中山大學
SUN YAT-SEN UNIVERSITY

以社会网络分析视角探究网络社区意见领袖的识别、定位与预测——以知乎为例

队伍名称：政务数据原地结婚队

队员：陈钦德、陈少捷、冷春江、王威龙、尹佳名、赵先铎、朱梓媛

2019 年 6 月

目录

一、导论

（一）研究背景

互联网时代信息爆炸导致整个互联网空间的信息总量极大，资讯丰富的同时也产生了大量冗余甚至误导性的信息，因此只有经过中间组织过滤后，有价值的信息才能传播给受众。

而在信息的筛选和传播过程中，意见领袖发挥着重要作用。意见领袖是指在团队中构成信息和影响的重要来源、并能左右多数人态度倾向的少数人，擅长于为他人过滤、解释或提供信息。**近年来新兴的经济形态和热点事件的出现，丰富了意见研究的应用研究内容。**

随着网络技术的发展，网络舆情场域的传播环境突破了地域限制，传播要素和传播互动方式变得复杂化和多元化（张伟，2014）。在此背景下，**知乎社区凭借理性认真的公共讨论在舆论场域中扮演了越来越重要的角色，成为可与主流媒体舆论场相抗衡的民间网络舆论场**（陈晨，2016）。

知乎是一个真实的网络问答社区，成立于 2012 年，连接各行各业的精英，用户分享着彼此的专业知识、经验和见解，提倡**精英文化**，聚集了互联网领域和创业**投资领域的精英人士，大量高质量的内容因此沉淀下来。**

在知乎这样的网络环境中，逐渐涌现出一批答案排序靠前、得赞率高、粉丝数多的答主。这些人具有一定的影响力和关注度，被网民称为“知乎大Ｖ”，即传统意义上的“意见领袖”。**在知乎这一知识分享和信息表达的平台上，意见领袖的作用举足轻重，尤其表现在一些热点的公共事件中**，促使知乎成为重要的舆论发源和传播阵地。

（二）研究问题

本研究试图从**社会网络分析方法**的视角，以社会网络的三个中心度为测量基准，探究知乎社区平台中**传统的“意见领袖”范围内，具有真正影响力的意见领袖**的识别方式，构建一套更科学的以知乎为例的**网络社区意见领袖的定位与识别指标**体系。

基于研究结果，本研究将对相关方提出**具有针对性的政策建议**，以促进以知乎为例的网络社区平台中的意见领袖发挥更积极更广泛的社会作用，为政府有关部门提供更精确更科学的社会舆情监测与疏导机制，促进社会舆情传播及媒体行

业的健康繁荣发展。

（三）研究意义

1、理论意义

网络意见领袖研究是传统意见领袖研究在新媒体领域中的探索，本研究对于意见领袖在知乎平台上的拓展，有利于**丰富意见领袖研究的理论体系**。

而在研究方法上，对意见领袖识别这一领域的研究，部分学者采用了聚类分析的方法构建意见领袖的识别指标，而本次研究上主要采用一种新的社会学方法——**社会网络分析法**，将传播学理论、运筹学理论与社会学研究方法相结合，达到多学科之间的相融相通。

2、社会实践意义

通过构建的知乎意见领袖识别模型可以将话题群体中处于意见领袖位置的用户识别出来，**对于政府、媒体、网络社区发展等都具有很大的实用性**。

对于政府而言，有助于政府**及时掌握网络舆情走向，树立公平公正、为人民服务的良好形象**，更加顺利的开展各方面的工作，建构良好的网络舆论秩序。

对于媒体而言，有利于**媒体更快更有效地筛选出有效的信息，并能够达到良好的传播效果**。媒体通过识别出意见领袖，并对意见领袖进行正确的引导可以使信息的传播达到事半功倍的效果。

对于网络社区而言，意见领袖具有多方面的作用：一方面他们是**积极的信息传播者**；另一方面长期活跃使他们成为了社区中的明星人物，信息通过他们的分享和转载可以吸引更多关注、激发更多讨论，在为**社区设置议程、引导舆论**等方面也扮演者十分重要的角色。

二、文献综述与研究方法

（一）文献综述

1、知乎的社会网络分析研究综述

知乎作为网络问答社区，因其强调用户之间及用户与话题建构的复杂社会网络而受到研究者关注。宋文丹（2015）在《社会化问答社区的社会网络分析》一文中用社会网络分析法对知乎社区进行了多层次的网络分析，提出知乎社区通过仿真人脑自身的思维模式实现了人与信息的高效连接，发现知乎社区中的话题和

用户网络相互交织。也有学者选取知乎社区中某个特定的话题为研究对象进行分析，该研究同样以知乎社区中某个单一话题下的参与用户作为抽样研究的样本，选取了其中的 96 名用户构建评论和回复关系矩阵，再借助社会网络分析软件进行数据处理，分析其网络基本属性、中心性和凝聚子群特征，发现知乎社区中的讨论多发生在一个个小团体内部，而且用户之间的关注关系非常稀松（宋学峰，2014）。

总体上国内目前对网络问答社区的社会网络分析研究总体上较少，有待进一步的深入研究。

2、关于知乎意见领袖识别方法的研究综述

对于知乎空间意见领袖识别的研究，不同学者从不同的角度运用不同的方法进行了研究。笔者以“知乎”“意见领袖识别”为关键词在知网进行文献检索，共出现 68 篇文章。其中较多文章注重于研究知乎作为社会化问答社区在网络舆情传播中的舆论功能、影响机制、影响分析等。

目前我国学界关于知乎这一新兴网络社区交流平台的意见领袖识别分析研究还有所欠缺，主**要研究主题局限于宽泛的网络平台意见领袖研究，主体也主要局限于微博这一网络平台，对知乎的意见领袖识别机制的研究十分有限**。

在此背景下，本文以知乎社区为例，通过建立一系列的指标体系，建立和筛选平台意见领袖的模型，并选取不同话题下的活跃用户作为研究样本，分析探究意见领袖的识别机制，以期待在网络舆情事件发生时能够最快最准确地定位意见领袖，控制舆情发展态势。

（二）研究方法

1、文献分析法

本文在构思写作的过程中主要是通过网络广泛的搜集国内外的相关文献，对文献进行分类分析，借鉴前人已有的研究成果和相关的理论，找到该领域的创新点形成自己的研究思路和结论。

2、社会网络分析法

本研究在使用社会网络分析方法时主要是采用了**度中心性分析、中间中心性分析和接近中心性分析**，通过利用这些研究方法确定意见领袖识别模型，并用这些方法对数据进行定量的分析，使得该网络群体内的知乎用户之间的关系能够通

过数据更加直观的展示。

三、数据获取与技术路线

（一）样本获取与研究假设

1、指标确定

根据文献中意见领袖的定义可知,“意见领袖”是指活跃在人际传播网络中，频繁的为他人提供信息、意见等并能够赢得他人的支持，通过个人的影响力对他人的态度、行为产生一定改变的“积极分子”，**影响力是意见领袖基本的能力。**

根据知乎自身的情况，结合相关文献与现实，我们将知乎中意见领袖影响力指标分为以下十二个方面：

粉丝数(fans)、关注数(concern)、回答数(answer)、提问数(question)、文章数(article)、专栏数(column)、优秀回答者数(if super)、认证情况(if identify)、专业认证(profession)、认证回答数(if zhihu)、认证文章数(num article)、赞数，感谢数，收藏数（将三者归一化后的均值表示为 means）。

3、数据抓取

本文在数据样本选取方面，**在知乎平台中选取“政治”、“经济”、“文化”“社会”、“法律”五个话题领域。**之所以选择这五个领域，是综合考虑社会舆情的分布领域情况和数据样本选取的可行性等情况得出的。

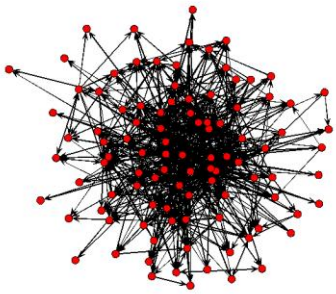
选取领域后，分别在各话题下选定一名关注者人数较多、具有代表性的用户，即传统意义上的意见领袖，以其作为起点，利用 Python 网络爬虫，分别从各起点开始“滚雪球式”向下遍历 10 万名用户，筛选出关注者人数在 1000 人以上的用户作为样本点，分别抓取各样本点的十二个属性值，以构造社会网络。同时，根据各样本点相互的关注情况，以关注为 1，未关注为 0 构造稀疏矩阵。

本文将以上十二个用户特征归纳为三个维度：知乎其他用户认证、知乎平台官方认证和用户自身三种特征。其中，将关注者数，赞数，感谢数，收藏数作为第一个维度，将是否优秀回答者，知乎认证回答数，认证文章数，专业认证数作为第二个维度，将认证情况，提问数，文章数，专栏数，关注数，回答数作为第三个维度。

