2018年9月13日星期四

上午10时31分

通过初步分析CERT5.2第二个攻击场景中的某些用户周围的离职情况，有以下结果：

用户163个

['**BYO1846**', '**2010-12**', ['Buffy Yolanda Ortiz', 'BYO1846', 'Buffy.Yolanda.Ortiz@dtaa.com', 'Salesman', '', '1 - Executive', '5 - SalesAndMarketing', '2 - Sales', '5 - RegionalSales', 'Donna Erin Black']] 分析

经过部门筛选分析，得到

同一团队的离职员工为: 3

相同部门下不同个团队的离职员工为: **16**

相同职能部下不同部门的离职员工为: 15

同一事业部下不同职能部的离职员工为: 46

不同事业部的离职员工为: 82

分析新的一个内部跳槽用户：

['CHP1711', '2010-10', ['Cody Hall Phelps', 'CHP1711', 'Cody.Hall.Phelps@dtaa.com', 'Salesman', '', '2 - Executive', '5 - SalesAndMarketing', '2 - Sales', '4 - RegionalSales', 'Ethan Branden Garrett']]

同一团队的离职员工为: 2

相同部门下不同个团队的离职员工为:  **7**

相同职能部下不同部门的离职员工为: 10

同一事业部下不同职能部的离职员工为: 43

不同事业部的离职员工为: 61

可以发现，与用户关系最为密切的是同一个团队，离职人数在3/2人，然后考虑同一部门下不同小组，发现有16/7人，最后考虑同一个职能部门下的不同部门，发现相近有15/10人。再往下考虑不同职能部的关系，则人数突然增长为46/43人，建议前期先不考虑。

2018年9月13日星期四

下午2时44分

接着上午的工作

自己猜测：是不是跳槽的员工与普通员工之间，人际关系中离职人数存在差异呢？

为此，自己决定写程序分析全部2000个用户中，其接触的离职人员个数，且分为上述五个层次：

1. 同一个团队
2. 同一部门不同团队
3. 同一职能不同部门
4. 同一事业部不同职能
5. 不同事业部

即存储格式为：

User\_id, a,b,c,d,e

2018年9月17日星期一

下午4时36分

上周，经过初步的统计分析，得到了CERT5.2中所有用户的离职情况分析：

1. 同一个团队
2. 同一部门不同团队
3. 同一职能不同部门
4. 同一事业部不同职能
5. 不同事业部

其中部分用户的第一个维度特征（同一个团队中离职人数）很少，一部分则很多。因此，为了进一步清晰分析出前三个维度层次的作用，计划首先分别针对三个层次来绘制图像，然后针对前二个维度/前三个维度的单个用户和来绘制图像，其中CERT5.2的场景二用户用红色标识。

