.165:9999',

'222.189.190.127:9999',

'110.73.41.9:8123',

'111.177.190.128:9999',

'119.102.25.119:9999',

'111.177.160.233:9999',

'183.148.154.84:9999',

'110.52.235.115:9999',

'111.177.178.239:9999',

'111.177.175.134:9999',

'111.177.190.105:9999',

'111.177.162.120:9999',

'111.177.191.140:9999',

'111.177.161.21:9999',

'111.177.164.56:9999',

'116.209.59.49:9999',

'125.126.194.70:9999',

'111.177.167.144:9999',

'218.91.112.137:9999',

'119.102.29.118:9999',

'111.177.187.179:9999',

'1.197.204.142:9999',

'119.102.29.251:9999',

'1.198.73.121:9999',

'111.177.171.166:9999',

'119.102.29.89:9999',

'112.85.169.100:9999',

'218.87.239.177:9999',

'125.126.214.253:9999',

'218.91.112.36:9999',

'111.177.161.238:9999',

'111.177.167.191:9999',

'111.177.173.16:9999',

'111.177.187.99:9999',

'111.177.169.123:9999',

'121.233.251.85:9999',

'111.177.190.177:9999',

'121.63.199.111:9999',

'110.52.235.241:9999',

'119.102.188.73:9999',

'119.102.190.20:9999',

'111.177.174.87:9999',

'119.102.189.118:9999',

'113.121.23.162:9999',

'119.102.188.230:9999',

'116.209.56.203:9999',

'111.177.178.101:9999',

'111.177.185.56:9999',

'111.177.179.18:9999',

'183.148.156.54:9999',

'111.177.180.197:9999',

'1.197.203.34:9999',

'222.188.178.4:9999',

'171.41.86.61:9999',

'171.41.84.108:9999',

'111.177.176.51:9999',

'171.41.84.139:9999',

'111.177.163.42:9999',

'111.177.178.13:9999',

'111.177.190.86:9999',

'111.177.177.234:9999',

'111.177.178.27:9999',

'171.41.82.220:9999',

'111.177.179.61:9999',

'171.41.80.183:9999',

'49.70.33.32:9999',

'111.177.190.117:9999',

'163.204.241.10:9999',

'114.104.135.220:9999',

'111.177.163.145:9999',

'49.86.177.38:9999',

'111.177.181.229:9999',

'111.177.174.57:9999',

'183.148.137.0:9999',

'171.83.167.182:9999',

'111.226.211.24:8118',

'163.204.240.202:9999',

'111.177.168.29:9999',

'111.177.190.73:9999',

'117.62.126.118:9999',

'111.177.181.148:9999',

'111.177.160.21:9999',

'110.52.235.233:9999',

'111.177.168.75:9999',

'111.177.179.148:9999',

'111.177.170.151:9999',

'111.177.178.230:9999',

'111.177.186.83:9999',

'111.177.172.59:9999',

'111.177.186.83:9999',

'111.177.172.59:9999',

'110.73.42.140:8123',

'111.177.184.145:9999',

'111.177.167.251:9999',

'111.177.164.77:9999',

'111.177.165.26:9999',

'111.177.169.243:9999',

'111.177.185.222:9999',

'111.177.163.177:9999',

'1.197.203.112:9999',

'221.235.234.138:9999',

'111.177.180.167:9999',

'111.177.185.48:9999',

'111.177.165.30:9999',

'119.102.185.69:9999',

'111.177.189.190:9999',

'111.177.183.15:9999',

'119.102.185.138:9999',

'183.148.147.21:9999',

'183.148.153.49:9999',

'111.177.162.7:9999',

'111.177.167.213:9999',

'111.177.160.60:9999',

'111.177.165.192:9999',

'180.116.59.33:9999',

'163.204.242.136:9999',

'111.177.183.41:9999',

'111.177.178.143:9999',

'111.177.161.89:9999',

'111.177.182.42:9999',

'111.177.175.51:9999',

'111.177.166.80:9999',

'111.177.163.45:9999',

'111.177.172.18:9999',

'58.219.63.38:9999',

'111.177.167.65:9999',

'111.177.160.25:9999',

'111.177.163.140:9999',

'111.177.181.253:9999',

'111.177.175.100:9999',

'111.177.165.13:9999',

'1.197.204.109:9999',

'111.177.184.132:9999',

'111.177.179.249:9999',

'111.177.163.54:9999',

'119.99.45.220:9999',

'111.177.186.214:9999',

'125.126.211.67:9999',

'111.177.160.62:9999',

'111.177.189.249:9999',

'183.148.135.109:9999',

'111.177.169.236:9999',

'112.85.164.241:9999',

'171.83.165.204:9999',

'111.177.183.97:9999',

'121.61.1.36:9999',

'171.112.168.198:9999',

'110.52.235.52:9999',

'112.85.169.16:9999',

'218.87.192.153:9999',

'110.52.235.190:9999',

'171.112.166.8:9999',

'163.204.246.127:9999',

'121.61.24.104:9999',

'116.208.52.235:9999',

'171.112.165.127:9999',

'121.63.198.70:9999',

'171.112.165.81:9999',

'171.112.165.199:9999',

'110.52.235.108:9999',

'1.198.72.12:9999',

'123.163.97.51:9999',

'171.112.165.57:9999',

'116.208.54.139:9999',

'116.209.52.140:9999',

'111.177.162.208:9999',

'116.208.53.20:9999',

'171.83.166.185:9999',

'116.209.52.108:9999',

'115.203.122.74:9999',

'171.80.136.121:9999',

'111.177.185.33:9999',

'122.193.246.0:9999',

'111.177.162.199:9999',

'111.177.184.195:9999',

'111.177.171.240:9999',

'111.177.180.245:9999',

'111.177.165.21:9999',

'27.29.46.108:9999',

'27.29.44.158:9999',

'222.189.191.45:9999',

'111.177.161.240:9999',

'222.189.191.45:9999',

'111.177.161.240:9999',

'110.52.235.30:9999',

'121.233.251.245:9999',

'119.102.128.135:9999',

'117.91.232.136:9999',

'183.148.149.158:9999',

'112.87.69.226:9999',

'113.121.22.240:9999',

'183.148.139.127:9999',

'115.53.23.56:9999',

'116.208.54.10:9999',

'119.102.24.200:9999',

'119.102.24.34:9999',

'119.102.28.195:9999',

'119.102.29.78:9999',

'119.102.29.153:9999',

'116.209.55.120:9999',

'119.102.29.253:9999',

'119.102.29.68:9999',

'125.126.219.61:9999',

'112.85.131.107:9999',

'171.80.137.35:9999',

'171.83.166.219:9999',

'116.209.52.80:9999',

'27.189.129.188:53281',

'171.83.167.191:9999',

'119.102.188.210:9999',

'115.53.16.65:9999',

'119.102.190.153:9999',

'119.102.188.229:9999',

'119.102.188.182:9999',

'119.102.188.166:9999',

'125.126.194.178:9999',

'112.85.130.123:9999',

'183.148.144.29:9999',

'59.37.33.62:54474',

'218.87.192.183:9999',

'112.87.69.227:9999',

'122.193.247.87:9999',

'112.87.68.122:9999',

'219.138.47.41:9999',

'171.41.81.140:9999',

'58.245.17.176:80',

'171.41.82.132:9999',

'112.85.166.18:9999',

'112.85.173.183:9999',

'49.86.181.209:9999',

'171.41.80.210:9999',

'171.41.80.141:9999',

'117.91.232.77:9999',

'171.41.81.11:9999',

'112.85.130.227:9999',

'58.50.3.28:9999',

'171.80.3.226:9999',

'183.148.157.22:9999',

'36.26.221.164:9999',

'222.189.190.146:9999',

'125.126.219.211:9999',

'218.91.112.104:9999',

'49.86.181.198:9999',

'183.148.136.149:9999',

'119.102.185.85:9999',

'119.102.184.173:9999',

'119.102.185.72:9999',

'119.102.184.244:9999',

'119.102.185.133:9999',

'171.83.166.114:9999',

'119.102.185.28:9999',

'110.52.235.125:9999',

'119.102.184.204:9999',

'112.85.170.186:9999',

'116.209.55.108:9999',

'112.85.130.166:9999',

