二〇一九年一月十日星期四

下午3时51分

由于自己之前的客观行为检测实验使用的是CERT4.2数据集的1000个用户，而18年9月份以来做主观检测实验不知道为什么用的竟然是CERT5.2数据集的2000个用户，结果就是，现在需要融合两类结果，首要任务是将主观检测的实验移植到CERT4.2上，并且希望结果一样有效。

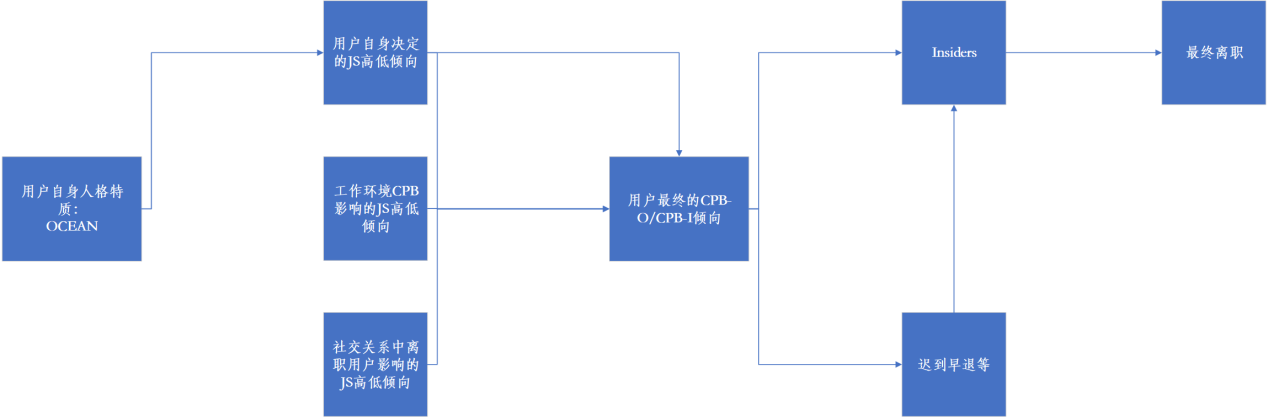
借着这个机会，我们将整体的主观检测实验过程过一遍，从而为日后融合实验与写作论文打好基础。

1. 心理模型

整体的心理模型基于几个重要关联：

1. 用户反生产行为与Insiders攻击的直接联系；
2. 用户的大五人格与反生产行为倾向的联系（个体人格直接影响+工作满意度的间接影响）

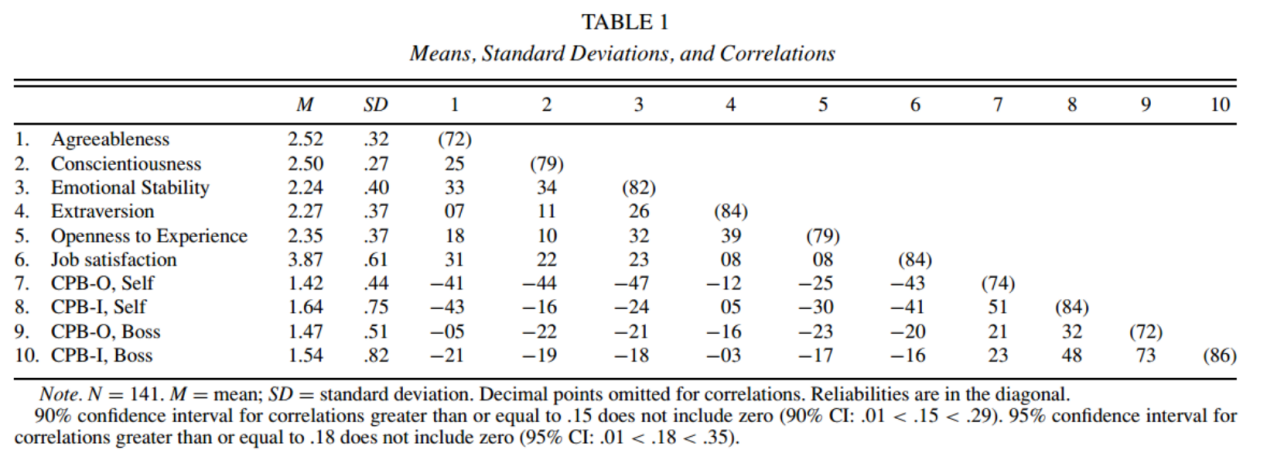
考虑内部攻击者攻击倾向的刻画，一种来自于自身人格因素决定的反生产行为倾向（CPB-I/CPB-O），一种来自于工作满意度的侧写；工作满意度的侧写又需要从自身人格决定的基本满意度状态（OCEAN:JS）、团队环境积极友善程度（团队虚拟人格的CPBs）、直接领导者的CPBs、缺勤情况统计以及与离职用户的通信联系情况（虚拟离职联系人的Email\_Info）来共同表述。