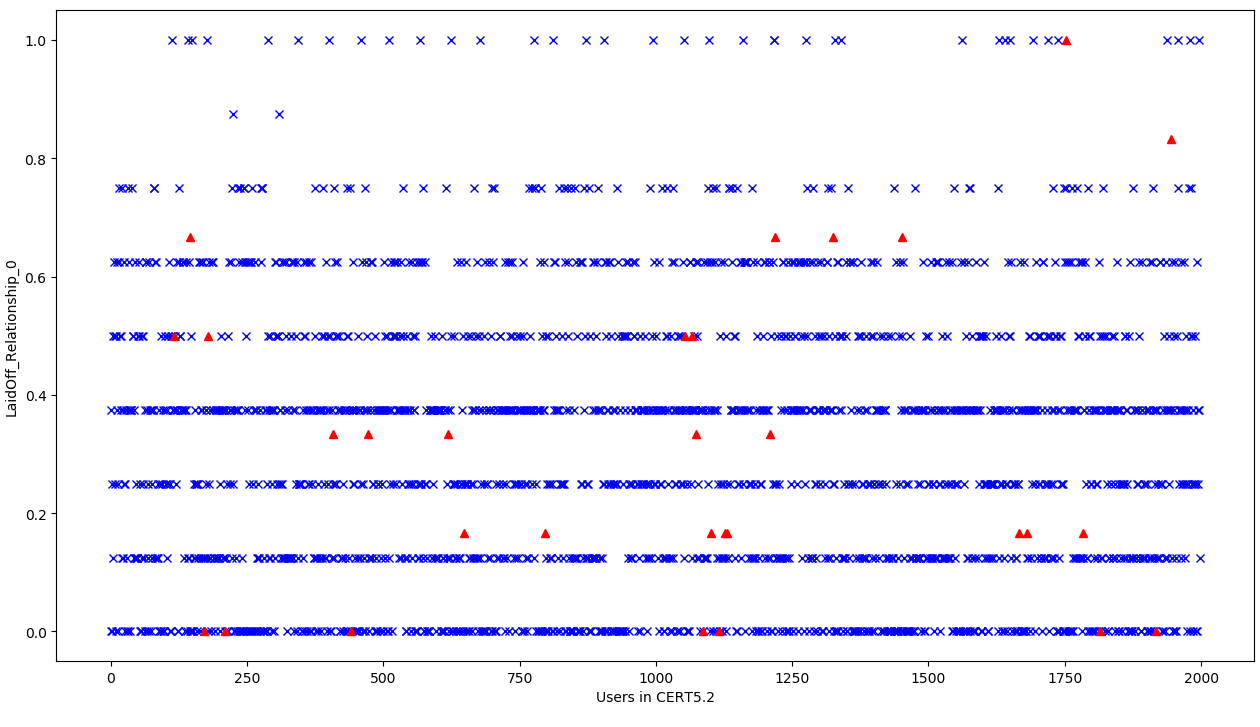
2018年9月19日星期三

上午10时51分

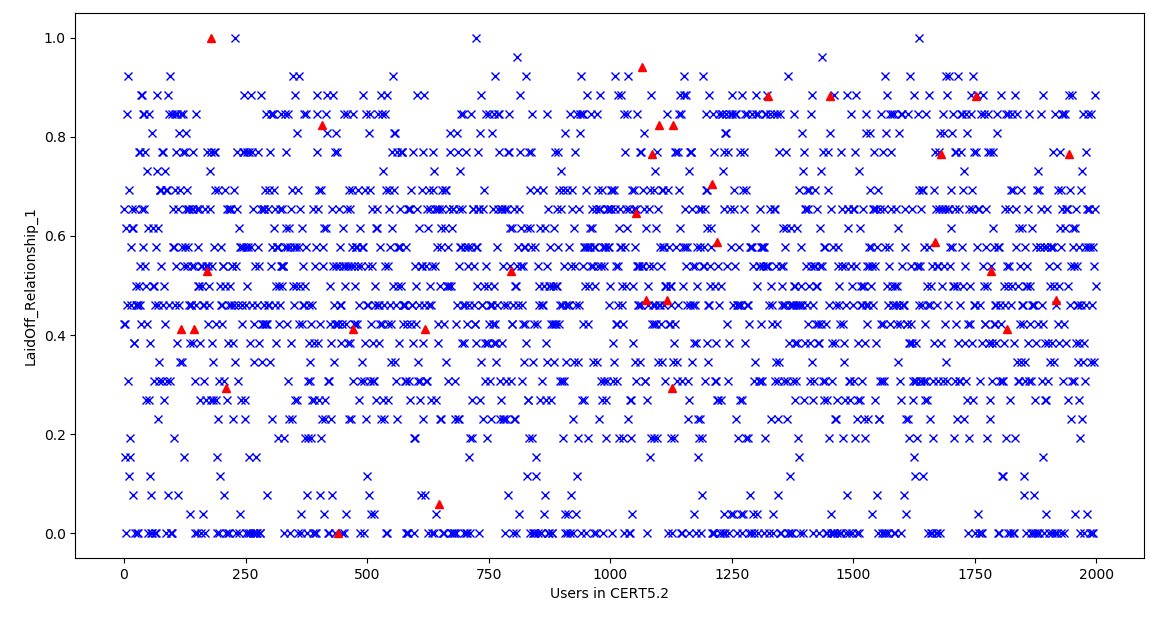
今天周三，终于开始完成上述实验。