'202.109.163.175:9999',

'117.62.46.31:9999',

'112.85.128.244:9999',

'121.63.199.1:9999',

'125.126.210.65:9999',

'112.87.71.56:9999',

'110.52.235.51:9999',

'119.102.185.32:9999',

'36.26.205.165:9999',

'58.55.196.98:9999',

'125.126.215.225:9999',

'119.102.185.100:9999',

'171.80.139.26:9999',

'171.83.166.45:9999',

'116.209.52.201:9999',

'116.209.53.86:9999',

'49.89.143.195:9999',

'112.87.70.41:9999',

'112.85.169.121:9999',

'218.87.192.195:9999',

'119.102.173.220:9999',

'115.53.37.111:9999',

'183.148.156.182:9999',

'115.223.101.88:8010',

'112.87.69.142:9999',

'116.209.54.6:9999',

'110.52.235.62:9999',

'113.121.22.1:9999',

'116.209.53.31:9999',

'111.176.20.157:9999',

'125.104.48.60:9999',

'119.102.173.192:9999',

'163.204.245.58:9999',

'110.52.235.219:9999',

'119.101.105.202:9999',

'163.204.245.14:9999',

'112.87.70.37:9999',

'223.241.78.194:8010',

'171.37.152.24:8123',

'116.209.57.105:9999',

'125.126.200.125:9999',

'113.121.22.16:9999',

'183.148.128.124:9999',

'112.87.68.237:9999',

'114.239.3.192:808',

'183.143.33.168:61234',

'116.208.53.184:9999',

'183.148.130.120:9999',

'110.52.235.113:9999',

'117.85.48.81:9999',

'116.208.53.11:9999',

'110.52.235.139:9999',

'116.209.55.94:9999',

'113.54.223.244:8118',

'222.189.190.204:9999',

'171.80.152.54:9999',

'36.26.225.24:9999',

'117.91.232.172:9999',

'112.85.168.236:9999',

'116.208.11.87:9999',

'116.209.54.254:9999',

'27.29.45.83:9999',

'110.52.235.148:9999',

'27.29.45.45:9999',

'27.29.44.179:9999',

'125.126.206.136:9999',

'27.29.44.87:9999',

'119.102.130.87:9999',

'110.52.235.145:9999',

'119.102.130.37:9999',

'119.102.130.193:9999',

'119.102.129.161:9999',

'110.52.235.127:9999',

'111.177.164.228:9999',

'119.102.129.12:9999',

'119.102.130.154:9999',

'119.102.128.163:9999',

'119.102.129.193:9999',

'119.102.130.66:9999',

'119.102.129.15:9999',

'222.189.191.134:9999',

'163.204.246.245:9999',

'116.209.55.230:9999',

'119.102.25.122:9999',

'110.52.235.174:9999',

'112.85.173.193:9999',

'111.177.181.59:9999',

'119.102.29.248:9999',

'119.102.29.119:9999',

'112.87.69.185:9999',

'117.80.137.74:9999',

'119.102.28.205:9999',

'163.204.241.124:9999',

'111.177.181.125:9999',

'111.177.185.215:9999',

'111.177.182.87:9999',

'111.177.175.7:9999',

'112.85.148.205:9999',

'218.87.239.133:9999',

'111.177.189.199:9999',

'110.52.235.10:9999',

'119.102.189.87:9999',

'183.148.136.166:9999',

'119.102.188.232:9999',

'110.52.235.84:9999',

'119.102.189.130:9999',

'119.102.189.248:9999',

'119.102.189.6:9999',

'112.85.171.149:9999',

'119.102.189.51:9999',

'112.87.70.168:9999',

'119.102.189.221:9999',

'116.209.58.235:9999',

'119.102.189.182:9999',

'111.79.198.155:9999',

'218.87.239.97:9999',

'110.52.235.133:9999',

'119.102.188.107:9999',

'119.102.188.169:9999',

'111.72.154.21:53128',

'119.102.188.91:9999',

'119.102.188.17:9999',

'171.41.84.213:9999',

'163.204.242.23:9999',

'183.148.135.221:9999',

'112.85.167.166:9999',

'119.101.104.170:9999',

'218.87.192.167:9999',

'223.241.118.132:18118',

'59.52.176.54:8118',

'183.148.134.215:9999',

'171.41.82.58:9999',

'171.41.81.226:9999',

'171.41.81.132:9999',

'112.85.130.240:9999',

'112.85.128.175:9999',

'112.85.168.98:9999',

'171.41.81.7:9999',

'171.41.80.202:9999',

'163.204.242.101:9999',

'112.85.130.180:9999',

'120.27.204.186:80',

'183.148.135.78:9999',

'110.52.235.129:9999',

'110.52.235.180:9999',

'183.148.135.203:9999',

'110.52.235.196:9999',

'163.204.246.58:9999',

'119.102.132.151:9999',

'125.126.196.48:9999',

'121.61.2.230:9999',

'119.102.133.73:9999',

'125.126.197.7:9999',

'59.32.37.47:808',

'112.85.130.83:9999',

'171.80.139.192:9999',

'116.209.57.85:9999',

'112.85.174.67:9999',

'183.148.137.170:9999',

'171.80.2.118:9999',

'183.148.134.28:9999',

'111.177.169.52:9999',

'110.52.235.167:9999',

'112.85.169.102:9999',

'218.87.192.155:9999',

'111.177.162.234:9999',

'115.53.33.194:9999',

'221.235.236.79:9999',

'110.52.235.64:9999',

'125.126.199.168:9999',

'221.233.46.216:9999',

'112.85.170.91:9999',

'112.87.68.35:9999',

'112.85.130.3:9999',

'119.99.45.123:9999',

'116.209.55.141:9999',

'112.85.130.212:9999',

'119.99.45.40:9999',

'110.52.235.246:9999',

'115.53.22.4:9999',

'110.52.235.70:808',

'111.177.183.235:9999',

'111.177.187.172:9999',

'111.177.162.241:9999',

'114.230.69.14:9999',

'223.241.116.196:8010',

'171.80.139.214:9999',

'117.91.255.59:9999',

'112.85.148.68:9999',

'116.208.52.65:9999',

'112.85.166.81:9999',

'111.177.182.188:9999',

'111.177.168.253:9999',

'111.177.161.18:9999',

'111.177.164.248:9999',

'111.177.186.146:9999',

'171.83.165.203:9999',

'119.101.104.250:9999',

'183.148.133.5:9999',

'119.29.138.103:8080',

'171.112.165.133:9999',

'111.177.176.83:9999',

'111.177.161.33:9999',

'171.112.164.238:9999',

'111.177.176.156:9999',

'111.177.170.255:9999',

'111.177.165.27:9999',

'218.87.239.229:9999',

'110.52.235.222:9999',

'171.80.112.232:9999',

'110.52.235.222:9999',

'171.80.112.232:9999',

'163.204.246.247:9999',

'111.177.171.94:9999',

'111.177.189.252:9999',

'111.177.160.120:9999',

'112.85.171.32:9999',

'119.102.128.158:9999',

'119.102.128.104:9999',

'119.102.129.87:9999',

'116.208.54.21:9999',

'111.177.167.200:9999',

'163.204.246.162:9999',

'119.102.24.210:9999',

'119.102.24.182:9999',

'119.102.24.179:9999',

'119.102.29.228:9999',

'110.52.235.5:9999',

'112.85.164.77:9999',

'110.52.235.55:9999',

'125.126.195.30:9999',

'110.73.4.245:8123',

'119.102.28.216:9999',

'125.126.216.126:9999',

'218.87.192.242:9999',

'121.233.207.31:9999',

'110.52.235.251:9999',

'125.126.193.18:9999',

'171.83.166.25:9999',

'175.0.36.108:8118',

'163.204.240.49:9999',

'115.216.119.174:808',

'202.109.163.246:9999',

'119.102.189.236:9999',

'111.177.170.198:9999',

'111.177.166.177:9999',

'110.52.235.112:9999',

'171.80.2.198:9999',

'112.85.129.87:9999',

'171.41.84.203:9999',

'111.177.183.173:9999',

'183.148.159.75:9999',

'110.52.235.225:9999',

'171.41.121.117:9999',

'112.87.69.12:9999',

'110.52.235.163:9999',

'182.92.113.183:8118',

'171.41.84.125:9999',