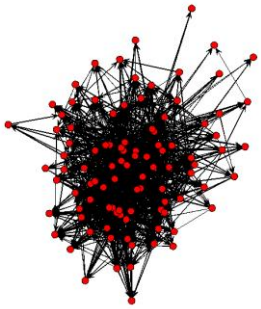
综上，**本研究在结合文献理论和现实情况的基础上，选取了知乎中“政治”、“经济”“文化”“社会”“法律”五大与社会舆情和政府管理息息相关的话题领**

域，在每个话题中抓取关注数超过一千的一百个传统意义上的“意见领袖”，根据知乎用户的十二个用户特征，将其归纳为三个维度，并以此作为自变量其在测量社会网络中心度中的作用，最终构建出一套不同于传统意义的、更具有针对性的知乎意见领袖定位与识别机制和体系，为政府的社会舆情治理和引导提供重要的参考作用。

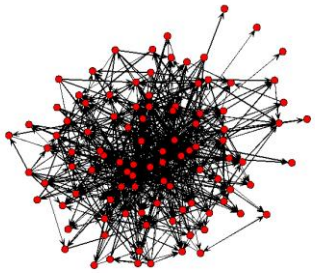
五大样本的网络可视化图形如下：



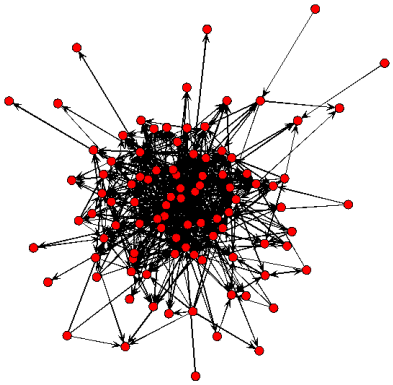
（文化领域网络图）



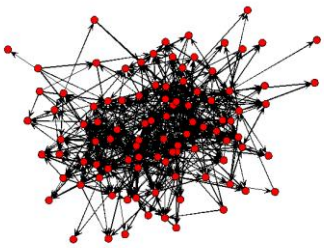
（经济领域网络图）



（法律领域网络图）



（政治领域网络图）



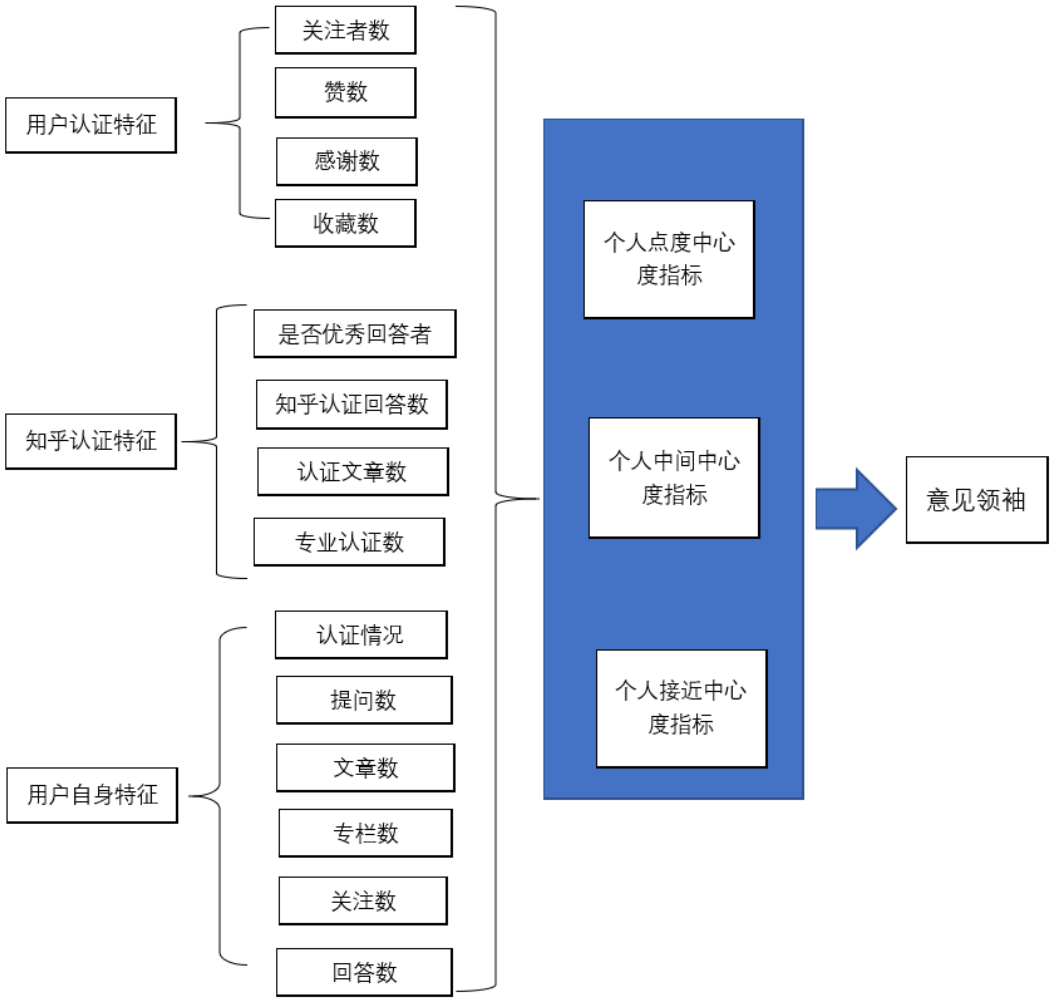
（社会领域网络图）

4、确立假设与研究框架

根据相关文献和数据分析，本研究确定知乎用户的十二个用户特征，即三个维度的用户特征为自变量，以社会网络的三个中心度为因变量，研究假设为：

十二个用户特征对于识别“政治”、“经济”、“文化”、“社会”、“法律”话题下传统意见领袖中真正的意见领袖具有显著作用。

本研究整体分析框架如下图所示：



（二）数据处理与模型

1、数据预处理

样本的特征值分别为：姓名，id，和以上十二个用户特征，而样本在知乎社交网络中的属性值有三种：degree, betweenness, closeness。如图所示：

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	知乎认证	认证文章	赞数	感谢数	收藏数	专业认证	degree	betweenn	closenes
https://wv	madaye	35910	295	78	1	15	1	未认证	是	26	8	37199	7293	21569	1	13	148.2582	0.002045
https://wv	kandy_kai	31836	0	67	12	44	1	未认证	否	0	0	24033	5960	18229	0	2	0	0.000101
https://wv	letsdream	4652	228	48	0	5	0	未认证	否	0	0	1265	236	8644	0	2	0	0.001443
https://wv	fatfox10	84293	1510	1543	10	37	0	未认证	否	0	0	679128	88366	110994	0	37	519.0286	0.002123
https://wv	ontheroac	335582	50	38	1	7	1	未认证	否	16	0	284139	37210	67018	2	15	5.056526	0.001653

（图为文化领域的 5 份样本）

我们将其知乎用户的两种类型：1、是否专业认证，2、是否为优秀回答者。其中类型 1 分为无专业认证和相关的专业认证，因此我们将有相关的专业认证的

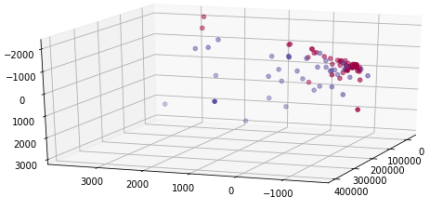
用户的该数值设定为 1，无相关专业认证设定为 0；类型 2 分为优秀回答者和非优秀回答者，将是优秀回答者为 1 和非优秀回答者设定为 0 。

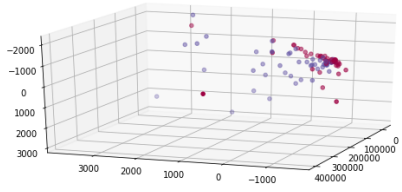
为了降低特征之间的共线性，我们对特征值之间进行了相关性运算。

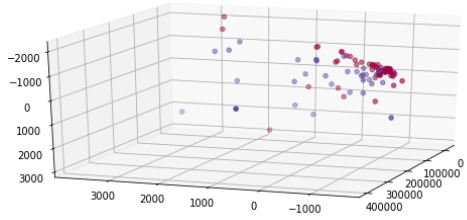
同时在所有线性相关性的图中，我们注意到**点赞，收藏，感谢三个特征值互相之间的线性相关性均极高**。因此为了简化运算，我们将三种特征值均进行了归一化，并取均值获得新的特征值。

之后，我们准备采取二分类的方法建模进行初步处理，所以我们对各个样本的网络权重数据进行了归一化和降维，并为了得到较好的数据分割点，我们采取了相应的三维建模，并最终获得了较好的数据图，并且不同的网络有着不同的阈值。

政治网络：degree 阈值为 0.22，closeness 对应阈值为 0.72，betweenness 对应阈值为 0.038。其中三维建模分类结果较好的图像有：