训练OCSVM遍历参数空间以确定最优参数空间（召回率优先，其次是误报率，或使用公式，上述公式综合考虑了召回率与误报率的重要性，默认x=0.9，实际实验可以依据结果修正）。实际实验中选择了x=1.0的Recall最高原则筛选；

需要声明的一点，最初的想法是在训练第N个月的OCSVM时，参数选择过程的验证集包含既往1:N-1个月离职用户的ATF特征，如此需要ATF中涉及数量的尽量转换成均值；而结果中由于设定了以召回率优先，会导致得到的OCSVM中FP过高；因此，暂时不考虑将验证过程引入上个月之前的离职用户ATF，但是ATF中涉及到虚拟人格的暂时全部均值化以统一处理方法。

此外，此次计算ATF由于需要计算JS分数，因而建议使用参考文献[10]中的完备结果，其中与CPB的关系建议采用用户自我评价，原因是自己更清楚真实的感受，而满意度无疑是决定于自我感觉的；其次可以发现关于CPB-O与C的关联性个体评价更显著，**因此使用个体评价取代Boss评价。**

为此，可能需要重新计算CERT5.2中各个用户团队样本中的CPB分数，并且以此计算JS分数；  


最终，我们的ATF特征应具有以下字段

|  |  |
| --- | --- |
| 自身人格特质因子 | OCEAN五个分数(5)[实际使用中删除了] |
| 反生产行为倾向CPB-I/O (2) |
| 工作满意度因子 | 自身OCEAN映射的JS分数（1） |
| Team中CPBs状态：(8)  Team\_CPB-I-mean,Team\_CPB-O-mean,  Users-less-mean-A,Users-less-mean-A and C,  Users-less-mean-C,Users-High-mean-N,  Team\_CPB-I-median,Team\_CPB-O-median |
| Leader的CPBs(2) |
| Leave\_Contacts\_EmailInfo:(9)  需全部转变为均值/len(Leave\_Contacts)  dis\_ocean, avg\_dis\_ocean, dis\_os, avg\_dis\_os, email\_ratio, cnt\_send/recv, cnt\_s/r\_size, cnt\_s/r\_attach, cnt\_s/r\_days, cnt\_email\_days |
| 缺勤特征（3）  cnt\_late\_days, cnt\_early\_days, //转变为比例  month\_work\_days, |

1. 实验数据准备

本部分主要说明如何一步一步构建主观检测实验所需要的用户主观特征ATF。

1. 用户工作环境因子影响的CPB因素

直接使用之前提取的CERT4.2-2009-12-New-26JS.csv文件，自动选取其中Team\_CPB与Leader\_CPB部分即可；

1. CERT4.2的基本数据集

包括Personality/LDAP-OS结构/Insiders的GT数据

由于自己编写的程序代码过于混乱，于是重新，从头对于主体实验v01-v09的全部实验文件进行注释说明

|  |  |  |
| --- | --- | --- |
| 文件名 | 模块 | 功能 |
| V09-LDAP\_AllUsers.py | 出勤率 | 获取CERT4.2中的四元部门及对应用户列表，便于进行后续的dis\_OS特征提取 |
| V09\_Extract\_Logon.py | 出勤率 | 顾名思义：将统一的Logon数据按照2010-01月份不同的用户进行梳理归类，便于计算用户的习惯上下班时间--Team上下班时间--用户迟到早退特征 |
| V09\_Extract\_WorkTime.py | 出勤率 | 提取用户的习惯上下班时间 |
| V09\_Extract\_Team\_WorkTime.py | 出勤率 | 提取用户所在Team的标准上下班时间 |
| V09\_Check\_Early\_Late.py | 出勤率 |  |
|  |  |  |
| V09-LaidOff Insiders.py | 离职用户列表 | 1. 统计CERT4.2中的Leave用户，并写入文件（需要前置获取用户最后一次登录的日期天） 2. 按用户提取各自的Logon数据，并写入Logon目录 |
| CERT4.2-Leave-Users\_OnDay\_0.9.csv | 离职用户列表 | 上述离职用户提取的结果，2010-01:2011-04的离职用户、最后一天工作日期以及其LDAP |
|  |  |  |
| AAE0190\_LCE\_Feats.csv | LCE特征 | 单个用户与Leave Contacts通信的Email四元特征 |
| CERT4.2\_Users\_LeaveContacts\_EmailFeats.csv | LCE特征 | 所有用户集合的按月份排列的LCE特征 |