'112.85.131.41:9999',

'218.87.239.247:9999',

'110.52.235.15:9999',

'112.85.151.153:9999',

'183.148.151.86:9999',

'110.52.235.93:9999',

'1.192.242.148:9999',

'112.85.130.217:9999',

'171.41.81.27:9999',

'119.5.35.196:53128',

'112.87.69.139:9999',

'171.41.80.213:9999',

'171.41.80.170:39471',

'116.209.57.87:9999',

'125.126.209.8:9999',

'114.230.69.187:9999',

'218.87.239.89:9999',

'112.85.169.129:9999',

'110.52.235.4:9999',

'116.209.57.154:9999',

'113.121.243.17:808',

'112.85.170.183:9999',

'115.203.120.20:9999',

'110.52.235.36:9999',

'221.233.46.191:9999',

'111.177.174.236:9999',

'116.209.63.80:9999',

'110.52.235.212:9999',

'114.230.69.238:9999',

'111.177.188.35:9999',

'112.85.166.116:9999',

'111.177.173.126:9999',

'110.52.235.151:9999',

'111.177.169.156:9999',

'119.102.185.145:9999',

'116.76.102.89:80',

'119.102.184.200:9999',

'119.102.184.189:9999',

'111.177.165.145:9999',

'125.126.197.96:9999',

'116.209.54.2:9999',

'183.148.146.206:9999',

'125.126.202.133:9999',

'115.203.97.103:9999',

'121.61.2.169:9999',

'171.83.165.139:9999',

'110.52.235.44:9999',

'112.85.165.16:9999',

'110.52.235.44:9999',

'112.85.165.16:9999',

'125.126.214.135:9999',

'112.87.69.148:9999',

'110.52.235.38:9999',

'171.83.167.68:9999',

'112.87.69.146:9999',

'153.35.7.5:8118',

'163.204.246.249:9999',

'163.204.240.45:9999',

'163.204.242.253:8118',

'125.126.215.89:9999',

'171.80.2.146:9999',

'114.106.135.35:9999',

'180.119.68.108:9999',

'111.177.176.250:9999',

'171.83.166.146:9999',

'119.51.38.141:80',

'218.87.192.203:9999',

'110.73.9.30:8123',

'116.209.58.167:9999',

'171.80.152.175:9999',

'171.83.164.16:9999',

'112.85.167.168:9999',

'111.177.184.157:9999',

'122.230.250.7:8010',

'110.52.235.213:9999',

'112.85.170.228:9999',

'183.148.150.26:9999',

'112.85.171.155:9999',

'111.177.175.58:9999',

'222.189.191.166:9999',

'117.91.232.121:9999',

'115.53.18.110:9999',

'183.148.145.251:9999',

'111.177.180.29:9999',

'115.203.103.203:9999',

'112.85.167.68:9999',

'110.52.235.88:9999',

'125.126.197.46:9999',

'223.241.116.154:8010',

'121.61.0.12:9999',

'27.29.44.148:9999',

'171.80.138.225:9999',

'27.29.46.85:9999',

'121.31.157.11:8123',

'27.29.45.232:9999',

'183.148.135.226:9999',

'1.83.126.129:8118',

'183.148.139.74:9999',

'119.102.128.141:9999',

'112.87.68.186:9999',

'119.102.128.32:9999',

'120.79.210.183:8118',

'116.208.54.66:9999',

'119.102.128.51:9999',

'119.102.128.81:9999',

'112.85.129.215:9999',

'121.61.27.135:9999',

'123.180.69.68:8010',

'110.52.235.177:9999',

'218.91.112.148:9999',

'111.79.198.172:9999',

'183.148.139.8:9999',

'110.52.235.65:9999',

'183.148.140.162:9999',

'171.83.165.125:9999',

'111.177.165.12:9999',

'183.148.156.139:9999',

'183.148.148.130:9999',

'183.148.143.245:9999',

'111.177.191.138:9999',

'115.219.105.96:8010',

'111.177.166.16:9999',

'111.177.169.249:9999',

'171.80.138.212:9999',

'163.204.246.88:9999',

'112.85.168.254:9999',

'123.185.220.115:8118',

'116.208.53.97:9999',

'111.177.113.5:9999',

'110.52.235.169:9999',

'111.177.181.116:9999',

'171.41.84.230:9999',

'116.208.53.144:9999',

'111.177.107.189:9999',

'222.189.190.81:9999',

'171.83.165.130:9999',

'171.41.85.52:9999',

'171.41.85.65:9999',

'171.83.166.105:9999',

'171.41.85.185:9999',

'111.177.174.253:9999',

'111.177.170.186:9999',

'111.177.181.4:9999',

'111.177.191.152:9999',

'112.85.149.179:9999',

'60.182.112.152:8118',

'171.41.84.163:9999',

'116.208.52.222:9999',

'171.41.84.228:9999',

'125.126.203.185:9999',

'111.177.171.209:9999',

'171.83.164.193:9999',

'112.87.69.169:9999',

'110.52.235.193:9999',

'171.80.2.192:9999',

'183.148.132.88:9999',

'182.88.188.2:8123',

'171.41.82.66:9999',

'171.41.80.212:9999',

'171.41.80.238:9999',

'171.41.81.4:9999',

'171.41.80.145:9999',

'121.31.101.182:8123',

'171.41.80.101:9999',

'171.41.80.214:9999',

'112.85.175.172:9999',

'171.41.80.121:9999',

'183.148.138.52:9999',

'36.99.212.226:8010',

'121.61.26.42:9999',

'112.85.172.58:9999',

'125.126.199.151:9999',

'183.148.132.122:9999',

'218.87.239.131:9999',

'183.148.137.20:9999',

'114.239.145.72:808',

'110.52.235.8:9999',

'117.70.38.131:9999',

'121.63.198.225:9999',

'119.102.132.134:9999',

'119.102.133.179:9999',

'119.102.132.232:9999',

'119.102.133.169:9999',

'221.233.46.193:9999',

'122.237.107.134:80',

'120.83.107.148:9999',

'121.61.3.155:9999',

'112.85.172.89:9999',

'183.148.145.115:9999',

'112.85.174.39:9999',

'116.208.53.221:9999',

'119.102.186.99:9999',

'111.177.175.234:9999',

'112.85.131.64:9999',

'218.87.192.218:9999',

'49.70.64.189:9999',

'111.79.199.11:9999',

'171.80.112.21:9999',

'110.52.235.105:9999',

'111.177.113.248:9999',

'110.52.235.239:9999',

'125.126.202.149:9999',

'116.209.57.127:9999',

'121.61.2.224:9999',

'110.52.235.24:9999',

'218.87.192.194:9999',

'112.85.148.215:9999',

'112.85.173.249:9999',

'110.52.235.198:9999',

'110.52.235.182:9999',

'112.85.169.72:9999',

'122.193.244.244:9999',

'125.125.211.84:9999',

'110.52.235.136:9999',

'223.241.78.195:8010',

'125.126.206.222:9999',

'112.85.168.233:9999',

'117.91.232.196:9999',

'112.87.71.82:9999',

'123.146.236.34:53281',

'115.53.18.252:9999',

'110.52.235.53:9999',

'110.52.235.210:9999',

'121.61.3.23:9999',

'183.148.134.101:9999',

'60.217.64.237:31923',

'27.29.46.17:9999',

'27.29.44.241:9999',

'27.29.46.11:9999',

'27.29.46.7:9999',

'27.29.45.59:9999',

'183.148.159.210:9999',

'27.29.44.188:9999',

'27.29.45.92:9999',

'171.83.167.177:9999',

'119.102.130.125:9999',

'116.208.55.64:9999',

'111.77.197.177:9999',

'117.81.224.157:9999',

'183.148.148.68:9999',

'125.126.223.44:9999',

'119.102.25.98:9999',

'120.78.185.175:8118',

'218.87.192.133:9999',

'121.61.25.16:9999',

'112.85.130.179:9999',

'116.209.59.60:9999',

'121.61.1.219:9999',

'119.102.24.244:9999',

'119.102.25.91:9999',

'110.52.235.83:9999',

'119.102.25.14:9999',

'121.61.24.198:9999',

'119.102.24.121:9999',

'119.102.25.21:9999',

'125.126.220.101:9999',

'183.148.156.119:9999',

'116.209.58.108:9999',

'112.85.166.41:9999',

'119.102.189.61:9999',