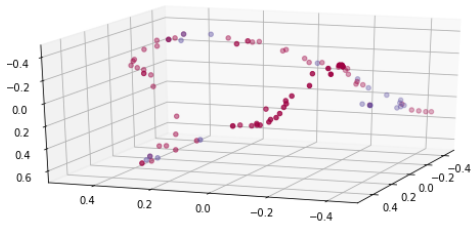


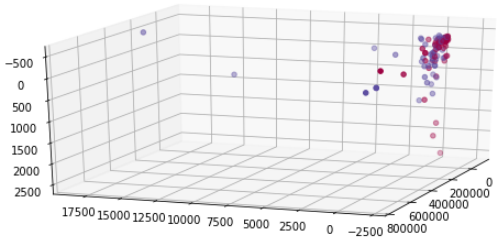


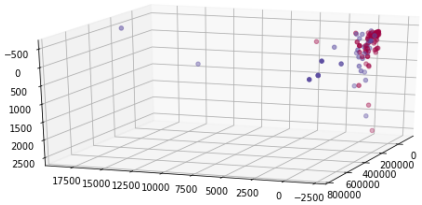


可以看到在此阈值下，分类效果较好。

文化网络：degree 阈值为 0.25，closeness 对应阈值为 0.72，betweenness 对应阈值为 0.04。其中三维建模分类结果较好的图像有：

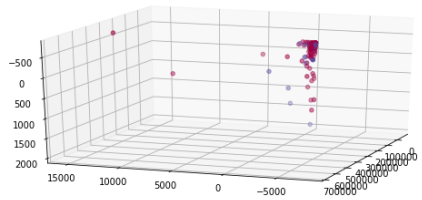


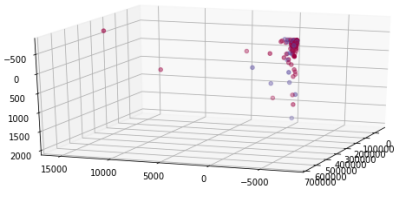


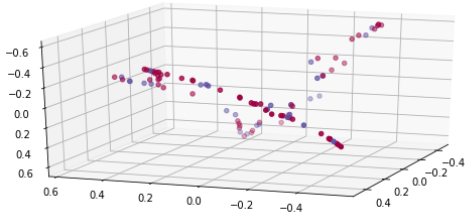


可以看到在此阈值下，分类效果较好。

法律网络：degree 阈值为 0.22，closeness 对应阈值为 0.85，betweenness 对应阈值为 0.038。其中三维建模分类结果较好的图像有：

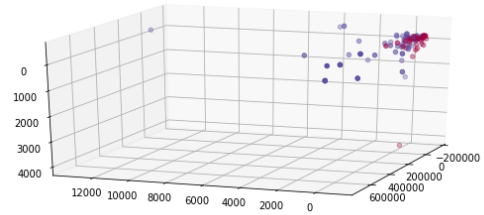
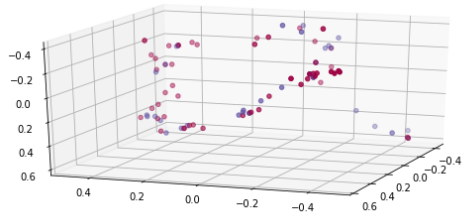
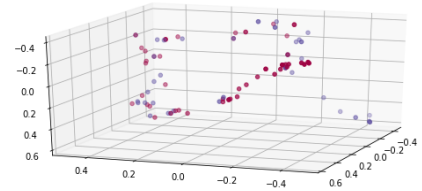






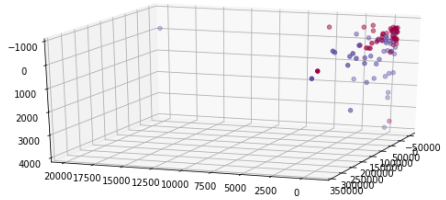
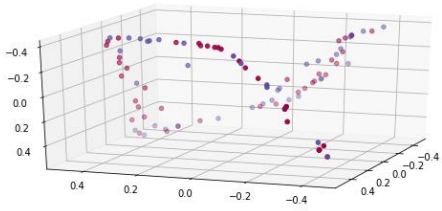
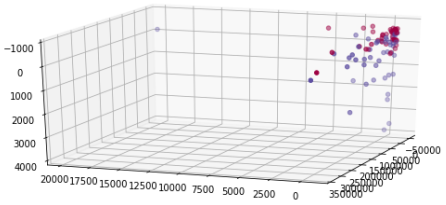
可以看到在此阈值下，分类效果较好。

经济网络：degree 阈值为 0.26，closeness 对应阈值为 0.57，betweenness 对应阈值为 0.038。其中三维建模分类结果较好的图像有：

可以看到在此阈值下，分类效果较好。

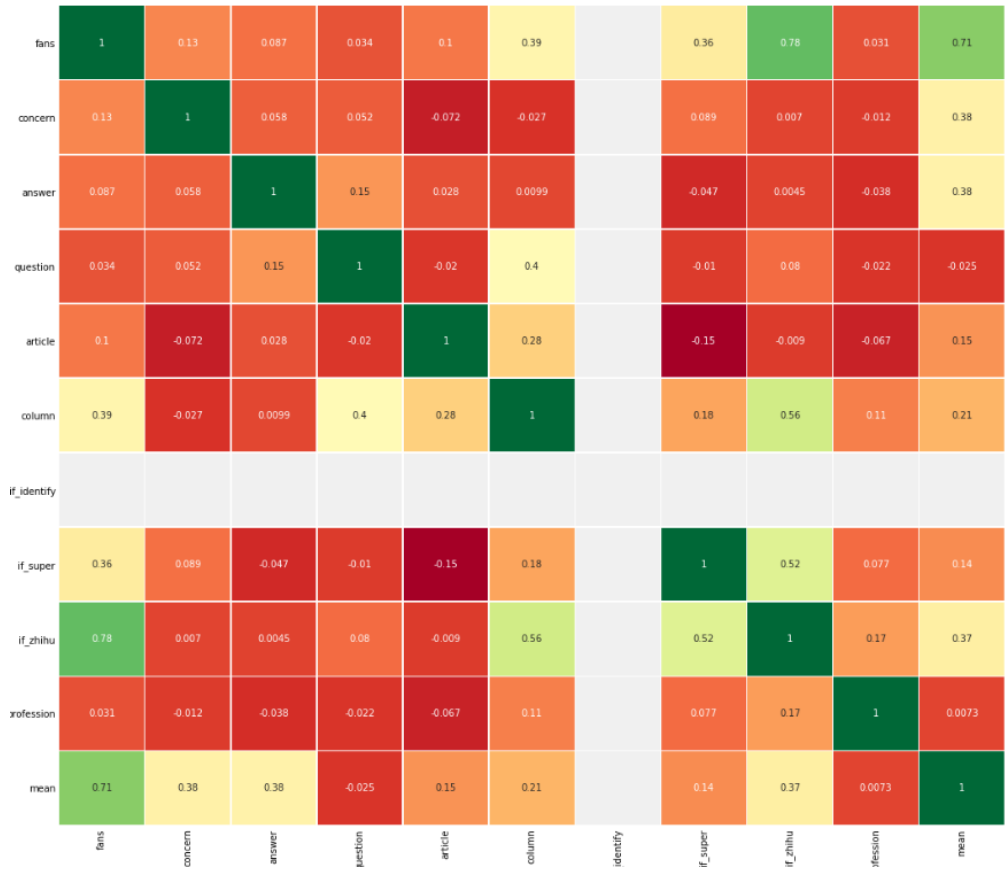
社会网络：degree 阈值为 0.24，closeness 对应阈值为 0.75，betweenness 对应阈值为 0.046。其中三维建模分类结果较好的图像有：

处理之后，以文化网络为例，删掉了认证的文章数，同时将三证网络属性值改为 1，0。如下图所示：

fans	concern	answer	question	article	column	if_identify	if_super	if_zhihu	profession	mean	degree	betweenn	closeness
35910	295	78	1	15	1	0	1	26	1	0.030793	0	1	1
31836	0	67	12	44	1	0	0	0	0	0.023383	0	0	0
4652	228	48	0	5	0	0	0	0	0	0.002687	0	0	0
84293	1510	1543	10	37	0	0	0	0	0	0.399992	1	1	1
335582	50	38	1	7	1	0	0	16	2	0.173866	0	0	0

同时更新后的线性相关图为



可以看到样本的特征值之间不在存在线性相关较高的两种特征，因而**预处理后的样本数据具有更高的可操作性和科学性。**

2、训练模型

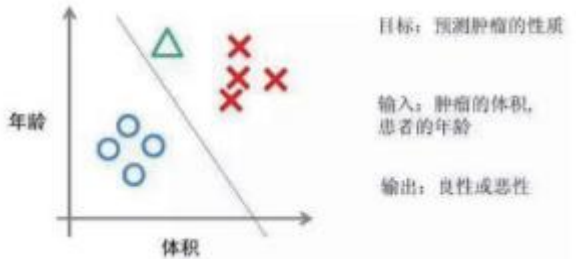
本文分别采用逻辑回归，决策树，KNN，朴素贝叶斯，梯度提升树，随机森林（Bagging），Stacking 集成算法，以预处理后的特征为自变量，以三项社会网络指标为因变量，建立预测模型，并依据最终预测准确率和预测精度，我们选择了 Stacking 集成模块算法作为识别，定位意见领袖的核心算法。