二〇一九年一月十一日星期五

上午8时55分

CERT4.2复现实验的数据验证：

1. V09\_Check\_Early\_Late.py验证2010-01

*Format: User\_id, workon time, workoff time, cnt\_late, cnt\_early, all work days, the days with late and early at the same time*

NGF0157,7.0,17.0,8.0,4.0,23,3,2010-01-04,2010-01-22,2010-01-29,

验证通过！在中心台式机上开始CERT4.2用户的全面出勤率统计

1. CERT4.2用户邮件内容归档

数据格式：

Fields: id, date, user, pc, to, cc, bcc, from, size, attachment\_count, content

{R3I7-S4TX96FG-8219JWFF},01/02/2010 07:11:45,LAP0338,PC-5758,Dean.Flynn.Hines@dtaa.com;Wade\_Harrison@lockheedmartin.com,Nathaniel.Hunter.Heath@dtaa.com,,Lynn.Adena.Pratt@dtaa.com,25830,0,middle f2 systems 4 july techniques powerful destroyed who larger speeds plains part paul hold like followed over decrease actual training international addition geographically side able 34 29 such some appear prairies still 2009 succession yet 23 months mid america could most especially 34 off descend 2010 thus officially southward slope pass finland needed 2009 gulf stick possibility hall 49 montreal kick gulf

验证通过！

1. 离职用户统计

CERT4.2的用户数据相比CERT5.2不规范，尤其是2011-05月的审计数据竟然不是到2011-05-31

需要先判断一次CERT4.2的所有Insiders的最后一次离职时间是多少？

为此，决定先对所有用户的最后登录时间进行升序排序，查看最后登录的时间；

经过分析：

CERT4.2中共有154个用户的最后一次登录时间在2011-05之前，最晚的一次是在

FSB0399,2011-04-29,

NAF0033,2011-05-12,

CEL0561,2011-05-16,

而2011-05最后一次登录系统的有1000-154=846个用户。故使用前154个用户作为Leave用户，且Insiders\_1/2/3均在2011-05前离职，因此，直接只考虑2010-01:2011-04的离职情况即可；

最终得到了所需的CERT4.2用户离职文件数据：CERT4.2-Leave-Users\_OnDay\_0.9.csv

其中包括154个离职用户（不考虑2011-05月份）

二〇一九年一月十一日星期五

下午4时37分

终于静下心来写了提取四元LCE特征的程序，运行一次测试验证：

User: AAE0190 确实是两个，日期也对，验证通过，开始1000用户的提取

Lcontact:Buffy.Vivien.Nichols@dtaa.com

2010-06-08,BVN0514,2.0['2010-05-18'; '2010-06-04'],66643.0,0.0

Line 1303: 05/18/2010 15:42:23,

AAE0190,PC-8915,Buffy.Vivien.Nichols@dtaa.com,Whoopi.Britanney.Camacho@dtaa.com,,August.Armando.Evans@dtaa.com,31885,0,

Line 1478: 06/04/2010 18:22:36,

AAE0190,PC-8915,Callum.Scott.Parrish@dtaa.com,Sawyer.Abel.Turner@dtaa.com;Buffy.Vivien.Nichols@dtaa.com;Eugenia.Whilemina.Harrington@dtaa.com,,August.Armando.Evans@dtaa.com,34758,0,

1. 构建均值化的LCE特征/每个月

现在就差最后一步就可以直接计算每个月的ATF和静态ATF了！

然后就可以重新实验了！

但是前提是将每个月在职用户的LCE计算出来：

副产品：

1. 得到每个月的离职用户列表以及时间+LDAP

下午10时10分

开始改写之前的程序，由所有用户的LCE特征生成2010-01:2011-04每月的用户的ATF特征以及每月的离职用户列表。

每月的ATF中关键的LCE部分是截止到当月所有用户（包含当月的离职用户）的数据的，如此才可以实现按月的Risk预测，如知晓了第N月的离职用户的ATF，才能对其进行高危预测；