'218.87.192.68:9999',

'112.85.129.217:9999',

'121.61.1.33:9999',

'119.102.189.80:9999',

'171.41.86.123:9999',

'171.41.85.159:9999',

'171.41.84.204:9999',

'171.41.84.105:9999',

'112.85.167.183:9999',

'125.126.218.27:9999',

'116.208.11.123:9999',

'121.233.207.153:9999',

'125.126.206.97:9999',

'112.85.164.63:9999',

'112.85.171.194:9999',

'125.126.201.44:9999',

'112.85.166.65:9999',

'117.91.232.188:9999',

'123.206.79.196:8118',

'111.77.197.221:9999',

'171.41.81.72:9999',

'171.41.82.190:9999',

'183.148.152.91:9999',

'121.61.1.67:9999',

'180.119.141.144:9999',

'112.85.164.93:9999',

'116.209.59.64:9999',

'182.122.189.15:9999',

'175.148.75.246:1133',

'111.177.113.81:9999',

'119.102.133.89:9999',

'116.209.55.252:9999',

'183.148.138.7:9999',

'125.104.50.177:9999',

'183.148.134.210:9999',

'183.148.129.31:9999',

'116.209.52.14:9999',

'183.148.154.160:9999',

'119.102.133.171:9999',

'113.121.23.126:9999',

'112.87.71.235:9999',

'115.53.38.139:9999',

'112.85.167.78:9999',

'116.209.56.106:9999',

'183.148.157.13:9999',

'114.230.69.126:9999',

'125.126.215.155:9999',

'218.87.239.139:9999',

'218.87.239.138:9999',

'125.126.197.230:9999',

'112.85.165.46:9999',

'111.77.196.233:9999',

'121.61.1.77:9999',

'111.177.181.67:9999',

'183.148.148.252:9999',

'116.209.56.109:9999',

'183.148.155.88:9999',

'116.209.58.54:9999',

'183.148.146.230:9999',

'183.148.152.122:9999',

'171.112.169.41:9999',

'112.85.128.209:9999',

'59.32.37.28:8010',

'221.235.238.115:9999',

'116.209.53.94:9999',

'163.204.242.209:9999',

'125.126.222.0:9999',

'163.204.242.209:9999',

'125.126.222.0:9999',

'218.87.192.146:9999',

'125.126.221.60:9999',

'111.176.30.188:9999',

'222.189.191.196:9999',

'112.85.171.40:9999',

'112.85.172.113:9999',

'110.52.235.61:9999',

'125.126.211.54:9999',

'27.29.46.162:9999',

'27.29.45.87:9999',

'27.29.45.21:9999',

'27.29.45.219:9999',

'27.29.46.149:9999',

'27.29.45.202:9999',

'115.53.18.111:9999',

'27.29.45.15:9999',

'180.119.141.84:9999',

'27.29.45.18:9999',

'27.29.44.252:9999',

'110.52.235.99:9999',

'27.29.45.134:9999',

'27.29.45.82:9999',

'27.29.44.224:9999',

'218.87.192.210:9999',

'27.29.45.79:9999',

'121.63.199.8:9999',

'27.29.44.237:9999',

'58.50.2.9:9999',

'27.29.44.222:9999',

'116.209.56.17:9999',

'183.148.134.41:9999',

'111.177.189.194:9999',

'112.85.149.252:9999',

'121.61.3.228:9999',

'110.52.235.232:9999',

'119.102.129.19:9999',

'119.102.129.119:9999',

'125.126.192.240:9999',

'111.177.180.119:9999',

'223.215.186.173:9999',

'110.52.235.172:9999',

'171.80.175.57:9999',

'116.209.59.5:9999',

'125.125.217.28:9999',

'112.85.164.67:9999',

'121.31.101.0:8123',

'125.126.192.212:9999',

'112.85.171.160:9999',

'163.204.247.30:9999',

'111.177.179.156:9999',

'49.86.176.151:9999',

'114.230.69.177:9999',

'171.80.0.5:9999',

'171.80.136.10:9999',

'183.148.130.184:9999',

'125.126.195.102:9999',

'112.87.70.80:9999',

'110.52.235.244:9999',

'110.52.235.63:9999',

'110.52.235.100:9999',

'119.102.26.109:9999',

'112.85.129.97:9999',

'116.208.55.9:9999',

'171.38.42.103:8118',

'116.209.54.241:9999',

'116.209.57.156:9999',

'119.102.24.238:9999',

'119.102.25.79:9999',

'110.52.235.179:9999',

'60.173.203.83:47300',

'116.209.56.239:9999',

'110.52.235.85:9999',

'119.102.28.232:9999',

'112.85.171.103:9999',

'119.102.29.84:9999',

'125.126.202.113:9999',

'119.102.29.165:9999',

'125.126.204.105:9999',

'121.61.0.169:9999',

'112.85.130.114:9999',

'124.167.36.240:80',

'125.126.219.2:9999',

'125.126.221.161:9999',

'122.193.247.75:9999',

'36.26.226.114:9999',

'218.91.112.209:9999',

'110.52.235.80:9999',

'116.209.55.219:9999',

'112.87.69.174:9999',

'122.193.244.235:9999',

'183.148.147.95:9999',

'36.26.207.69:9999',

'110.52.235.143:9999',

'117.70.39.86:8118',

'121.61.3.162:9999',

'112.87.68.232:9999',

'111.177.177.175:9999',

'119.102.189.34:9999',

'183.148.135.33:9999',

'125.126.196.2:9999',

'112.85.128.165:9999',

'112.85.169.52:9999',

'171.41.84.144:9999',

'116.209.56.2:9999',

'125.126.212.72:9999',

'116.209.54.127:9999',

'111.79.198.127:9999',

'218.87.192.200:9999',

'116.209.55.241:9999',

'112.85.169.206:9999',

'116.209.58.251:9999',

'125.126.220.50:9999',

'116.208.53.247:9999',

'111.77.197.181:9999',

'125.126.217.159:9999',

'222.189.190.87:9999',

'112.85.175.119:9999',

'119.102.132.104:9999',

'183.148.148.98:9999',

'183.157.168.251:8118',

'116.209.52.45:9999',

'202.109.163.202:9999',

'116.209.55.204:9999',

'112.85.129.88:9999',

'222.189.191.246:9999',

'115.203.97.72:9999',

'112.87.69.149:9999',

'111.177.176.178:9999',

'112.85.129.202:9999',

'183.148.134.136:9999',

'110.52.235.23:9999',

'112.85.170.208:9999',

'183.148.150.153:9999',

'183.148.158.7:9999',

'112.85.151.119:9999',

'119.102.173.70:9999',

'112.87.70.246:9999',

'116.209.58.179:9999',

'112.85.131.57:9999',

'112.85.131.238:9999',

'116.209.53.35:9999',

'116.209.56.84:9999',

'125.126.218.174:9999',

'116.209.54.146:9999',

'125.126.205.233:9999',

'112.87.70.141:9999',

'125.126.198.247:9999',

'183.148.156.247:9999',

'112.85.170.84:9999',

'183.148.158.148:9999',

'111.77.197.127:9999',

'183.148.128.190:9999',

'183.148.146.199:9999',

'110.52.235.201:9999',

'116.209.53.228:9999',

'125.126.203.92:9999',

'125.126.215.228:9999',

'116.209.52.54:9999',

'116.209.52.101:9999',

'116.209.52.121:9999',

'163.204.242.82:9999',

'223.215.187.225:9999',

'112.87.71.154:8090',

'112.85.131.93:9999',

'112.87.70.125:9999',

'27.29.45.64:9999',

'116.209.56.38:9999',

'112.85.128.141:9999',

'218.87.192.137:9999',

'27.29.45.54:9999',

'125.126.195.241:9999',

'110.52.235.200:9999',

'125.126.198.133:9999',

'111.177.189.31:9999',

'111.177.178.229:9999',

'119.102.130.106:9999',

'116.209.58.109:9999',

'112.85.128.66:9999',

'110.52.235.218:9999',

'119.102.128.255:9999',

'223.241.116.121:8010',

'223.241.116.129:8010',

'119.102.129.152:9999',

'112.85.131.225:9999',

'119.102.129.7:9999',

'119.102.128.235:9999',

'119.102.128.229:9999',

'116.208.55.219:9999',

'183.148.148.189:9999',

'183.148.158.9:9999']