经过统计分析1999个用户（没考虑CEO）

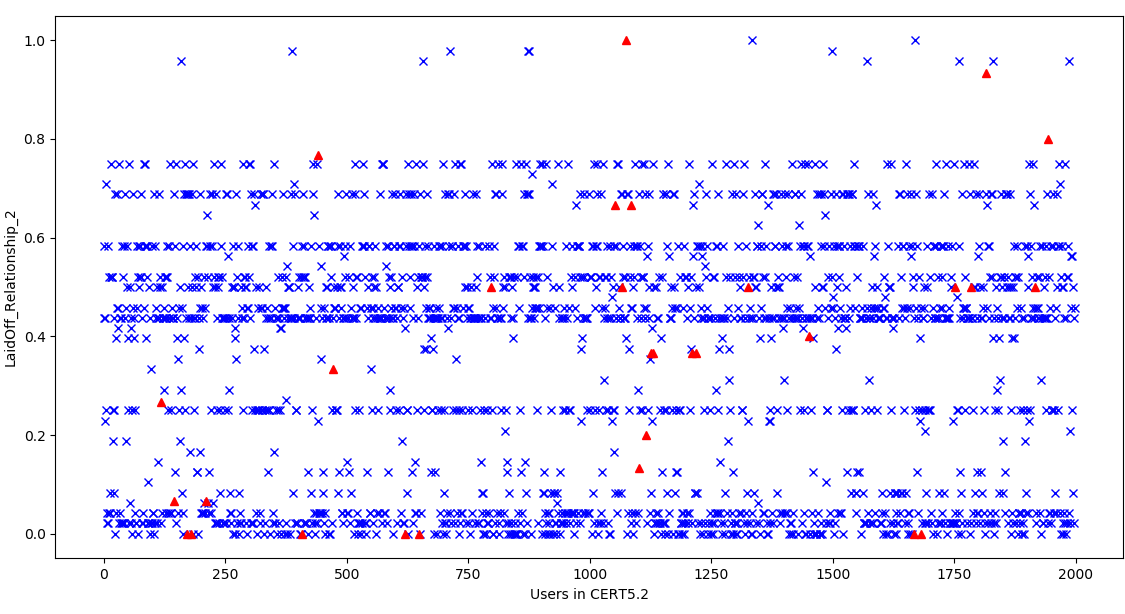
对于同一团队而言（LaidOff\_Relationship\_MinMax\_0)）