3、算法与模型介绍

（1）Stacking 集成算法介绍

Stacking 集成算法是将多种分类器继承于一体的集成算法，指训练一个模型用于组合其他各个模型的方法。Stacking 模型分为 2 层，第 0 层首先训练多个不同的模型（基学习器），然后再以之前训练的各个模型的输出输入第 1 层来训练一个模型（元学习器）。在 0 层首先对训练集 S，使用类似 k 折交叉验证法的方式将数据分成 J 个部分 $S_1,\ldots,S_j$。对于第 j 次训练，保留 S_j 的数据，然后将 $S^{(-1)} = S - S_j$ 用来训练每一个基础分类器 K_j，训练完成后，使用每个基础分

类器对数据 S_j 进行预测分类，产生该数据的所属各类别的后验概率，若训练数据存在 N 个类别，那么每个基础分类器将会产生 N 个由后验概率组成的新的特征维度 p_{kj}，K 个分类器将组成 K×N 个新维度，这些新增的特征维度将作为第 1 层中的训练数据。根据 k 折交叉验证的原理，算法结束时将会计算 J 次，直至训练集中的所有数据都被转换成由后验概率构成的新数据，第 0 层阶段结束。由于使用了交叉验证技术，故训练集并不会存在数据泄露问题。



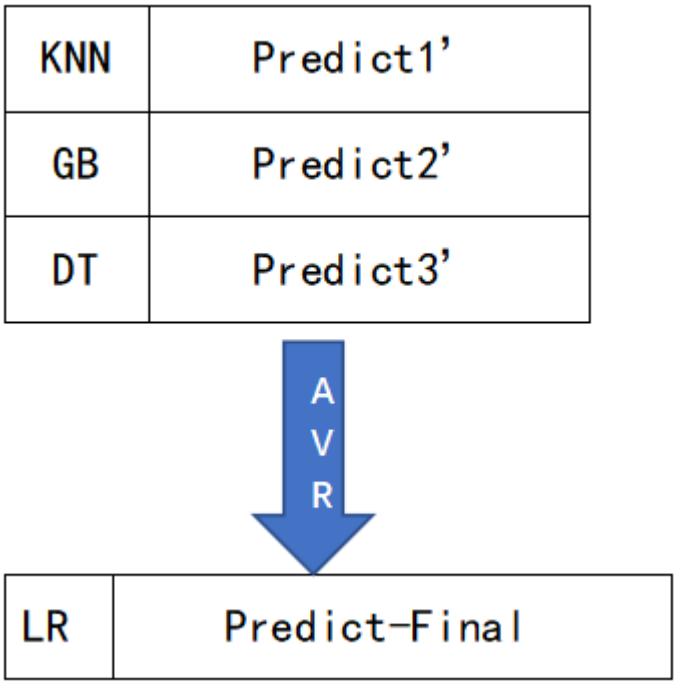
（2）研究模型构建

本研究选择将 KNN，梯度提升树和决策树作为 stacking 第一层模块，选择逻辑回归分类器作为第二层模块。

KNN	Trains:Data1,2	Predict:Data3’
GB	Trains:Data2,3	Predict:Data1’
DT	Trains:Data1,3	Predict:Data2’

LR	Train:Data1’,2’,3’

将训练集三分之后，每个第一层模块的分类器取不完全相同的两份进行训练，并预测剩余一份数据的结果，获得的三份预测结果作为第二层模块分类器的训练数据进行训练。



在预测的过程中，我们让第一层三个分类器分别预测三个结果，然后将预测结果取平均值，放入第二层的逻辑回归分类器中进行预测，得到的最终预测结果。（由于平均后的数值可能介于 01 之间，因此我们将最后的结果在进行一次划分，大于 0.5 的赋值为 1，小于 0.5 的赋值为 0。）

（三）预测结果

economy-degree				
Five-Fold				
0.8	0.8	0.5	0.75	0.68
平均准确率		0.71		
精度误差		0.0125		
economy-closeness				
Five-Fold				
0.8	0.7	0.7	0.7	0.79
平均准确率		0.738		
精度误差		0.003165		
economy-betweenness				
Five-Fold				
0.85	0.7	0.65	0.75	0.737
平均准确率		0.737		
精度误差		0.04375		
law-degree				

Five-Fold				
0.75	0.65	0.8	0.65	0.79
平均准确率		0.73		
精度误差		0.0043		
law-closeness				
Five-Fold				
0.8	0.85	0.8	0.8	0.58
平均准确率		0.765		
精度误差		0.009		
law-betweenness				
Five-Fold				
0.7	0.75	0.85	0.5	0.63
平均准确率		0.686		
精度误差		0.0137		
police-degree				
Five-Fold				
0.65	0.35	0.75	0.65	0.7
平均准确率		0.62		
精度误差		0.018		
police-closeness				
Five-Fold				
0.9	0.85	0.95	0.8	0.59
平均准确率		0.82		
精度误差		0.0194		
police-betweenness				
Five-Fold				
0.7	0.4	0.7	0.75	0.59
平均准确率		0.62		
精度误差		0.0184		
culture-degree				
Five-Fold				
0.9	0.9	0.75	0.75	0.8
平均准确率		0.82		
精度误差		0.0046		
culture-closeness				
Five-Fold				
0.85	0.7	0.9	0.75	0.8
平均准确率		0.8		
精度误差		0.005		
culture-betweenness				
Five-Fold				
0.65	0.65	0.85	0.55	0.6
平均准确率		0.66		

精度误差		0.0104		
social-degree				
Five-Fold				
0.9	0.8	0.75	0.65	0.737
平均准确率		0.767		
精度误差		0.00673		
social-closeness				
Five-Fold				
0.75	0.9	0.6	0.8	0.42
平均准确率		0.694		
精度误差		0.028		
social-betweenness				
Five-Fold				
0.8	0.65	0.55	0.5	0.42
平均准确率		0.58		
精度误差		0.017155		

五、研究分析与发现

（一）五大网络特征属性分析

1、法律

（1）整体网络特征

图属性					
	文化	经济	社会	政治	法律
密度	0.0869697	0.1883117	0.0768988	0.07570876	0.08168317
互惠性	0.8806061	0.7895279	0.9055333	0.8996993	0.890495
传递性	0.3512489	0.4971265	0.2680498	0.35354	0.2864464
连通性	1	1	1	1	1
效率	0.9222528	0.8199708	0.9326177	0.9339193	0.9275
平均路径长度	1.8644	1.716151	2.151395	2.163035	1.938045
最长路径长度	3	4	4	5	3

①互惠性较强。

在选取的五个领域中，“法律”话题下的网络互惠性较高为 0.890495，这说明知乎上“法律”话题下的意见领袖中两两互相关注的比例较大。这可能是由于**法律话题相比于其他话题来说具有较强的学科专业性和独特性**，在这一领域下产生较多学界普遍认可和推崇的“大牛”们，他们之间一般也相互熟知，所以在互惠性上表现较为明显。

②传递性较差。

“法律”话题的传递性较低为 0.2864464，这说明在“法律”领域中，任意三人或多人间信息传递的速率较慢。其中的原因可能在于法律领域相比于其他领域来说，**学科的专业性和创新性要求较高，不同学者和学派之间的观点存在较大的差异，**有时甚至完全对立相反，相比于“经济”等具有较强“主流”思想的领域来说，“法律”领域下的“小团体”现象较少，故而表现为网络的传递性较差。

（2）用户特征权重

	degree	closeness	betweenness
concern	0.1687	0.165	**0.083**
answer	0.18	0.1224	**0.172**
question	0.132	0.066	0.18
article	0.1555	0.118	0.14
column	0.015	0.038	0.024
if_identify	0	0	0
if_super	0	0.0049	0.013
if_zhihu	0.058	0.055	0.0679
profession	0.068	0.024	0.05
means	**0.131**	**0.305**	**0.168**
fans	**0.117**	**0.102**	**0.1**

根据数据结果，三个维度的不同特征对于测量“法律”领域下的社会网络中心度具有不同的作用和影响。

在三个维度的特征中，**用户自身特征对于中间中心性的影响较弱。**而用户自身特征相比于其他两个维度的特征对此的影响较小的原因可能在于，**法律话题领域下回答相关的问题专业门槛较高，难度较大，知乎认证和用户认证即可在较大程度上说明该节点是否处于话题网络中的主导地位，**即对整体网络资源的控制程度，而用户的自身特征如举办专栏数、回答问题数等对于测量此变量的相关性不高。

在测量点度中心度中，用户的自身特征如回答数、关注数、文章数等对于测量的权重较大，说明“法律”领域下的用户越活跃、参与度越高，获得的关注和与其他用户的互动随之增加，其在网络中与其他节点的联系就越紧密。

而在测量接近中心度中，其他用户认证的特征如感谢数、点赞数、收藏数等权重较为明显，这说明在“法律”领域中，由于该话题的专业门槛高，需要较深厚的专业法律素养才能够用通俗的话语将深刻的道理呈现出来，获得其他用户的