**Update\_ip.py文件**

import requests

import re

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36'

}

def update\_ip(start,end):

'''

更新ip池

'''

with open("ip.py","w") as f:

f.write("ip = [")

for j in range(start,end + 1):

url = "https://www.xicidaili.com/wt/" + str(j)

content = requests.get(url,headers = headers).text

tr = re.compile("<tr class[\w\W]+?>([\w\W]+?)</tr>").findall(content)

L = []

for i in range(len(tr)):

a = re.compile("<td>([\w\W]+?)</td>").findall(tr[i])

b = re.compile("<td class[\w\W]+?>([\w\W]+?)</td>").findall(tr[i])

# print(a[0] + a[1] + a[3])

# print(b[1])

if(b[1] == '高匿' and a[3] == "HTTP"):

if(i < len(tr)-1 or j != end):

f.write("'" + a[0] + ":" + a[1] + "',\n")

else:

f.write("'" + a[0] + ":" + a[1] + "']")

if \_\_name\_\_ == '\_\_main\_\_':

update\_ip(10,20)

Topic\_attr.py:

from zhihu import spiders

import os

import time

import random

import csv

from follow import get\_length

from follow import to\_get\_html

from follow import get\_all

import threading

import re

import sys

all = []

topic\_name = "社会"

def to\_do(name1):

with open(name1,"r") as f:

data = f.readlines()[0:]

print(data)

for i in range(len(data)):

data[i] = data[i].strip().split(",")

page = {}

page1 = {}

for i in data:

page[i[1]] = i[0]

page1[i[1]] = i[0]

with open(topic\_name + "\_attr.csv","r") as f:

data = f.readlines()

for i in range(len(data)):

data[i] = data[i].strip().split(",")[1]

name = data

with open(name1 + "\_attr.csv",'w',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

spamwriter.writerow(["name","id","关注者数","关注数","回答数","提问数","文章数","专栏数","认证情况","是否优秀回答者","知乎认证回答数","认证文章数","赞数","感谢数","收藏数","专业认证数"])

for i in range(len(name)):

#print(page[name[i]])

t = spiders(page[name[i]])

spamwriter.writerow([page1[name[i]] , name[i] ] + t)

if((i + 1) % 20 == 0):

print(i)

time.sleep(random.randint(5,15))

def get\_all\_title(url\_token):

length = 0

url = "https://www.zhihu.com/api/v4/members/" + url\_token + "/following-topic-contributions?include=data%5B\*%5D.topic.introduction&offset=0&limit=20"

content = to\_get\_html(url)

if( "该帐号已停用，主页无法访问" in content):

return False

if(length == 0):

t = re.findall(re.compile('"totals": ([\w\W]+?),'),content)

if len(t) != 0:

length = int(t[0]) // 20 + 1

name = re.findall(re.compile('"name": "([\w\W]+?)"'),content)

# all\_topic = []

# all\_topic = all\_topic + name

if(topic\_name in name):

return True

for i in range(1,length):

url = "https://www.zhihu.com/api/v4/members/" + url\_token + "/following-topic-contributions?include=data%5B\*%5D.topic.introduction&offset=" + str(i \* 20) + "&limit=20"

content = to\_get\_html(url)

name = re.findall(re.compile('"name": "([\w\W]+?)"'),content)

# all\_topic = all\_topic + name

if(topic\_name in name):

return True

return False

# return all\_topic

def do\_it(url,pos):

content = to\_get\_html(url)

url\_list, name\_list, url\_token\_list, follower\_count\_list = get\_all(content)

# print(url\_token\_list)

# print(len(url\_list),len(name\_list),len(url\_token\_list),len(follower\_count\_list))

for i in range(len(url\_token\_list)):

if(int(follower\_count\_list[i]) > 1000):

all.append((url\_list[i],url\_token\_list[i]))

def get\_follower(name):

url1 ="https://www.zhihu.com/api/v4/members/"+ name +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset=0&limit=20"

length = get\_length(url1)

threads = []

flag = 0

if (length and length!= 0):

for i in range(0,length):

url = "https://www.zhihu.com/api/v4/members/"+ name +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset="+ str(i \* 20) +"&limit=20"

t = threading.Thread( target = do\_it, args = (url,i) )

threads.append(t)

if(i % 20 == 0):

for th in threads:

th.setDaemon(True)

th.start()

for th in threads:

th.join()

threads = []

time.sleep(random.randint(5,15))

for th in threads:

th.setDaemon(True)

th.start()

for th in threads:

th.join()

if \_\_name\_\_ == '\_\_main\_\_':

if(not os.path.isdir("file")):

os.mkdir("file")

os.chdir("./file")

id = "rouni"

t = ("https://www.zhihu.com/people/rouni/activities",id)

if(not os.path.isfile(topic\_name + "\_attr.csv")):

with open(topic\_name + "\_attr.csv",'w',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

# spamwriter.writerow(list(t))

if os.path.isfile("process") :

with open("process","r") as f:

data = f.readlines()

for i in data:

i = i.strip().split(",")

all.append((i[0],i[1]))

else:

get\_follower(id)

num = 1

now = all[0]

try:

for i in all:

now = i

all\_topic = get\_all\_title(i[1])

if all\_topic:

num += 1

print(i)

with open(topic\_name + "\_attr.csv",'a',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

spamwriter.writerow(list(i))

if num >= 100:

break

get\_follower(i[1])

except Exception as e:

print(sys.exc\_info()[0],sys.exc\_info()[1])

with open("process","w") as f:

pos = all.index(now)

for i in all[pos:]:

f.write(str(i[0]) + "," + str(i[1]) + "\n")

name1 = "社会\_attr.csv"

to\_do(name1)

**final.py文件**

#coding:utf-8

import pandas as pd

from zhihu import spiders

from follow import get\_length

from follow import get\_all

from follow import to\_get\_html

import os

import csv

import time

import random

# import threading

from threading import Thread

topic\_name = "社会"

def do\_it(url):

content = to\_get\_html(url)

url\_list, name\_list, url\_token\_list, follower\_count\_list = get\_all(content)

return url\_token\_list

class MyThread(Thread):

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

Thread.\_\_init\_\_(self)

self.url = url

def run(self):

self.result = do\_it(self.url)

def get\_result(self):

return self.result

def do(name,name2):

data = []

with open(name2,"r") as f:

a = f.readlines()

# print(a)

for i in a:

i = i.strip().split(",")

data.append(i[1])

if(not os.path.isfile("proces\_" + topic\_name)):

with open("proces\_" + topic\_name,"w") as f:

f.write("0")

with open("proces\_" + topic\_name,"r") as f:

start = int(f.readline().strip())

print(start)

end = start

try:

with open(name+'.csv','a',newline='') as csvfile:

spamwriter = csv.writer(csvfile,dialect='excel')

if(start == 0):

spamwriter.writerow(['name'] + data)

for i in data[start:]:

print(i)

url1 ="https://www.zhihu.com/api/v4/members/"+ i +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset=0&limit=20"

length = get\_length(url1)

print("length : ",length)

all = []

threads = []

for j in range(length):

url = "https://www.zhihu.com/api/v4/members/"+ i +"/followees?include=data%5B\*%5D.answer\_count%2Carticles\_count%2Cgender%2Cfollower\_count%2Cis\_followed%2Cis\_following%2Cbadge%5B%3F(type%3Dbest\_answerer)%5D.topics&offset="+ str(j \* 20) +"&limit=20"