最初的CERT42\_Users中包含所有用户，进入到2月份时去掉1月份离职的用户，如此类推；

先生成了2010-02的LCE特征，验证：

CEL0561,40.0,39.0,36.0,19.0,40.0,-7.81,-20.47,45.6043474837,22.8021737418,30.0,15.0,**21.0,658953.0,24.0,16.0,**

2010-02-11,DGM0754,1.0,['2010-01-29'],20578.0,0.0

2010-02-17,LDD0560,20.0,['2010-01-11'; '2010-01-14'; '2010-01-19'; '2010-01-20'; '2010-01-22'; '2010-01-25'; '2010-01-28'; '2010-01-28'; '2010-02-01'; '2010-02-02'; '2010-02-05'; '2010-02-05'; '2010-02-09'; '2010-02-10'; '2010-02-10'; '2010-02-11'; '2010-02-15'; '2010-02-16'; '2010-02-16'; '2010-02-16'],638375.0,24.0

上述验证通过！

二〇一九年一月十二日星期六

下午7时16分

今天晚上，开始CERT4.2数据集上，全体用户的KMeans+OCSVM的复现实验。

1. 目标：验证Static方法是否可以在CERT4.2上应用；
2. 目标：验证RealTime方法是否可以在CERT4.2上应用（逐月应用KMeans+OCSVM检测当月离职用户中的Insiders）

实际构造ATF时突然发现竟然还没有计算LED特征，于是赶紧补上：

在Build\_ATF中直接添加LED模块即可，因为前期已经得到了每月的用户LED特征，只需要根据每月用户进行累加构造ATF的最后三个特征即可。

按月的累积ATF编写完毕，开始验证：

2010-03

user\_id,O\_Score,C\_Score,E\_Score,A\_Score,N\_Score,CPB-I,CPB-O,JS\_Score,Team\_CPB-I-mean,Team\_CPB-O-mean,Users-less-mean-A,Users-less-mean-A and C,Users-less-mean-C,Users-High-mean-N,Team\_CPB-I-median,Team\_CPB-O-median,leader-CPB-I,leader-CPB-O,dis\_ocean,dis\_os,cnt\_send,cnt\_send\_size,cnt\_send\_attach,cnt\_send\_days,cnt\_email\_days,cnt\_late\_days,cnt\_early\_days,month\_work\_days,

CEL0561,40.0,39.0,36.0,19.0,40.0,-7.81,-20.47,11.35,-14.0682666667,-17.6715733333,10,5,8,13,-10.164,-10.164,-9.8596,45.6043474837,46.0,21.0,658953.0,24.0,16.0,**22.0,17.0,63.0**,

单项验证：

Calvin Edan Love,CEL0561,40,39,36,19,40

CEL0561,40.0,39.0,36.0,19.0,40.0,-14.0682666667,-17.6715733333,10,5,8,13,-10.164,-10.164,-9.8596,

CEL0561,40.0,39.0,36.0,19.0,40.0,-7.81,-20.47,45.6043474837,46.0,21.0,658953.0,24.0,16.0,

2010-03：CEL0561,9.0,18.0,7.0,5.0,23,1,2010-03-16,

2010-02：CEL0561,9.0,18.0,10.0,6.0,20,5,2010-02-02,2010-02-03,2010-02-04,2010-02-09,2010-02-24,

2010-01：CEL0561,9.0,18.0,5.0,6.0,20,3,2010-01-05,2010-01-13,2010-01-15,

2010-04：CEL0561,9.0,18.0,5.0,7.0,21,4,2010-04-08,2010-04-09,2010-04-26,2010-04-29,

验证通过！

晚上回家开始验证Static模式的CERT4.2数据集的KMeans+OCSVM

在继续实验前，需要先整理出CERT4.2中Insiders\_1/2/3的用户列表即离职日期

1. 获取Insiders列表
2. 获取Leave Users列表

二〇一九年一月十三日星期日

上午12时7分

在CERT4.2上进行实验：

Miu = 1.0

PCA=1

0 [0.05, 1.0, **0.6714285714285714**, 0.6714285714285714, 0.02843601895734597, 0.05052526263131566, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **0.7666666666666667, 0.5666666666666667, 0.7**,

**如果采用PCA=1，但是nu最小0.01**

**0.01, 1.0, 0.7714285714285715, 0.7714285714285715, 0.028962611901000527, 0.054527263631815905, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',**