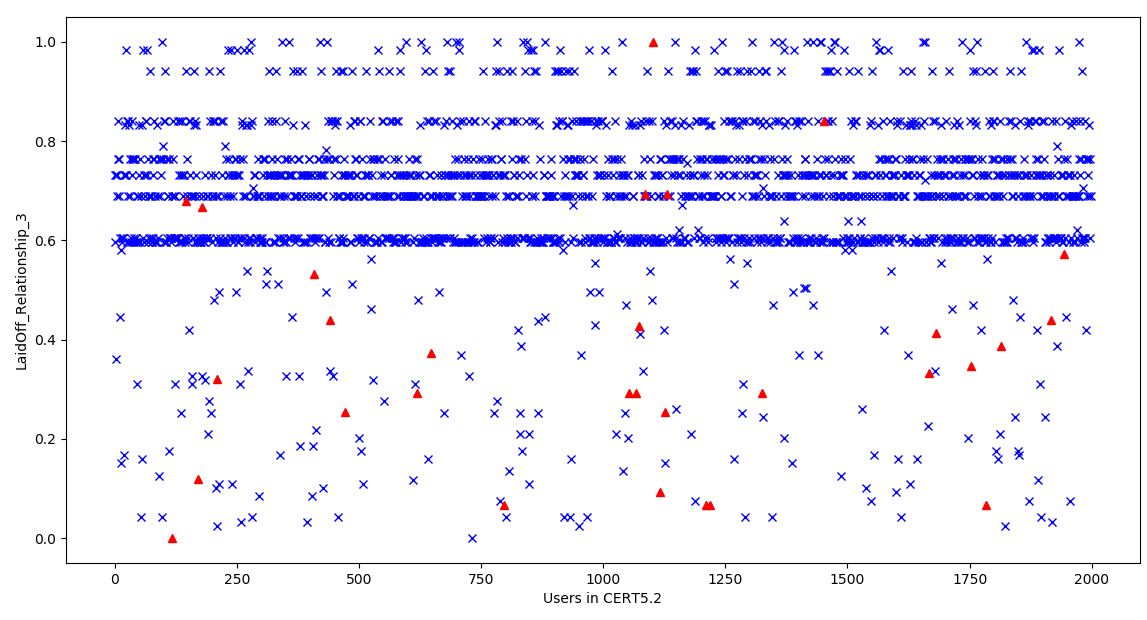
考虑同一部门内不同团队（LaidOff\_Relationship\_MinMax\_1）



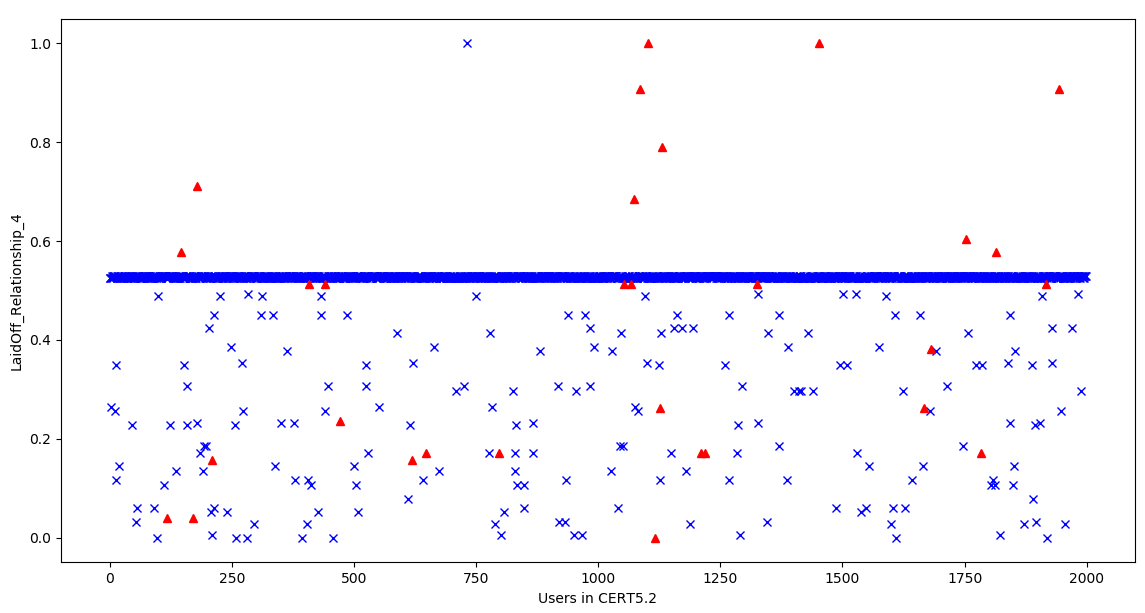
对于同一职能部门内不同部门（LaidOff\_Relationship\_MinMax\_2）:



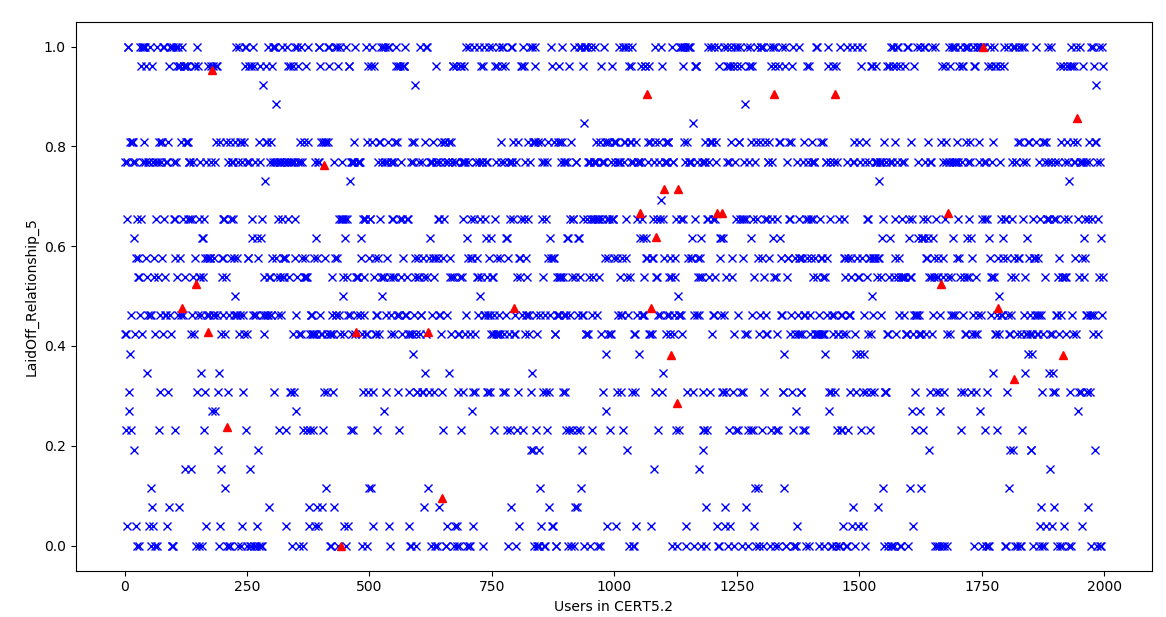
对于同一事业部内不同职能部门（LaidOff\_Relationship\_MinMax\_3）:



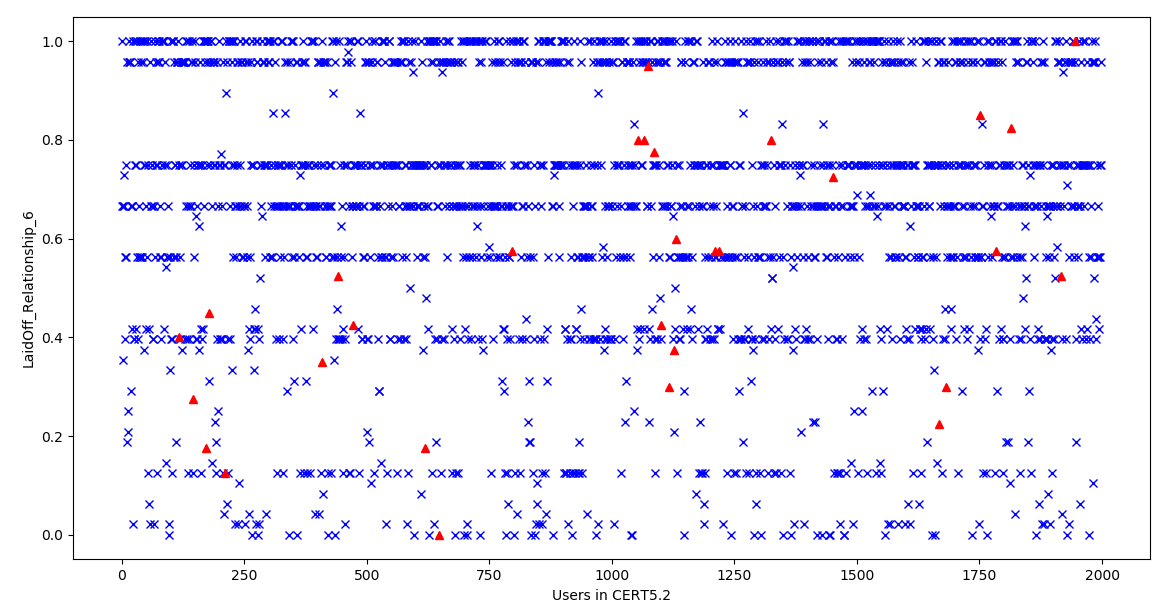
对于不同事业部（LaidOff\_Relationship\_MinMax\_4）:



对于同一部门内离职人员（LaidOff\_Relationship\_MinMax\_5）:



对于同一职能部门内离职人员（LaidOff\_Relationship\_MinMax\_6）:



2018年9月19日星期三

下午10时17分

睡前小结，整理思路。

目前已有的实验分析结果有：

1. CERT5.2数据构造说明中提到了隐含的动态的工作满意度变量，并在因果关系图中，由用户的人际关系Relationship与离职人员LaidOff共同决定了用户的工作满意度Job Satisfactory，而JS又决定了用户的跳槽行为（场景二）与不守时行为；
2. 现有实验分析了基于组织架构层次的用户周围离职人员关系，发现对于场景二而言，不存在明显的区分度，即单单从离职人员关系数量角度无法有效区分场景二跳槽用户；
3. CERT5.2数据说明中，提到了每个用户赋予的初始静态OCEAN大五人格分数，其中外倾性E驱动着用户交朋友的数量多少；而尽责性C驱动着迟到行为的多少。
4. 原有数据图中说明：...relationships to laid off employees affect job satisfaction；字面意思理解即与解雇人员的关系影响着用户的工作满意度，可以推测，一般解雇用户会不会影响我们的工作满意度呢？那要看我们与这个人的关系如何，如果是好朋友或者熟识的关系，那么就影响比较大；如果仅仅是一般同事的关系，那么可能影响比较小。依据本点分析，之前的工作借助了LDAP数据分析了CERT5.2中的部门组织结构，并在此基础上统计了每个用户相关的五种层次解雇人员数量；之后只需要分析邮件通讯确定好友关系即可（邮件往返达到某个阈值，或者超过一般水平等，是否要借助分析Topic尚且未定）。但是该方法考虑未涉及到用户的OCEAN数据。

因此，我的一个初步想法是：

* 选择一个场景二跳槽用户
* 分析该用户每个月的邮件通讯情况，列出与其通信的用户，以及send/recv的封数；
* 基于每月可以对通讯数量进行一个排序，越靠前的联系越紧密，越重要；
* 查看离职用户是否在紧密关系中；
* 考虑python实现一个图结构

1. 明天的一个主要任务是提取CERT5.2的数据中的用户月记录，即按照不同用户，将其每月的五类数据记录按照天记录为相应文件，在单位台式机上部署，分析数据结果；
2. 同时第二个任务是通过python绘制CERT5.2的用户邮件图（按月积累，即第一个月，前两个月，前三个月...）

2018年9月27日星期四

下午3时10分

今天是国庆假期前第四天，继续中秋之前的工作。

目前已经在中心台式机上部署了数据提取程序，基本上每天可以提取整理100个用户，预计需要三周时间完成全部的数据提取工作。

等不及那么久，我们先将30个场景二用户的数据提取完毕，首先进行JS变量分析。

今天主要计划完成两个实验分析。

1. 首先简单基于原始的用户离职关系，即每个用户的五个层次的离职用户数目，做一个自动PCA的聚类，然后查看场景二用户聚类是否存在规律或者可区分；
2. 根据上一步的结果，试着提取分析一个特定场景二用户的邮件数据，建立邮件关系图模型，从中分析出‘亲密度度量假设’，即不同的人际关系其亲密度不同，结果导致不同人离职的影响也不同，我们只侧重那些更加亲密的人的被解雇行为；
3. 上述实验分析中，需要重新进行一个统计，即我们关注的是被解雇人员，而非离职人员，因此应当从各个用户的离职关系列表中去掉主动离职的那些人；（依据CERT数据说明，JS相关的是那些被解雇的用户，而非主动跳槽离职的用户），即需要剔除场景1-2-3的攻击用户，保留场景四的攻击用户