肯定和认可，所以其他用户对某一用户的认可程度越高，该用户在网络中处于越核心的地位。而用户自身的特征作用也较为明显。

（3）可视化——网络内的用户特征

Law		
	1	0
if_identify	0	**99**
if_super	31	68

值得格外注意的是，由上图可知知乎上“法律”领域下具有“知乎认证”身份的用户比率极低，这可能是由于“法律”话题的政治敏感性较高，“知乎认证”身份较为显眼，或由以上意见领袖活跃度较低可推测，法律界的“大牛”不愿花时间和精力去申请知乎的认证，故而显现为**法律网络下传统意见领袖中的“知乎认证”用户较少。**

这与以上的定位与识别体系相矛盾，所以本文对此提出建议，政府可以与知乎平台联手合作，推出一系列激励机制提高知乎内意见领袖的知乎认证比率，以便政府更快、更准地识别真正的意见领袖。

综上，**若政府想要在“法律”网络下识别出真正的意见领袖，可以把目光着眼于“知乎认证”和“用户认证”这两类特征，其中较为重要的指标是感谢数、关注数、点赞数、是否为知乎认证优秀回答等，而不用过多关注用户自身的活跃度等特征。**

2、政治

（1）整体网络特征

图属性					
	文化	经济	社会	**政治**	法律
密度	0.0869697	0.1883117	0.0768988	**0.07570876**	0.08168317
互惠性	0.8806061	0.7895279	0.9055333	**0.8996993**	0.890495
传递性	0.3512489	0.4971265	0.2680498	**0.35354**	0.2864464
连通性	1	1	1	**1**	1
效率	0.9222528	0.8199708	0.9326177	**0.9339193**	0.9275
平均路径长度	1.8644	1.716151	2.151395	**2.163035**	1.938045

最长路径长度	3	4	4	5	3

①密度小，平均路径长度长。

“政治”话题下的网络密度为 0.07570876，在五个话题中密度最小，传递信息的平均路径长度为 2.163035，在五个话题中信息传递所需经过的用户个数最多。**这说明“政治”话题内粉丝数较多的传统意见领袖之间的分布稀疏，网络数据流通与信息传递性不好，原因可能在于“政治”是较为敏感、封闭、专业性高的话题，舆论导向主要由权威性高的官方主体、专业人士来引导**，不是普通知乎用户可以随意议论的，稍有不慎可能触犯国家与法律权威。而且非内部人士，很难获取第一手政治信息，政治信息机密性极高。

②互惠性较高。

在选取的五个话题中，“政治”话题的互惠性位居第二，为 0.8996993，即“政治”话题内的传统意见领袖之间互相关注度高。我们可以推测，“政治”话题相比于其他话题，权威性、机密性都较强，**使得受到用户们普遍认可的政治界专业人士数量相对较少，较小的圈子意见领袖们彼此认识的机会也会大大增加**。

（2）用户特征权重

	degree	closenness	betweenness
concern	**0.3748**	**0.5442**	**0.376**
means	**0.2277**	**0.124**	**0.1665**
fans	0.1234	0.0341	0.114
column	0.0779	0.00525	0.0089
article	0.0722	0.05442	0.125
answer	0.0687	0.0959	0.15
if_zhihu	0.0299	0.0115	0.017
question	0.0253	0.0904	0.0294
if_identify	0	0	0
num_artic	0	0.0104	0.01
profession	0	0.0297	0.0038

根据以上数据结果可见，三个维度的不同特征对于测量“政治”话题内的社会网络中心度具有不同的作用和影响。

传统意见领袖在度中心性（degree）、接近中心性（closeness）、中介中心性（betweenness）三个中心度中占有权重最大的都是“关注数”，这一特征值得我们去重点关注。我们可以预测，一个关注数很高的传统意见领袖，在知乎“政治”领域这个团体中很有可能是中心人物，这位意见领袖会跟其好友用户联系紧密，并对他们的观点起引导作用。

我们可以发现“赞数”、“感谢数”、“收藏数”这三个归一后的特征值在三个中心度中占比权重位居第二，同时“粉丝数”占有较大权重，**这说明粉丝对传统意见领袖的喜爱在“政治”话题中十分重要。**

这与我们选取粉丝较多的用户为传统意见领袖相符合，说明在“政治”话题中粉丝较多的用户很可能就是真正的意见领袖。

（3）可视化——网络内的用户特征

	Politic	
	1	0
if_identify	0	97
if_super	2	95

如上图所示，“政治”话题内的传统意见领袖基本上都被知乎认证和成为优秀回答者，这也进一步印证了我们之前所推测的“政治”话题专业性、权威性较高。

综上，政府若要在“政治”话题领域识别真正的意见领袖，**应更注重于该用户是否具有较高的“关注数”、“粉丝数”等，在受其他用户关注度较高且拥有良好好友圈子的传统意见领袖群体中寻找，**以便更精确和快速地识别真正的意见领袖。

3、社会

（1）整体网络特征

图属性					
	文化	经济	社会	政治	法律
密度	0.0869697	0.1883117	0.0768988	0.07570876	0.08168317
互惠性	0.8806061	0.7895279	0.9055333	0.8996993	0.890495

传递性	0.3512489	0.4971265	**0.2680498**	0.35354	0.2864464
连通性	1	1	**1**	1	1
效率	0.9222528	0.8199708	**0.9326177**	0.9339193	0.9275
平均路径长度	1.8644	1.716151	**2.151395**	2.163035	1.938045
最长路径长度	3	4	**4**	5	3

①密度较小，传递性较差。

“经济”话题下的网络密度偏低为 0.0768988，传递性偏低为 0.2680498，在五个话题网络中传递效率最低，传递的平均路径长度为 2.151395，信息传递所需经过的用户较多为 4 人。这说明**“社会”话题下粉丝数较多的传统意见领袖之间的分布较为稀疏，联系较为松散；网络中信息传递效率较低，要获得核心信息所需要花费的时间精力较多。**

这可能是由于“社会”话题领域较为宽泛，专业性不如“法律”“政治”等领域强，专业门槛不高，**意见领袖呈现“草根化”的特征**，他们之间也并没有很多的渠道和平台可以互相认识和互动，所以呈现为网络的密度较小；再者，由于社会话题领域中的问题相比其他领域更为“平民化”和“日常化”，故而社会话题下的意见领袖间联系较为松散，用户对热点信息的关注度不高，造成信息传递效率较差，获取重要信息的经过路径长度较长。

②互惠性高。

“经济”话题下的互惠性最高为 0.9055333，即“社会”话题内的传统意见领袖之间互相关注率最高。原因可能是由于上文所讲，“社会”话题下的传统意见领袖的“草根”身份特征较为突出，因此有机会和渠道相互认识的意见领袖较少，一旦有，他们之间就**极易形成“小团体”**，互相“抱团”分享信息资源；且“社会”领域中的用户关注点较为分散，能够获得较多关注的意见领袖需要有较丰富的信息资源的获取、整合、表达等能力，这对于单个非体制内的不知名用户来说是很困难的，故而“社会”领域中的“小团体”较多，呈现为网络的互惠性较强。

（2）用户特征权重

	degree	**closeness**	**betweenness**
concern	**0.478693**	**0.59694**	**0.4288**
answer	**0.11336**	**0.09693**	**0.08127**
question	0.0277	0.033	0.079
article	0.1039	0.099	0.09587

column	0.0351	0.0077	0.07297
if_identify	0	0	0
if_zhihu	0	0	0
num_artic	0.025125	0.0171	0.0321
profession	0.0311	0	0.01415
means	0.1291	0.092389	0.00808
fans	0.0559	0.0569	0.1148

根据以上数据结果可见，三个维度的不同特征对于测量“社会”话题内的社会网络中心度具有不同的作用和影响。

可以发现，**传统意见领袖的关注数在三个中心度中的权重都是较高的，且相比于其他网络来说特征更为明显**，有的甚至达到 0.6 的权重，可见传统意见领袖的“关注数”这一特征十分重要。

而“**回答数**”在度中心性（degree）和中介中心性（betweenness）中都占比权重第二高，在亲近中心性（closenness）中的权重排第三，可见此特征也较为重要。可以推测，在“社会”话题领域中，较为活跃和积极的回答问题者能获得其他用户较为持久和集中的关注，在整个网络中起到较强的影响力。