**max\_iter=-1, nu=0.01, random\_state=10, shrinking=True, tol=0.01,**

**verbose=False), 0.9, 0.7333333333333333, 0.5**

**使用上述最好结果：**

**场景1：召回率0.9**

**场景2：召回率0.73**

**场景3：召回率0.5**

**或者可以使用**

**[0.04, 1.0, 0.7142857142857143, 0.7142857142857143, 0.027909426013691417, 0.05152576288144072, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',**

**max\_iter=-1, nu=0.04, random\_state=10, shrinking=True, tol=0.01,**

**verbose=False), 0.8, 0.6, 0.8,**

**哈哈，重新调整了nu的值终于得到**

**[0.003, 1.0, 0.7857142857142857, 0.7857142857142857, 0.030015797788309637, 0.0560280140070035, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',**

**max\_iter=-1, nu=0.003, random\_state=10, shrinking=True, tol=0.01,**

**verbose=False), 0.9333333333333333, 0.7, 0.6, （最终Static结果可以选择这个，nu=0.001:1.000）**

**保存该结果！**

**可以根据需要，灵活使用nu=0.003和nu=0.01两种情况**

**上述结论的FPR除以了2000，未修正，但是Recall是正确的**

0:19:08

接下来周一需要考虑的是

1. 将上述实验改写为按月分析，即每月结束时，自动使用上述方法分析当月离职用户中的Risk用户，Recall指标为当月离职用户中的Insiders，而FPR指标依旧为离职中的非Insiders占所有用户的比例

发现了计算FPR与Risk\_Ratio时的错误，显示写入了总用户数

最终取信结果为

**[0.003, 1.0, 0.7857142857142857, 0.7857142857142857, 0.06129032258064516, 0.112, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',**

**max\_iter=-1, nu=0.003, random\_state=10, shrinking=True, tol=0.01,**

**verbose=False), 0.9333333333333333, 0.7, 0.6, array([ 1, -1, 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1,**

**1, 1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, 1, 1, 1, -1, 1,**

**-1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, 1, 1, 1, 1,**

**1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1, -1, -1, 1, 1,**

**-1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1,**

**1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1,**

**1, 1, -1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1,**

**-1, 1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1, 1, -1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 1, 1, -1, -1,**

1. **, dtype=int64)]**
2. 今天还需找Y老师修改论文，晚上早休息；

二〇一九年一月十五日星期二

下午3时4分

决定依据上述主观建模实验写作一篇文章投学报，以达到毕业标准。

之前的分析是关于Static的，即假设获取了一年半的所有数据然后静态分析，这次选择从每个月的数据中进行分析，并保存每个月的结果：

1. CERT4.2数据集中每个月份（2010-01:2011-04）中的选择OCSVM，整体与三类场景的Recall以及误报率，并将预测结果保存在KMeans+OCSVM目录中；

经过修正，CERT4.2中Insiders的分布从2010-06：2011-04

因此，我们着重分析下述月份的预测结果

2010-06：

[0.001, 1.0, 1.0, 1.0, **0.0030864197530864196**, 0.00513347022587269, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **'-1', '-2', 1.0**, array([ 1, -1, 1, 1, 1, -1, 1], dtype=int64)]

2010-07：

[0.019, 1.0, 1.0, 1.0, **0.0**, 0.0031023784901758012, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.019, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, '-2', 1.0**, array([ 1, -1, -1, -1, 1, -1, 1], dtype=int64)]

2010-08：

[0.019, 1.0, 1.0, 1.0, **0.0,** 0.0031023784901758012, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.019, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, '-2', 1.0**, array([ 1, -1, -1, -1, 1, -1, 1], dtype=int64)]

2010-09：

[0.009, 1.0, 1.0, 1.0, **0.00533049040511727**1, 0.013742071881606765, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.009, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, '-3'**, array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-10：

[0.002, 1.0, 0.8333333333333334, 0.8333333333333334, **0.004343105320304018**, 0.015005359056806002, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **0.8571428571428571, 1.0, 0.5**, array([ 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1],

dtype=int64)]

2010-11

[0.004, 1.0, 1.0, 1.0, **0.0022002200220022**, 0.010905125408942203, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.004, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, 1.0**, array([ 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-12

[0.002, 1.0, 1.0, 1.0, **0.006688963210702341**, 0.015469613259668509, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, 1.0**, array([ 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1],

dtype=int64)]

2011-01

[0.002, 1.0, 1.0, 1.0, **0.006779661016949152**, 0.010135135135135136, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, '-3**', array([ 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1], dtype=int64)]