# all = all + url\_token\_list

t = MyThread(url)

threads.append(t)

# if(len(threads) == 20):

# for th in threads:

# th.setDaemon(True)

# th.start()

# for th in threads:

# th.join()

# all = all + th.get\_result()

# threads = []

# time.sleep(random.randint(1,3))

for th in threads:

th.setDaemon(True)

th.start()

for th in threads:

th.join()

all = all + th.get\_result()

threads = []

L = [i]

for j in data:

if(j in all):

L.append(1)

else:

L.append(0)

spamwriter.writerow(L)

end += 1

time.sleep(random.randint(5,15))

finally:

with open("proces\_" + topic\_name,"w") as f:

f.write(str(end))

if \_\_name\_\_ == "\_\_main\_\_":

os.chdir("file")

name = "final\_" + topic\_name

name2 = topic\_name + "\_attr.csv"

do(name,name2)

**可视化.py文件**

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("social\_attr.csv\_attr - del(2).csv")

# In[3]:

fan\_value = data['fans'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

concern\_value = data['concern'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

answer\_value = data['answer'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

question\_value = data['question'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

article\_value = data['article'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

column\_value =data['column'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

if\_zhihu\_value = data['if\_zhihu'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

#num\_artic\_value = data['num\_artic'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

profession\_value = data['pro'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

mean\_value = data['mean'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

degree = data['degree'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

betweenness = data['betweenness'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

closenes = data['closeness'].sort\_values(ascending=False).to\_frame().reset\_index().reset\_index()

plt.bar(fan\_value['level\_0'],fan\_value['fans'],color='yellow', edgecolor='red', label='fans')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('fans.png')

plt.show()

# In[4]:

plt.bar(concern\_value['level\_0'],concern\_value['concern'],color='yellow', edgecolor='red', label='concern')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('concern.png')

plt.show()

# In[ ]:

plt.bar(answer\_value['level\_0'],answer\_value['answer'],color='yellow', edgecolor='red', label='answer')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('answer.png')

plt.show()

# In[6]:

plt.bar(question\_value['level\_0'],question\_value['question'],color='yellow', edgecolor='red', label='question')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('question.png')

plt.show()

# In[7]:

plt.bar(article\_value['level\_0'],article\_value['article'],color='yellow', edgecolor='red', label='article')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('article.png')

plt.show()

# In[8]:

plt.bar(column\_value['level\_0'],column\_value['column'],color='yellow', edgecolor='red', label='column')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('column.png')

plt.show()

# In[9]:

plt.bar(if\_zhihu\_value['level\_0'],if\_zhihu\_value['if\_zhihu'],color='yellow', edgecolor='red', label='if\_zhihu')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('if\_zhihu.png')

plt.show()

# In[11]:

plt.bar(profession\_value['level\_0'],profession\_value['pro'],color='yellow', edgecolor='red', label='profession')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('profession.png')

plt.show()

# In[12]:

plt.bar(mean\_value['level\_0'],mean\_value['mean'],color='yellow', edgecolor='red', label='mean')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('mean.png')

plt.show()

# In[13]:

plt.bar(degree['level\_0'],degree['degree'],color='yellow', edgecolor='red', label='degree')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('degree.png')

plt.show()

# In[14]:

plt.bar(closenes['level\_0'],closenes['closeness'],color='yellow', edgecolor='red', label='closeness')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('closeness.png')

plt.show()

# In[15]:

plt.bar(betweenness['level\_0'],betweenness['betweenness'],color='yellow', edgecolor='red', label='betweenness')

plt.legend(loc='upper right')

plt.xticks([])

plt.savefig('betweenness.png')

plt.show()

# In[16]:

y = np.random.randn(9)

plt.figure()

data = data.reset\_index()

n = 0

n\_new = 0

for i in data['if\_identify']:

if i == 1:

n+=1

elif i == 0:

n\_new+=1

s = 0

s\_new =0

for i in data['if\_super']:

if i == 1:

s+=1

elif i == 0:

s\_new+=1

table\_vals = [[n,n\_new],[s,s\_new]]

col\_labels = [1,0]

row\_labels=['if\_identify','if\_super']

row\_colors = ['red','gold']

my\_table = plt.table(cellText=table\_vals, colWidths=[0.2]\*2,rowLabels=row\_labels, colLabels=col\_labels,rowColours=row\_colors, colColours=row\_colors,loc='best')

# In[ ]:

**数据预处理**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib.ticker import NullFormatter

from sklearn.manifold import LocallyLinearEmbedding

from sklearn.manifold import Isomap

from sklearn.manifold import SpectralEmbedding

data = pd.read\_excel(r"C:/Users/Lenovo/Desktop/polic\_key.xlsx")

GRA = pd.read\_excel(r"C:/Users/Lenovo/Desktop/polic\_values.xlsx")

X = []

for i in range(97):

X.append(list(data.iloc[i]))

degree = GRA.degree

closeness = GRA.closeness

betweenness = GRA.betweenness

ratedeg = 0.22

degreecut = (degree/(max(degree)-min(degree)))> ratedeg

degreecut.apply(int)

rateclo = 0.72

closenesscut = (closeness/(max(closeness)-min(closeness)))> rateclo

closenesscut.apply(int)

ratebet = 0.038

betweennesscut = (betweenness/(max(betweenness)-min(betweenness)))> ratebet

betweennesscut.apply(int)

###PCA-3D

pca = PCA(n\_components = 3)

reduced = pca.fit\_transform(X)

xlabel = []

ylabel = []

zlabel = []

for i in range(97):

xlabel.append(reduced[i][0])

ylabel.append(reduced[i][1])

zlabel.append(reduced[i][2])

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = degreecut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = closenesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = betweennesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

###SpectralEmbedding-3D

SE = SpectralEmbedding(n\_components=3) #可修正n\_neighbors

reduced = SE.fit\_transform(X)

xlabel = []

ylabel = []

zlabel = []

for i in range(97):

xlabel.append(reduced[i][0])

ylabel.append(reduced[i][1])

zlabel.append(reduced[i][2])

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = degreecut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = closenesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = betweennesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

###isomap-3D

iso = Isomap(n\_components=3) #可修正n\_neighbors

reduced = iso.fit\_transform(X)

xlabel = []

ylabel = []

zlabel = []

for i in range(97):

xlabel.append(reduced[i][0])

ylabel.append(reduced[i][1])

zlabel.append(reduced[i][2])

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = degreecut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = closenesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = betweennesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

###LLE-2D

lle = LocallyLinearEmbedding(n\_components=3)

reduced = lle.fit\_transform(X)

xlabel = []

ylabel = []

zlabel = []

for i in range(97):

xlabel.append(reduced[i][0])

ylabel.append(reduced[i][1])

zlabel.append(reduced[i][2])

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = degreecut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = closenesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = betweennesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

###TSNE-3D

tsne = TSNE(n\_components=3)

reduced = tsne.fit\_transform(X)

xlabel = []

ylabel = []

zlabel = []

for i in range(97):

xlabel.append(reduced[i][0])

ylabel.append(reduced[i][1])

zlabel.append(reduced[i][2])

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = degreecut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = closenesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

fig = plt.figure(figsize=(10,10))

ax = fig.add\_subplot(211,projection = '3d')

ax.scatter(xlabel,ylabel,zlabel,c = betweennesscut,cmap=plt.cm.Spectral)

ax.view\_init(200,-200)

plt.show()

**特征之间的线性相关性**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('fivethirtyeight')

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

excel\_path = 'C:/Users/Lenovo/Desktop/culture\_attr.csv\_attr.csv'

file = open(excel\_path,'r')

data = pd.read\_csv(file)

data.head()

sns.heatmap(data.corr(),annot=True,cmap='RdYlGn',linewidths=0.5) #data.corr()->correlation matrix

fig=plt.gcf()

fig.set\_size\_inches(20,16)

plt.show()