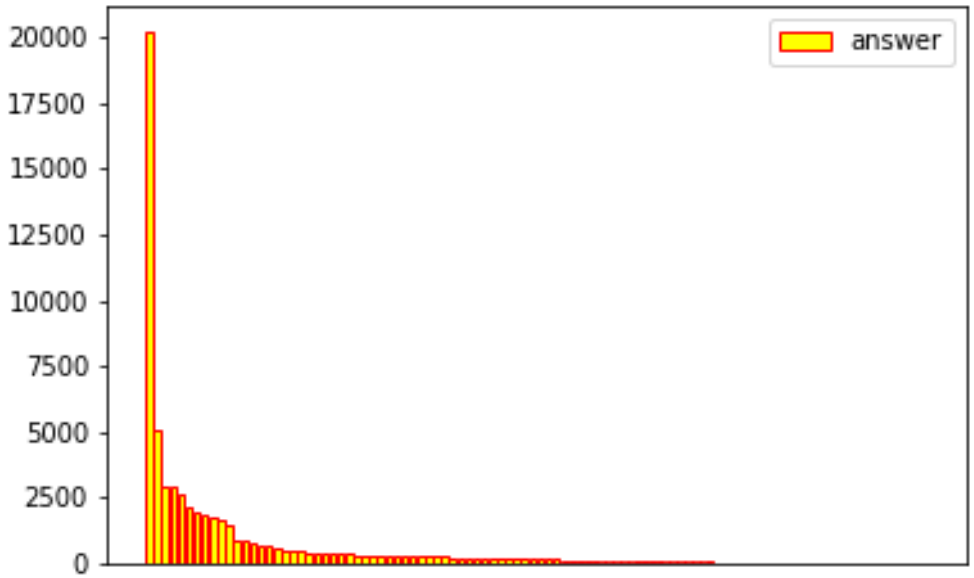
值得注意的是，在“社会”网络中，“知乎认证”和“知乎认证的回答数”等知乎认证特征在测量三个维度的中心度时作用为零，这说明**在“社会”话题领域下，知乎的认证作用对于定位和识别真正的意见领袖作用微弱**。

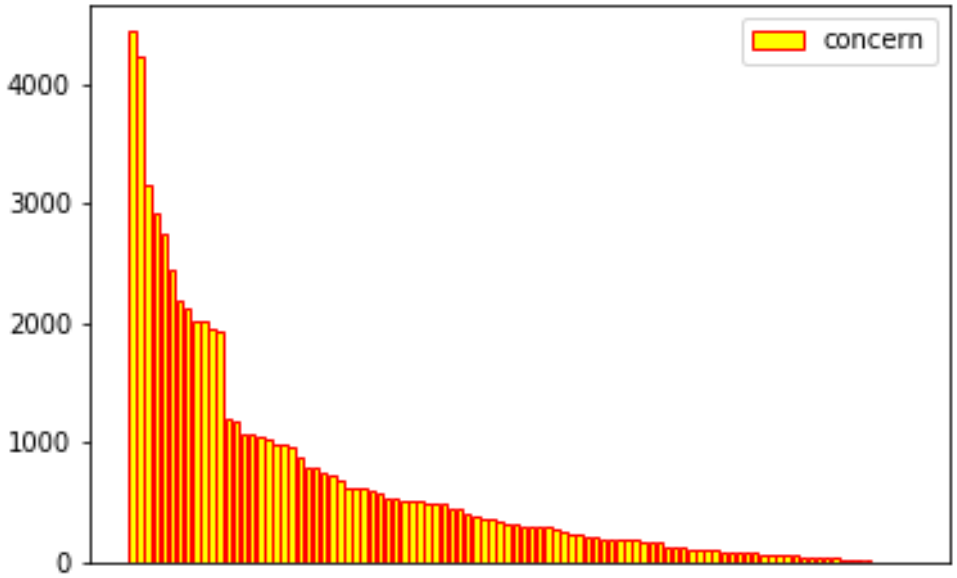
这是由于社会话题领域专业门槛较低，发言内容质量参差不齐，且并无专业上的明确标准，在某种程度上，**“专业性”和“影响力”之间并不呈同比增长**，所以在此网络中，其他用户的认可和该意见领袖自身的活跃度和参与度较为重要，而知乎认证的特征则不能起到很好的定位和识别作用。

（3）可视化——网络内的用户特征

Society		
	1	0
if_identify	0	98
if_super	9	89

如上图所示，“社会”话题领域下，**具有知乎认证特征的传统意见领袖用户不多，即传统意义上使用“专业性”来衡量意见领袖的方法在此领域中失效。**





而根据以上“回答数”和“关注数”的可视化图可见，两个特征均呈现快速下降的右尾型分布，而由以上数据结果可知，这部分“关注数”和“回答数”都较高的用户较少，且并不与“知乎认证”的“专业”用户重叠，即**前者才是“社会”网络中真正意义上的具有影响力的意见领袖。**

综上，政府若要在“社会”话题领域识别真正的意见领袖，可以**排除知乎认**

证的“专业”用户，而更注重着眼于该用户自身的特征，如“关注数”、“回答数”等，在活跃度和参与度较高的意见领袖群体中寻找，以便更精确和快速地识别真正的意见领袖。

4、文化

（1）整体网络特征

图属性					
	文化	经济	社会	政治	法律
密度	0.0869697	0.1883117	0.0768988	0.07570876	0.08168317
互惠性	0.8806061	0.7895279	0.9055333	0.8996993	0.890495
传递性	0.3512489	0.4971265	0.2680498	0.35354	0.2864464
连通性	1	1	1	1	1
效率	0.9222528	0.8199708	0.9326177	0.9339193	0.9275
平均路径长度	1.8644	1.716151	2.151395	2.163035	1.938045
最长路径长度	3	4	4	5	3

①密度较高，传递性较好。

“文化”话题下的网络密度为 0.0869697，传递性为 0.3512489，在五个话题中密度和传递性均处于中上等的水平，说明话题网络下的意见领袖相互关注的比例较高、信息传递所需经过的用户也较少。由于文化这一话题包含的内容非常广泛，且相比于经济、法律话题，文化话题下意见领袖数量较少，所以不同文化领域意见领袖相互关注的比例较高，信息传递经过的用户数也较少。

②互惠性、传递性、效率等特征均处于五个话题中的中间位置（2-4 位），特征值是五个话题中的中位值或与中位值大小相差不大，说明在互惠性、传递性、效率特征上，文化这一话题处于中间水平，即意见领袖两两关注的比例适中、信息传递的效率适中。相比于其他几个话题，文化这一话题没有“法律”“政治”那么强的专业性，也没有“经济”这一话题与人民联系密切、受关注程度高，所以特征值较多处于中位水平。

（2）用户特征权重

	degree	closeness	betweenness
concern	0.3053	0.6494	0.2785
answer	0.0492	0.097	0.061
question	0.019	0.0294	0.1069
article	0.0082	0.0939	0.1226
column	0.01576	0.0078	0.059

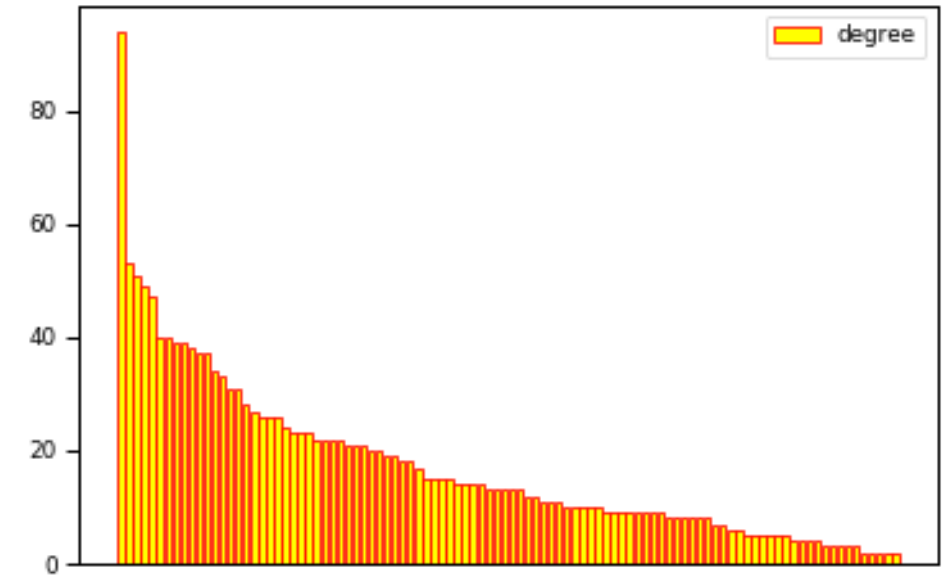
if_identify	0	0	0
if_super	0.01131	0.0066	0.00188
if_zhihu	0.0653	0.00894	0.00895
profession	0.048	0.0039	0.08543
means	0.1234	0.0065	0.1414
fans	**0.3542**	0.00377	0.0539