2011-02：

[0.006, 1.0, 1.0, 1.0, **0.003444316877152698**, 0.010262257696693273, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, '-3'**, array([ 1, 1, 1, 1, 1, 1, 1, 1, -1, 1], dtype=int64)]

2011-03

[0.001, 1.0, 1.0, 1.0, **0.006952491309385863**, 0.011534025374855825, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, '-3'**, array([ 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1], dtype=int64)]

2011-04

[0.006, 1.0, 1.0, 1.0, **0.005875440658049354**, 0.011682242990654205, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **'-1', 1.0, 1.0**, array([ 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1], dtype=int64)]

接下来开始思考围绕该研究写作学报论文，一周完成。

二〇一九年一月十六日星期三

上午9时54分

上午准备开始整理写作小论文，突然发现CERT4.2 的CPB部分团队使用的是2个特征的方法，与整体不符合，故重新修改计算，重新运行一次吧！

重新修正了Team的CPB\_Mean/Median后，发现结果果然同之前的结果相差不大，对于整体形式而言：

[0.003, 1.0, 0.7857142857142857, **0.7857142857142857, 0.06129032258064516**, 0.112, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.003, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.9333333333333333, 0.7, 0.6,

如果继续按照月份进行：

补充，如果考虑半监督自动生成最好的Nu初步计划采用



事实证明，除了对于第三种场景有影响外，对于前两类攻击场景基本影响不大，可以使用前三月均值的连续均值确定后续最佳

2010-06：

[**0.001**, 1.0, 1.0, 1.0, 0.0030864197530864196, 0.00513347022587269, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), '-1', '-2', 1.0, array([ 1, -1, 1, 1, 1, -1, 1], dtype=int64)]

2010-07：

[**0.019**, 1.0, 1.0, 1.0, 0.0, 0.0031023784901758012, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.019, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, '-2', 1.0, array([ 1, -1, -1, -1, 1, -1, 1], dtype=int64)]

2010-08：

[**0.003**, 1.0, 0.9090909090909091, 0.9090909090909091, 0.002107481559536354, 0.0125, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.003, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.8333333333333334, 1.0, 1.0, array([ 1, 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1],

dtype=int64)]

2010-09

均值Nu= 0.0077

[0.007, 1.0, 0.875, 0.875, 0.005330490405117271, 0.012684989429175475, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.007, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 0.8571428571428571, '-3', array

[0.009, 1.0, 1.0, 1.0, 0.005330490405117271, 0.013742071881606765, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.009, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-10

如果nu=0.009(三次均值)

[0.009, 1.0, 0.75, 0.75, 0.004343105320304018, 0.013933547695605574, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.009, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.8571428571428571, 1.0, 0.0, array([ 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, 1, -1, 1, 1],

dtype=int64)]

[0.002, 1.0, 0.8333333333333334, 0.8333333333333334, 0.004343105320304018, 0.015005359056806002, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.8571428571428571, 1.0, 0.5, array([ 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1],

dtype=int64)]

2010-11

三次均值nu=0.006

[0.006, 1.0, 0.875, 0.875, 0.0044004400440044, 0.011995637949836423, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 0.0, array([ 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1], dtype=int64)]

[0.004, 1.0, 1.0, 1.0, 0.0022002200220022, 0.010905125408942203, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.004, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([ 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-12

三次均值Nu=0.007

[0.007, 1.0, 1.0, 1.0, 0.004459308807134894, 0.013259668508287293, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.007, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([-1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1],

dtype=int64)]

[0.002, 1.0, 1.0, 1.0, 0.006688963210702341, 0.015469613259668509, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([ 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1],

dtype=int64)]

2011-01

三次均值nu=0.007

[0.007, 1.0, 0.6666666666666666, **0.6666666666666666, 0.00903954802259887**, 0.01126126126126126, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.007, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 0.5, '-3', array([ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1], dtype=int64)]

[0.002, 1.0, 1.0, **1.0, 0.006779661016949152**, 0.010135135135135136, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1], dtype=int64)]

2011-02

三次均值nu=0.006同最好结果一样

[0.006, 1.0, 1.0, 1.0, 0.003444316877152698, 0.010262257696693273, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, 1, 1, 1, 1, -1, 1], dtype=int64)]

2011-03

三次均值Nu=0.006

[0.006, 1.0, 1.0, **1.0, 0.008111239860950173**, 0.012687427912341407, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