#print(data)

**数据清洗**

#-\*- coding=utf-8 -\*-

import pandas as pd

import numpy as np

excel\_path = 'C:/Users/Lenovo/Desktop/social\_attr.csv\_attr - d.csv'

file = open(excel\_path,'r')

a = pd.read\_csv(file)

#a=pd.read\_csv(excel\_path，encoding='gbk')

#print(a)

#print(a)

#np.set\_printoptions(suppress=True)

train\_data = np.array(a)

print(train\_data)

train\_data.shape

print(train\_data[:,7])

a=[]

for i in train\_data[:,6]:

if i == '是':

a.append(1)

else:

a.append(0)

print(len(a))

print(a)

data\_write('culture.csv', a)

import xlwt

def data\_write(file\_path, data):

f = xlwt.Workbook()

sheet1 = f.add\_sheet(u'sheet1',cell\_overwrite\_ok=True) #创建sheet

#将数据写入第 i 行，第 j 列

i = 0

for i in range(len(data)):

sheet1.write(i,1,data[i])

f.save(file\_path) #保存文件

a=[]

b=[]

c=[]

a\_new=[]

b\_new=[]

c\_new=[]

for i in train\_data[:,11]:

a.append(i)

print(a)

for i in a:

i = (i-min(a))/(max(a)-min(a))

a\_new.append(i)

print(a\_new)

for i in train\_data[:,12]:

b.append(i)

#print(b)

for i in b:

i = (i-min(b))/(max(b)-min(b))

b\_new.append(i)

print(b\_new)

for i in train\_data[:,13]:

c.append(i)

#print(c)

for i in c:

i = (i-min(c))/(max(c)-min(c))

c\_new.append(i)

print(c\_new)

mean=[]

for i in range(99):

x=(a\_new[i]+b\_new[i]+c\_new[i])/3

mean.append(x)

print(mean)

data\_write('aaaaaaaaaaaaaaaaaaamean.csv', mean)

data\_write('aaaaaaaaaaaaaaaaaaa\_a\_new.csv', a\_new)

data\_write('aaaaaaaaaaaaaaaaaaab\_new.csv', b\_new)

data\_write('aaaaaaaaaaaaaaaaaaac\_new.csv', c\_new)

**数据建模分析**

###导入必须的库  
#功能库  
import numpy as np  
import pandas as pd  
#分析库  
from sklearn import metrics  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
from sklearn.model\_selection import KFold  
#算法库  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
from sklearn.ensemble import RandomForestRegressor  
#其他库  
import warnings  
#忽视警告  
warnings.filterwarnings("ignore")  
  
###导入数据并且进行分组  
data = pd.read\_excel(r"C:/Users/NHT/Desktop/social\_feature.xlsx")  
#data = data[['fans','mean']]  
data = data[['concern','answer','question','article','column','if\_identify']]  
#data = data[['if\_zhihu','if\_super','profession']]  
GRA = pd.read\_excel(r"C:/Users/NHT/Desktop/social\_result.xlsx")  
degree = GRA.degree  
closeness = GRA.closeness  
betweenness = GRA.betweenness  
ratedeg = 0.24  
degree = (degree/(max(degree)-min(degree)))> ratedeg  
degree.apply(int)  
ratebet = 0.046  
betweenness = (betweenness/(max(betweenness)-min(betweenness)))> ratebet  
betweenness.apply(int)  
rateclo = 0.75  
closeness = (closeness/(max(closeness)-min(closeness)))> rateclo  
closeness.apply(int)  
#基础分类器数据准备  
x\_train1, x\_test1, y\_train1, y\_test1 = train\_test\_split(data,degree, test\_size=20, random\_state=0)  
x\_train2, x\_test2, y\_train2, y\_test2 = train\_test\_split(data,closeness, test\_size=20, random\_state=0)  
x\_train3, x\_test3, y\_train3, y\_test3 = train\_test\_split(data,betweenness, test\_size=20, random\_state=0)  
  
###特征重要性提取  
rf = RandomForestRegressor()  
X = data  
y = degree  
rf.fit(X, y)  
print("Degree-")  
print ("Features sorted by their score:")  
print(rf.feature\_importances\_)  
rf = RandomForestRegressor()  
X = data  
y = closeness  
rf.fit(X, y)  
print("closeness-")  
print ("Features sorted by their score:")  
print(rf.feature\_importances\_)  
rf = RandomForestRegressor()  
X = data  
y = betweenness  
rf.fit(X, y)  
print("betweenness-")  
print ("Features sorted by their score:")  
print(rf.feature\_importances\_)  
  
###逻辑回归  
#degree  
logitmodel = LogisticRegression()  
logitmodel.fit(x\_train1, y\_train1)  
print("逻辑回归-degree")  
print(classification\_report(y\_test1, logitmodel.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(logitmodel,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
logitmodel = LogisticRegression()  
logitmodel.fit(x\_train2, y\_train2)  
print("逻辑回归-closeness")  
print(classification\_report(y\_test2, logitmodel.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(logitmodel,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
logitmodel = LogisticRegression()  
logitmodel.fit(x\_train3, y\_train3)  
print("逻辑回归-betweenness")  
print(classification\_report(y\_test3, logitmodel.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(logitmodel,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
  
###KNN  
#degree  
model = KNeighborsClassifier(n\_neighbors=2)  
model.fit(x\_train1, y\_train1)  
print("KNN-degree")  
print(classification\_report(y\_test1, model.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
model = KNeighborsClassifier(n\_neighbors=2)  
model.fit(x\_train2, y\_train2)  
print("KNN-closeness")  
print(classification\_report(y\_test2, model.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
model = KNeighborsClassifier(n\_neighbors=2)  
model.fit(x\_train3, y\_train3)  
print("KNN-betweenness")  
print(classification\_report(y\_test3, model.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
  
###NB  
#degree  
model = MultinomialNB()  
model.fit(x\_train1, y\_train1)  
print("NB-degree")  
print(classification\_report(y\_test1, model.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
model = MultinomialNB()  
model.fit(x\_train2, y\_train2)  
print("NB-closeness")  
print(classification\_report(y\_test2, model.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
model = MultinomialNB()  
model.fit(x\_train3, y\_train3)  
print("NB-betweenness")  
print(classification\_report(y\_test3, model.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
  
###GBoost  
#degree  
model = GradientBoostingClassifier()  
model.fit(x\_train1, y\_train1)  
print("GB-degree")  
print(classification\_report(y\_test1, model.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
model = GradientBoostingClassifier()  
model.fit(x\_train2, y\_train2)  
print("GB-closeness")  
print(classification\_report(y\_test2, model.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
model = GradientBoostingClassifier()  
model.fit(x\_train3, y\_train3)  
print("GB-betweenness")  
print(classification\_report(y\_test3, model.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
  
###Decision Tree  
#degree  
model = DecisionTreeClassifier()  
model.fit(x\_train1, y\_train1)  
print("DT-degree")  
print(classification\_report(y\_test1, model.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
model = DecisionTreeClassifier()  
model.fit(x\_train2, y\_train2)  
print("DT-closeness")  
print(classification\_report(y\_test2, model.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
model = DecisionTreeClassifier()  
model.fit(x\_train3, y\_train3)  
print("DT-betweenness")  
print(classification\_report(y\_test3, model.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
  
###Random Forest  
#degree  
model = RandomForestClassifier()  
model.fit(x\_train1, y\_train1)  
print("RF-degree")  
print(classification\_report(y\_test1, model.predict(x\_test1)))  
X = data  
y = degree  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#closeness  
model = RandomForestClassifier()  
model.fit(x\_train2, y\_train2)  
print("RF-closeness")  
print(classification\_report(y\_test2, model.predict(x\_test2)))  
X = data  
y = closeness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#betweenness  
model = RandomForestClassifier()  
model.fit(x\_train3, y\_train3)  
print("RF-betweenness")  
print(classification\_report(y\_test3, model.predict(x\_test3)))  
X = data  
y = betweenness  
scores = cross\_val\_score(model,X,y,cv=5, scoring='accuracy')  
print (scores)  
print(scores.mean())  
print(np.var(scores))  
#stacking-degree  
#将基础模型分成两部分，主要供stacking第二层来使用  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[20:99], degree[20:99], test\_size=0.5)  
  