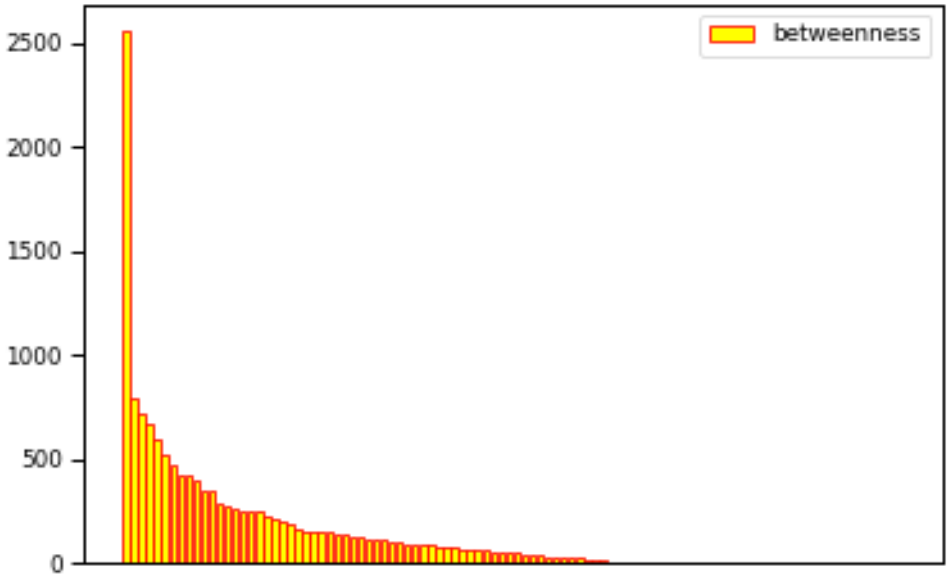
根据以上数据结果可见，三个维度的不同特征对于测量“文化”话题内的社会网络中心度具有不同的作用和影响。

在文化领域，度中心性最高的是 fans，接近中心性、中介中心性最高的是 concern，即意见领袖的“关注数”，因此我们可以推论，在识别知乎文化话题意见领袖获取信息的敏感性、是否有更好的视野及时了解信息的产生和流向时，以及意见领袖获取信息资源的能力、控制其他用户之间的信息交流的能力时，**“关注数”具有较强的说服力。**此外，**传统意见领袖的关注数在三个中心度中的权重都是较高的，**可见传统意见领袖的“关注数”这一特征十分重要。

在文化网络中，“知乎认证”特征在测量三个维度的中心度时作用为零，这说明**在文化话题领域下，知乎的认证作用对于定位和识别真正的意见领袖作用微弱**。这是由于文化话题领域内容广泛，专业门槛相对较低，所以在此网络中，其他用户的认可和该意见领袖自身的活跃度和参与度较为重要，而知乎认证的特征则不能起到很好的定位和识别作用。

（3）可视化——网络内的用户特征





我们可以看到，在文化这一话题中，度中心性和中介中心性的个体之间差距大，我们可以推测，在这些意见领袖中存在着**“意见领袖的意见领袖”**，其对于信息的获取和传播、对于其他意见领袖的信息交流起着重要作用。

综上，文化领域的意见领袖识别，**“知乎认证”指标的作用微弱，需要重点关注用户自身的“关注数”、“粉丝数”等特征。**

5、经济

（1）整体网络特征

图属性					
	文化	经济	社会	政治	法律
密度	0.0869697	0.1883117	0.0768988	0.07570876	0.08168317
互惠性	0.8806061	0.7895279	0.9055333	0.8996993	0.890495
传递性	0.3512489	0.4971265	0.2680498	0.35354	0.2864464
连通性	1	1	1	1	1
效率	0.9222528	0.8199708	0.9326177	0.9339193	0.9275
平均路径长度	1.8644	1.716151	2.151395	2.163035	1.938045
最长路径长度	3	4	4	5	3

①密度大，传递性好。

“经济”话题下的网络密度为 0.1883117，在五个话题中密度最大，传递性为 0.4971265，在五个话题中传递效率最高。**这说明“经济”话题下普遍网络数据流通性好，信息传递效率高，传统意见领袖分布较为集中。**经济一直是人们的热门讨论话题，国外贸易战略、国内经济政策、个人工资涨幅都跟人们息息相关。而且**“经济”话题大众认知度与接受度较高，**不像“政治”、“法律”话题，需要相关领域内的专业人士进行发言与回答，外行人需谨慎发言。

②互惠性高。

“经济”话题下的互惠性为 0.7895279，在五个话题中互惠性最低，即“经济”话题内的传统意见领袖之间互相关注率低。**原因可能是用户们乐于集体讨论当下的经济热点，易形成大团体，**不易形成小团体，彼此之间可能互不相识。“经济”话题内的用户很可能主要关注的是话题，而不是关注某一个用户。

（2）用户特征权重

	degree	closenness	betweenness
concern	0.297856	0.451	0.285
if_zhihu	0.1163	0.00465	0.101
question	0.0989	0.0068	0.037
column	0.084	0.0457	0.0191
mean	0.0664	0.13	0.0423

article	0.0545	0.07	0.029
num_artic	0.0136	0.0089	0.0413
profession	0.00496	0.00961	0.065
answer	0.003827	0.134	0.167
if_identify	0	0	0
if_super	0	0	0.022

据以上数据结果可见，三个维度的不同特征对于测量“经济”话题内的社会网络中心度具有不同的作用和影响。

将度中心性（degree）以升序排列，接近中心性（closeness）、中介中心性（betweenness）随其排列，可以发现，**传统意见领袖的关注数在三个中心度中都是最高的，说明关注数这一特征值十分重要。**我们可以预测，如果一位传统意见领袖的关注数较多，那么很可能这位传统意见领袖的同行业社交圈子较广，相邻好友数多，与好友联系紧密，其可能位于经济领域网络图中偏中心的位置，能够在信息传输方面起到关键节点的作用，造成的影响力较大。

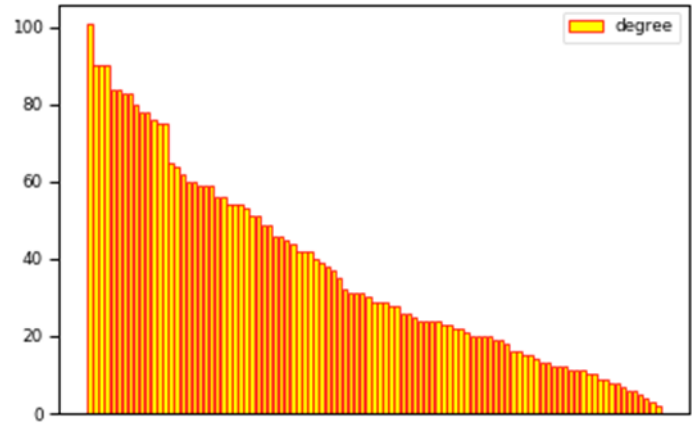
度中心性（degree）中占比权重第二高，中介中心性（betweenness）占比权重第三高的都是**知乎认证的回答数**。可以推测，尽管“经济”话题内大众乐于集体讨论，但**如果某一传统意见领袖发言极为专业与优秀，受众度高，那么其将获得比一般传统意见领袖更高的知乎认证的回答数，成为真正意义的意见领袖**。通过这位意见领袖的“中介”作用，很多用户都会互相认识，进而促进信息交流。

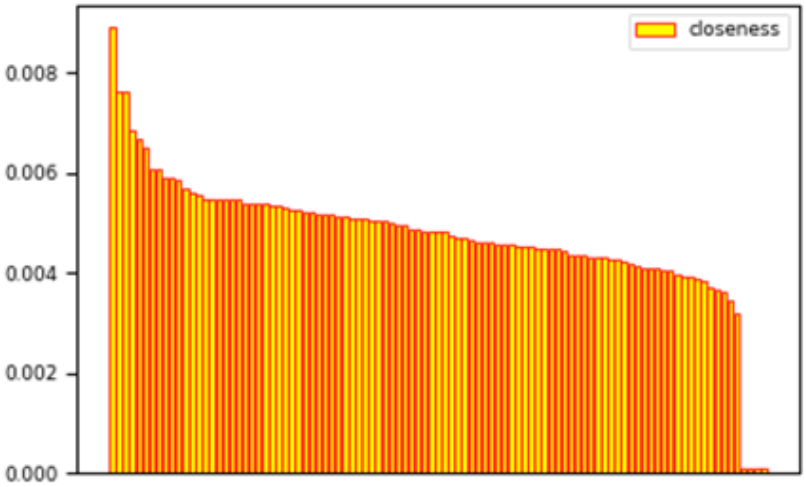
值得注意的是，在“经济”网络中，“知乎认证”这一特征在测量三个维度的中心度时作用为零，这说明在“经济”话题领域下，**“知乎认证”作用对于定位和识别真正的意见领袖作用微弱。**

（3）可视化——网络内的用户特征

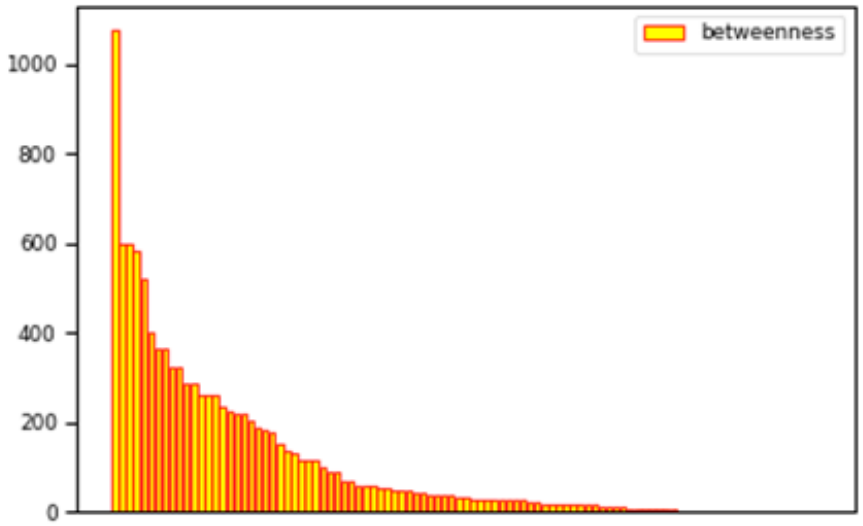
Economy		
	1	0
if_identify	0	99
if_super	27	72