[0.001, 1.0, 1.0, **1.0, 0.006952491309385863**, 0.011534025374855825, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1], dtype=int64)]

2011-04

三次均值nu=0.006

[0.006, 1.0, 1.0, 1.0, 0.005875440658049354, 0.011682242990654205, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.006, random\_state=10, shrinking=True, tol=0.01,

verbose=False), '-1', 1.0, 1.0, array([ 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1], dtype=int64)]

结果验证完毕，同最终结果影响不大。

二〇一九年一月十八日星期五

下午6时17分

在写论文时发现，如果采用动态CPB特征如何呢？即每月的CPB特征重新计算，去掉上月离职的用户，仅考虑当月在职用户组成CPB即可。

先重新生成2010-01:2011-04的26JS特征

突然发现原来使用的是2009-12的Team数据，因而需要同步更新cert/team/dpt数据才可以生成正确的每月份的工作环境CPB。等写完论文再来处理，目前先默认使用按月份的的CPBs

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

二〇一九年一月十九日星期六 最新的Static测试结果！！！

下午9时1分

晚上去给昂拿电源，回来突然想到，CERT4.2数据集中1000个用户中，去掉离职的154个用户，再加上聚类筛选得到的剩余类别，合起来也就100多用户，即便都用来做Static分析，又能有多少误报率呢？

CERT4.2再分类实验

KMeans所有OnJob用户，得到

0 732

1 113

按照既往的规则，仅测试离职用户，则最好结果有：

[0.0028, 1.0, **0.8714285714285714**, 0.8714285714285714, **0.08279569892473118**, 0.138, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0028, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **0.9333333333333333, 0.8666666666666667, 0.7**, 】

整体召回率0.87, 单类召回率分别为0.93-0.86-0.7，误报率8%，可以说结果非常理想；

**自动筛选使用732个用户训练OCSVM，而将113+154=267个用户作为测试集，则有**

PCA=1时：

[0.0029, 1.0, 0.8615384615384616, 0.8615384615384616, **0.2053763440860215**, 0.247, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0029, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.9333333333333333, 0.7333333333333333, 0.6, array]

[0.0034, 1.0, 0.8461538461538461, 0.8461538461538461, **0.17849462365591398**, 0.221, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0034, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 0.9333333333333333, 0.7, 0.6,

其中，nu=0.0034的结果更为合理，虽然Recall下降了0.02，但是FPR下降了0.03。且此时的最好Recall达到了0.86

分析：

使用其他PCA>1的结果都不如PCA=1，因此还是使用PCA=1作为结果；

但是从实验数据来看，267-70=197个非Insiders离职用户中，有158个用户判断为了“Insider”，有近40个用户仅仅是离职而非缺勤。

实际可以提高到0.5的197即约100个用户判断为“Insiders”，而胜于的97个用户为离职而非低CPB；

进一步分析FP中有缺勤比例的行为的用户，说明其已经属于中高CPB过渡阶段的用户。

要点是说明两点：

1. 虽然仅检测离职用户，但是离职用户中并不都是positive，还有约0.4比例的N用户是Negtive，即因为种种其他原因离职，而非低CPB，说明并非简单取巧分析LeaveUsers;
2. 即便在识别为P的N用户中，也有一定比例是确实表现出高于均值的缺勤，因此应当说模型是有效的；

针对逐月模式，nu精确到10000

2011-04

[0.0019, 1.0, 1.0, 1.0, 0.007050528789659225, 0.012850467289719626, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0019, random\_state=10, shrinking=True, tol=0.01,

verbose=False), '-1', 1.0, 1.0, array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2011-04 Mean:0.0048

*[0.0048, 1.0, 0.6, 0.6, 0.005875440658049354, 0.009345794392523364, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0048, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), '-1', 0.5, 1.0, array([ 1, 1, -1, 1, 1, -1, -1, 1, 1, 1, 1], dtype=int64)]*

2011-03

[0.001, 1.0, 1.0, 1.0, 0.006952491309385863, 0.011534025374855825, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1], dtype=int64)]

2011-03 Mean:0.0048

*[0.0048, 1.0, 0.75, 0.75, 0.006952491309385863, 0.010380622837370242, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0048, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 0.5, 1.0, '-3', array([ 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1], dtype=int64)]*

2011-02

[0.0001, 1.0, 1.0, 1.0, 0.001148105625717566, 0.00798175598631699, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([ 1, -1, 1, -1, 1, 1, 1, 1, -1, 1], dtype=int64)]