#批量模型构建、批量模型训练、批量模型预测的函数  
def get\_models():  
 """Generate a library of base learners."""  
 knn = KNeighborsClassifier(n\_neighbors=3)  
 GB = GradientBoostingClassifier()  
 DT = DecisionTreeClassifier()  
  
 models = {  
 'knn': knn,  
 'GB': GB,  
 'DT': DT  
 }  
  
 return models  
def train\_base\_learners(base\_learners, inp, out, verbose=True):  
 """Train all base learners in the library."""  
 if verbose: print("Fitting models.")  
 for i, (name, m) in enumerate(base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 m.fit(inp, out)  
 if verbose: print("done")  
def predict\_base\_learners(pred\_base\_learners, inp, verbose=True):  
 """Generate a prediction matrix."""  
 P = np.zeros((inp.shape[0], len(pred\_base\_learners)))  
  
 if verbose: print("Generating base learner predictions.")  
 for i, (name, m) in enumerate(pred\_base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 p = m.predict\_proba(inp)  
 # With two classes, need only predictions for one class  
 P[:, i] = p[:, 1]  
 if verbose: print("done")  
  
 return P  
#定义基础模型  
base\_learners = get\_models()  
#定义权重分配模型  
meta\_learner = LogisticRegression()  
#训练stacking模块  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[0:20])  
n1 = sum((p>0.5)==degree[0:20])/20  
print(n1)  
#五折交叉  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:20].append(data.iloc[40:99]),degree[0:20].append(degree[40:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[20:40])  
n2 = sum((p>0.5)==degree[20:40])/20  
print(n2)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:40].append(data.iloc[60:99]),degree[0:40].append(degree[60:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[40:60])  
n3 = sum((p>0.5)==degree[40:60])/20  
print(n3)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:60].append(data.iloc[80:99]),degree[0:60].append(degree[80:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[60:80])  
n4 = sum((p>0.5)==degree[60:80])/20  
print(n4)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:80],degree[0:80], test\_size=0.75)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[80:99])  
n5 = sum((p>0.5)==degree[80:99])/19  
print(n5)  
total = np.array([n1,n2,n3,n4,n5])  
print(total.mean())  
print(np.var(total))  
#stacking-closeness  
#将基础模型分成两部分，主要供stacking第二层来使用  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[20:99], closeness[20:99], test\_size=0.5)  
  
#批量模型构建、批量模型训练、批量模型预测的函数  
def get\_models():  
 """Generate a library of base learners."""  
 knn = KNeighborsClassifier(n\_neighbors=3)  
 GB = GradientBoostingClassifier()  
 DT = DecisionTreeClassifier()  
  
 models = {  
 'knn': knn,  
 'GB': GB,  
 'DT': DT  
 }  
  
 return models  
def train\_base\_learners(base\_learners, inp, out, verbose=True):  
 """Train all base learners in the library."""  
 if verbose: print("Fitting models.")  
 for i, (name, m) in enumerate(base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 m.fit(inp, out)  
 if verbose: print("done")  
def predict\_base\_learners(pred\_base\_learners, inp, verbose=True):  
 """Generate a prediction matrix."""  
 P = np.zeros((inp.shape[0], len(pred\_base\_learners)))  
  
 if verbose: print("Generating base learner predictions.")  
 for i, (name, m) in enumerate(pred\_base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 p = m.predict\_proba(inp)  
 # With two classes, need only predictions for one class  
 P[:, i] = p[:, 1]  
 if verbose: print("done")  
  
 return P  
#定义基础模型  
base\_learners = get\_models()  
#定义权重分配模型  
meta\_learner = LogisticRegression()  
#训练stacking模块  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[0:20])  
n1 = sum((p>0.5)==closeness[0:20])/20  
print(n1)  
#五折交叉  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:20].append(data.iloc[40:99]),closeness[0:20].append(closeness[40:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[20:40])  
n2 = sum((p>0.5)==closeness[20:40])/20  
print(n2)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:40].append(data.iloc[60:99]),closeness[0:40].append(closeness[60:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[40:60])  
n3 = sum((p>0.5)==closeness[40:60])/20  
print(n3)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:60].append(data.iloc[80:99]),closeness[0:60].append(closeness[80:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[60:80])  
n4 = sum((p>0.6)==closeness[60:80])/20  
print(n4)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:80],closeness[0:80], test\_size=0.75)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[80:99])  
n5 = sum((p>0.5)==closeness[80:99])/19  
print(n5)  
total = np.array([n1,n2,n3,n4,n5])  
print(total.mean())  
print(np.var(total))  
###stacking-betweenness  
#将基础模型分成两部分，主要供stacking第二层来使用  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[20:99], betweenness[20:99], test\_size=0.5)  
  
#批量模型构建、批量模型训练、批量模型预测的函数  
def get\_models():  
 """Generate a library of base learners."""  
 knn = KNeighborsClassifier(n\_neighbors=3)  
 GB = GradientBoostingClassifier()  
 DT = DecisionTreeClassifier()  
  
 models = {  
 'knn': knn,  
 'GB': GB,  
 'DT': DT  
 }  
  
 return models  
def train\_base\_learners(base\_learners, inp, out, verbose=True):  
 """Train all base learners in the library."""  
 if verbose: print("Fitting models.")  
 for i, (name, m) in enumerate(base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 m.fit(inp, out)  
 if verbose: print("done")  
def predict\_base\_learners(pred\_base\_learners, inp, verbose=True):  
 """Generate a prediction matrix."""  
 P = np.zeros((inp.shape[0], len(pred\_base\_learners)))  
  
 if verbose: print("Generating base learner predictions.")  
 for i, (name, m) in enumerate(pred\_base\_learners.items()):  
 if verbose: print("%s..." % name, end=" ", flush=False)  
 p = m.predict\_proba(inp)  
 # With two classes, need only predictions for one class  
 P[:, i] = p[:, 1]  
 if verbose: print("done")  
  
 return P  
#定义基础模型  
base\_learners = get\_models()  
#定义权重分配模型  
meta\_learner = LogisticRegression()  
#训练stacking模块  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[0:20])  
n1 = sum((p>0.5)==betweenness[0:20])/20  
print(n1)  
#五折交叉  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:20].append(data.iloc[40:99]),betweenness[0:20].append(betweenness[40:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[20:40])  
n2 = sum((p>0.6)==betweenness[20:40])/20  
print(n2)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:40].append(data.iloc[60:99]),betweenness[0:40].append(betweenness[60:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[40:60])  
n3 = sum((p>0.5)==betweenness[40:60])/20  
print(n3)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:60].append(data.iloc[80:99]),betweenness[0:60].append(betweenness[80:99]), test\_size=0.5)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[60:80])  
n4 = sum((p>0.5)==betweenness[60:80])/20  
print(n4)  
xtrain\_base, xpred\_base, ytrain\_base, ypred\_base = train\_test\_split(data.iloc[0:80],betweenness[0:80], test\_size=0.75)  
train\_base\_learners(base\_learners, xtrain\_base, ytrain\_base)  
P\_base = predict\_base\_learners(base\_learners, xpred\_base)  
meta\_learner.fit(P\_base,ypred\_base)  
#打包stacking模块  
def ensemble\_predict(base\_learners, meta\_learner, inp, verbose=True):  
 """Generate predictions from the ensemble."""  
 P\_pred = predict\_base\_learners(base\_learners, inp, verbose=verbose)  
 return P\_pred, meta\_learner.predict\_proba(P\_pred)[:, 1]  
P\_pred, p = ensemble\_predict(base\_learners, meta\_learner, data.iloc[80:99])  
n5 = sum((p>0.5)==betweenness[80:99])/19  
print(n5)  
total = np.array([n1,n2,n3,n4,n5])  
print(total.mean())  
print(np.var(total))