如上图所示，“经济”话题内的传统意见领袖基本上都会被知乎认证，但这其中只有大部分用户成为了优秀回答者。**这实际上印证了我们前面的推测，“知乎认证”作用对于定位和识别真正的意见领袖作用微弱，只有传统意见领袖的回答质量足够优秀，才能成为真正的意见领袖。**





如上图所示，在选取的五个话题中，相比于其他四个话题，“经济”话题的度中心性（degree）是最高的。我们可以推测，在“经济”话题内，真正的意见领袖往往处于与之直接相连的邻居用户中的中心位置，对其他知乎用户的行为会产生重要影响。知乎用户对“经济”话题广为讨论，话题内的舆论导向会常常变动，导向主要由意见领袖引导。所以，意见领袖在“经济”话题内的团体网络位置十分重要。



如上图所示，但“经济”话题下的接近中心性（closeness）、中介中心性（betweenness）却处于五个领域中的居中位置。这说明“经济”话题内的传统意见领袖之间互动信息较少，很多用户对传统意见领袖的信服度较低，并不愿意关注传统意见领袖本身，而更愿意关注所感兴趣的话题。**所以经济领域内只有当传统意见领袖获得更高的“知乎认证的回答数”时，才是真正的意见领袖。**

综上，我们在“经济”话题领域识别真正的意见领袖，**需要重点关注其“关注数”、“知乎认证的回答”是否较高**，这对其是否是真正的意见领袖有较大影响。而不是传统意义上寻找其是否是知乎认证的“专业”用户。

（二）研究结果

1、在十二个用户特征中，“关注数”、“回答数”、“点赞、感谢、收藏数”对于识别“法律”话题下传统意见领袖中真正的意见领袖具有显著作用，其他的特征影响较小。

2、在十二个用户特征中，“关注数”、“点赞、感谢、收藏数”、“粉丝数”对于识别“政治”话题下传统意见领袖中真正的意见领袖具有显著作用，其他的特征影响较小。

3、在十二个用户特征中，“关注数”、“回答数”对于识别“社会”话题下传统意见领袖中真正的意见领袖具有显著作用，其他的特征影响较小。

4、在十二个用户特征中，“关注数”、“粉丝数”对于识别“文化”话题下传统意见领袖中真正的意见领袖具有显著作用，其他的特征影响较小。

5、在十二个用户特征中，“关注数”、“知乎认证的回答”对于识别“经济”话题下传统意见领袖中真正的意见领袖具有显著作用，其他的特征影响较小。

五、研究总结

（一）政策建议

综上，本文对政府在社会舆情的管理和引导方面提出以下政策建议：

1、**将社会舆情按话题领域区分，对症下药**，根据本研究的研究结果进行传统意见领袖中具有真正影响力的意见领袖的定位与识别，借助他们的力量对社会舆情进行更好更精确的管理和引导；

2、**与知乎等社会网络平台联手合作**，创新工作思维和管理方式，借助意见领袖等主体在无形中落实工作，达到社会舆情管理和引导的良好效果；

3、本研究为知乎平台五大话题领域意见领袖的定位和识别建构一套具有创新性的体系，建议政府可与相关数据平台合作，将此体系推广应用，在实践中加以验证和完善，并不断构建更多的网络平台的意见领袖识别体系，**将大数据与政府治理结合得更加紧密，让数据更好为治理服务。**

（二）研究不足

1、本研究虽然基于社会网络分析，但是分析层次较浅，尚未站在网络集群和子群的角度分析各个领域网络的内在特征，也仅仅基于静态网络的分析，并不能真正还原和透析时事热点在社会网络中的传播规律。

2、本研究缺乏对用户更多元的特征，比如用户的性格特征，可以通过文本分析获取。这些更加多元的特征有助于将研究的价值推广到全社会网络中意见领袖的定位，将研究范畴同社会心理学和组织行为学联系起来。

3、对于意见领袖网络中的意见领袖的确认，我们简单的采取二分类的方式过于粗暴，也不利于进一步探究其在网络中的信息影响能力，获得的结论和见解因此也是较浅薄。

（三）研究展望

本文中运用的中心性分析方法，以一种定量的方法对意见领袖进行识别。我们期待我们的研究结果可以为政府提供一个更快速、更准确地识别、定位知乎意见领袖的方法，更好地控制知乎上舆情发展的态势。我们的研究有很多不足的地方，且此结论不是一成不变的，随着时间的推移在不断变化，我们认为，此后的相关研究可以在以下几个方面进行深入的探究：

一方面，对于知乎意见领袖的研究可以从凝聚子群，派系等角度对意见领袖的产生因素进行深入的分析研究；另一方面，我们的研究缺少对于具体的社会事件的分析，期待此后的研究可以从具体的社会事件出发进行系统的研究探讨。

参考文献：

[1]吴江,赵颖慧,高嘉慧.医疗舆情事件的微博意见领袖识别与分析研究[J].数据分析与知识发现,2019,3(04):53-62.
[2]施艳萍,袁曦临,宋歌.社会化问答平台意见领袖的知识共享行为特征探析[J].图书情报知识,2018(06):103-112.
[3]蔡骐,陈月.基于社会网的知乎网意见领袖研究[J].湖南师范大学社会科学学报,2018,47(05):128-138.
[4]陈敏,黄睿.“大V”去哪儿了?——基于微博、微信、知乎南海仲裁案讨论文本的分析[J].新闻记者,2018(07):61-72.
[5]应雨辰,纪浩,梦非.网络意见领袖识别模型构建与实证分析[J].软件导刊,2017,16(11):163-167.
[6]王腾. 网络社区意见领袖的识别及其影响分析[D].首都经济贸易大学,2017.
[7]唐巧盈. 知乎意见领袖的形成及影响机制研究[D].上海社会科学院,2017.
[8]陈静. 网络舆情传播中知乎意见领袖的影响机制[D].武汉大学,2017.
[9]刘雨农,刘敏榕.社会化问答平台的社区网络形态与意见领袖特征——以知乎网为例[J].情报资料工作,2017(02):106-112.
[10]陈远,刘欣宇.基于社会网络分析的意见领袖识别研究[J].情报科学,2015,33(04):13-19+92.
[11]王秀丽.网络社区意见领袖影响机制研究——以社会化问答社区“知乎”为例[J].国际新闻界,2014,36(09):47-57.
[12]王君泽,王雅蕾,禹航,徐晓林,王国华,曾润喜.微博客意见领袖识别模型研究[J].新闻与传播研究,2011,18(06):81-88+111.
[13]刘志明,刘鲁.微博网络舆情中的意见领袖识别及分析[J].系统工程,2011,29(06):8-16.

附录：源代码

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="MjgxZTlxZjQ1ZmVmNDlhODhmMjgwMjJkOThlZGUzNDg=|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|965
```

```
bb1a73087c52ebf3256a999cbdb8b14240fb9";
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="AJAgybl8iA2PTiF1-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_c
ount%2Cis_followed%2Cis_following%2Cbadge%5B%3F(type%3Dbest_answerer)%5D.topics&offs
et=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_c
ount%2Cis_followed%2Cis_following%2Cbadge%5B%3F(type%3Dbest_answerer)%5D.topics&offs
et="+ 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_c
ount%2Cis_followed%2Cis_following%2Cbadge%5B%3F(type%3Dbest_answerer)%5D.topics&offs
et="+ 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_c
ount%2Cis_followed%2Cis_following%2Cbadge%5B%3F(type%3Dbest_answerer)%5D.topics&offs
et=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[40:9
9]), 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[60:9
9]), 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[80:9
```

```
9]), 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(betweenn
ess[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))
```

R 语言社会网络分析源码

```
library(sna)
library(statnet.common)
library(network)
library(igraph)
social <- read.table(file = "C:\\Users\\NHT\\Desktop\\social_graph.txt",header
= T)
social <- social[,-1]
colnames(social)<- NULL
social <- as.matrix(social)
gplot(social)
gden(social)
grecip(social)
connectedness(social)
efficiency(social)
mean(distances(graph_from_adjacency_matrix(social)))
max(distances(graph_from_adjacency_matrix(social)))
degsoc <- degree(graph_from_adjacency_matrix(social))
closoc <- closeness(graph_from_adjacency_matrix(social))
betsoc <- betweenness(graph_from_adjacency_matrix(social))
write.csv(degsoc,"degsoc.csv")
write.csv(closoc,"closoc.csv")
```

```
write.csv(betsoc,"betsoc.csv")
```