2011-02 Mean:0.0048

*[0.0048, 1.0, 0.8333333333333334, 0.8333333333333334, 0.003444316877152698, 0.009122006841505131, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0048, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 1.0, 0.5, '-3', array([ 1, 1, -1, 1, 1, 1, 1, 1, -1, 1], dtype=int64)]*

2011-01

*[0.002, 1.0, 1.0, 1.0, 0.006779661016949152, 0.010135135135135136, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1], dtype=int64)]*

*2011-01 Mean:0.0049*

*[0.0049, 1.0, 1.0, 1.0, 0.007909604519774011, 0.01126126126126126, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0049, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 1.0, 1.0, '-3', array([ 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1], dtype=int64)]*

2010-12

[0.002, 1.0, 1.0, 1.0, 0.006688963210702341, 0.015469613259668509, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.002, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([ 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1],

dtype=int64)]

2010-12 Mean: 0.0047

*[0.0047, 1.0, 0.875, 0.875, 0.006688963210702341, 0.014364640883977901, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0047, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 1.0, 0.75, 1.0, array([ 1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1, 1, 1, 1, 1, 1, 1],*

*dtype=int64)]*

2010-11

[0.0029, 1.0, 1.0, 1.0, 0.0033003300330033004, 0.011995637949836423, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0029, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([ 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-11 Mean: 0.0048

*[0.0048, 1.0, 0.75, 0.75, 0.0033003300330033004, 0.009814612868047983, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0048, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False),* ***0.8, 0.5, 1.0****, array([ 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, -1, 1], dtype=int64)] nu采用三均值方式推算不合适，Recall降低太快*

2010-10

[0.0005, 1.0, 0.9166666666666666, 0.9166666666666666, 0.004343105320304018, 0.01607717041800643, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0005, random\_state=10, shrinking=True, tol=0.01,

verbose=False), **1.0, 1.0, 0.5**, array([ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1],

dtype=int64)]

2010-10 Mean:0.0054

*[0.0054, 1.0, 0.6666666666666666, 0.6666666666666666, 0.004343105320304018, 0.012861736334405145, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0054, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 0.7142857142857143, 0.6666666666666666, 0.5, array([ 1, -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1],*

*dtype=int64)]*

*采用新公式：*

*Nu\_0 = Nu\_Mean \**

2010-09

[0.0001, 1.0, 1.0, 1.0, 0.005330490405117271, 0.013742071881606765, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0001, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, '-3', array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-09 Mean: 0.0041

*[0.0041, 1.0, 1.0, 1.0, 0.005330490405117271, 0.013742071881606765, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',*

*max\_iter=-1, nu=0.0041, random\_state=10, shrinking=True, tol=0.01,*

*verbose=False), 1.0, 1.0, '-3', array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]*

2010-08

[0.0049, 1.0, 1.0, 1.0, 0.003161222339304531, 0.014583333333333334, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0049, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, 1.0, 1.0, array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)]

2010-07

[0.0072, 1.0, 1.0, 1.0, 0.0031120331950207467, 0.0062047569803516025, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0072, random\_state=10, shrinking=True, tol=0.01,

verbose=False), 1.0, '-2', 1.0, array([ 1, 1, 1, -1, 1, 1, 1], dtype=int64)]

2010-06

[0.0003, 1.0, 1.0, 1.0, 0.00411522633744856, 0.006160164271047228, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.0003, random\_state=10, shrinking=True, tol=0.01,

verbose=False), '-1', '-2', 1.0, array([ 1, -1, 1, 1, 1, 1, 1], dtype=int64)]

————————————————————————————————————————

采用三月逐月均值nu的方式得到的结果与最好结果有一定差距，可以强行解释个别用户离职逐渐改变原有用户分布，因此可以采用均值的形式缓慢实现改变

使用随机森林分析了21维度特征的重要性，有：

0 0.039644621895032935

1 0.029184738371074423

2 0.02791343500017892

3 0.0346967729834734

4 0.03313966386169054

5 0.022544385066981802

6 0.020322145776432585

7 0.02069748584791118

8 0.028560593959520004

9 0.03184875415598664

10 0.0

11 0.0

12 0.061215713233698885

13 0.060821684696370906

14 0.0533380006445595

15 0.058251435829502714

16 0.0289515266480221

17 0.041668494632307804

18 0.04382009367379904

19 0.03985608974081651

20 0.3235243639826405