因此，我们首先的工作是得到一个仅考虑laidoff用户的新数据。

经过程序修改，我们得到了两个用户关系数据：

一个包含了所有离开公司的用户数据，以Leave标识，即

CERT5.2-Leave-Relationship.csv (2000个用户的组织架构中离开公司的人员表，没统计CEO，因为CEO没有LDAP架构关系)

CERT5.2-Leave-Relationship-Counts.csv（1999个用户的组织架构中离开人员数量表）

CERT5.2-Leave-Relationship-Statistical.csv（试着对上述数量表进行了MinMax归一化）

另外一个则提出了三个场景中的主动离职用户，仅考虑剩余的Leave公司的用户：

CERT5.2-LaidOff-Relationship.csv

CERT5.2-LaidOff-Relationship\_Counts.csv

接下来，我们需要进行第一个实验，即原始离职用户数目的KMeans实验。

针对f\_1\_Leave与f\_2\_LaidOff两组数据，我们分别用轮廓系数自动选择最优的K值（未事先PCA），有

对于f\_1\_Leave而言，有：最好的K值为2

各个K值的K均值聚类的最优轮廓系数为：

K值为： 2 轮廓系数为 0.7451675437027069

K值为： 3 轮廓系数为 0.5242322608775718

K值为： 4 轮廓系数为 0.5342708699462355

K值为： 5 轮廓系数为 0.5361262200592684

K值为： 6 轮廓系数为 0.5162626464065303

K值为： 7 轮廓系数为 0.5163122774501875

K值为： 8 轮廓系数为 0.5338989394875452

K值为： 9 轮廓系数为 0.5482335534608274

K值为： 10 轮廓系数为 0.595114640915684

对于f\_2\_LaidOff而言，直接轮廓系数分析最佳K值也为2

各个K值的K均值聚类的最优轮廓系数为：

K值为： 2 轮廓系数为 0.7248503877805943

K值为： 3 轮廓系数为 0.5192092410005896

K值为： 4 轮廓系数为 0.5239977272493982

K值为： 5 轮廓系数为 0.5062004405472608

K值为： 6 轮廓系数为 0.4775261900571844

K值为： 7 轮廓系数为 0.498805947031619

K值为： 8 轮廓系数为 0.5019477858512982

K值为： 9 轮廓系数为 0.5385688611613426

K值为： 10 轮廓系数为 0.5511768551731365

从直接结果上看，最佳K值都是2，对于2000个用户而言未免太少了，因此直接尝试用自动PCA降维后再尝试KMeans

对于f\_1\_Leave而言，同样最好的也是2

各个K值的K均值聚类的最优轮廓系数为：

K值为： 2 轮廓系数为 0.7451675437027069

K值为： 3 轮廓系数为 0.5242322608775718

K值为： 4 轮廓系数为 0.5427688057826405

K值为： 5 轮廓系数为 0.5361262200592684

K值为： 6 轮廓系数为 0.5162626464065303

K值为： 7 轮廓系数为 0.5188816180553412

K值为： 8 轮廓系数为 0.5362025192705445

K值为： 9 轮廓系数为 0.550218257156329

K值为： 10 轮廓系数为 nan

其中轮廓系数最高的K值为:

2 0.7451675437027069

自己继续指定了PCA=2，结果相同

其中轮廓系数最高的K值为:

2 0.7451675437027069

如果换成是f\_2\_LaidOff，采用PCA=2，则结果为

其中轮廓系数最高的K值为:

2 0.7248503877805943