2018年12月19日星期三

上午10时7分

从9月份开始，走了许多弯路，这次利用18年最后的两周，争取有所突破，结束这部分工作；否则，自己明年的毕业将十分被动。

按照计划，19年1月份将主要进行主客观关联研究，1月底要找Y老师汇报论文大提纲；2月份完成毕业论文初稿，3月初准备预答辩；关系到自己的毕业大事情，决不能有半点马虎与分心。

今天上午自己重新整理了实验的思路，基于以往的实验研究经历，总结起码有以下两点经验：

1. 基于问题域提出理论假设，接下来需要在数据实验中验证；在编写实验程序验证前，首先要完成必要性检验，即验证该假设是否具备成立的必要条件，比如可以借助随机森林方法分析上帝视角的特征重要性，看是否具有区分度；
2. 编写完备的大型实验程序前（面向对象），首先针对特定用户、特定月份进行实验，验证分析实验效果，以确定是否继续编写完毕实验或者是否对模型进行改进；

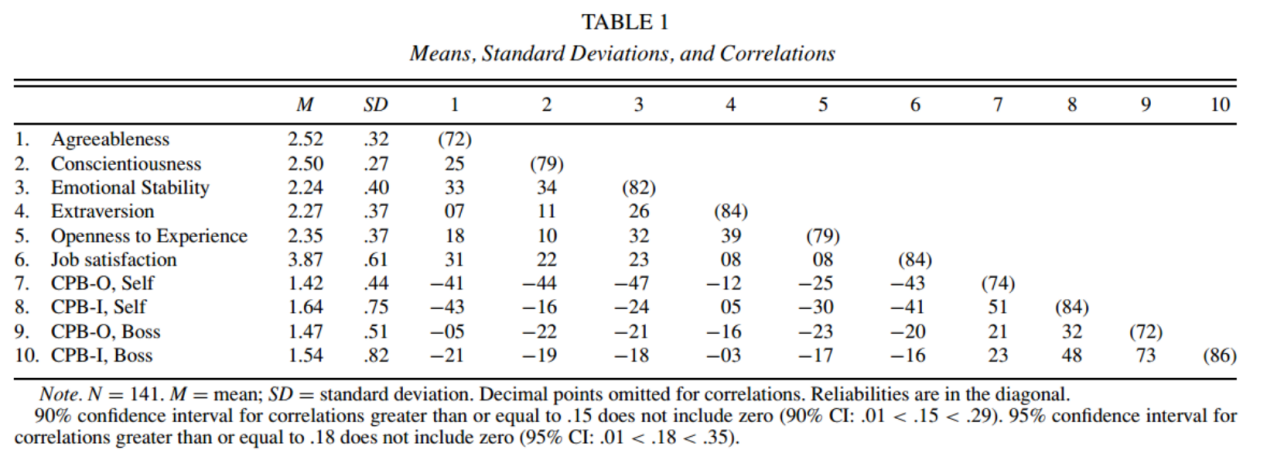
新的模型假设

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。

按照月份建立用户的上述攻击倾向特征（Attack Tendency Feats, ATF），训练的**当月用户数据**分为在职[-1]与离职[+1]，从中对[-1]进行五折交叉，每次选取80%的用户用作训练OCSVM，而20%的用户与当月[+1]用户作为测试集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）。上述结果对于同一个月数据可以得到五个局部最优OCSVM，然后分别对下一个月的用户ATF数据进行预测，输出下一个月用户的预测标签[Label\_1, Label\_2, Label..Lable\_5]，依据投票决定是[-1]还是[+1]，并将预测结果与GroundTruth比对得到Recall与FPR；

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) |
| 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, |

好了，接下来首先生成新的CERT5.2的Team\_Affects特征：

经过下午1点半回来奋战1个半钟头，顺利完成原始CPB特征与新的LC特征的修改，并进入到实验提取阶段，先来验证下新的CPB特征：

原始CPB：

user\_id,O\_Score,C\_Score,E\_Score,A\_Score,N\_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,No-JobState-in-Team,Dpt-CPB-I-mean,Dpt\_CPB-O-mean,Dpt-Less-A-mean,Dpt-Less-AC-mean,Dpt-less-C-mean,Dpt-High-N-mean,Dpt\_CPB-I-median,Dpt\_CPB-O-median,No-JobState-in-Dpt,Job State,Leader\_CPB-I,Leader\_CPB-O

MMK1532,17.0,17.0,16.0,22.0,28.0,-14.9717333333,-19.38576,16,7,12,16,-16.94,-18.686,0,-15.617477707,-19.3226853503,89,39,59,90,-17.424,-17.7936,11,18,-11.132,-12.7548,

新的CPB：

MMK1532,17.0,17.0,16.0,22.0,28.0,-13.761999999999999,-24.254666666666665,16,7,12,16,-14.555000000000001,-25.135,0,-13.22647435897436,-23.644999999999996,88,38,58,90,-13.729999999999999,-23.83,1,18,-4.79,-10.23,

经过分析，新的CPB中Team特征与以前一致；Dpt特征中用户个数减少了一个，一方面我们不以Dpt作为考虑，另一方面以新的为准。

其中唯一没有考虑的用户为：

['Anjolie Evangeline Houston', 'AEH0001', 'Anjolie.Evangeline.Houston@dtaa.com', 'ChiefExecutiveOfficer'] 作为CEO，不需要考虑

旧CPB计算时对于部门内部自身用户匹配代码错误写成了dpt\_nm，导致多加了用户自身。

**因此，使用新数据。**

同时，按照LC均值的LC特征也在运行中，之后需要将两个特征组合起来形成新的用户特征ATF，然后在此基础上进行运算。

**LCF计算的依据是截止到用户离职前一个月的LC情况。**

下午3时48分

今天后续的研究任务：

* 结合CPBs与LCF形成完整的ATF；
* 查看分析2010-08/09/10三个月份的ATF特征直接scale后，使用RandomForest方法检验特征重要性（在职[-1], 离职[+1]）
* 可能的话实现五折交叉验证（顺序使用80%训练，20%验证）的多OCSVM投票方法判断类别，并选取最好的结果作为ATF量化指标；g



2018年12月22日星期六

下午5时54分

晚上去保养摩托车来回路上，突然奇想想到了现在实验建模用户攻击倾向的方法。

1. 合并组成新的用户ATF；
2. 有两种思路可以选择：

* 基于五折交叉验证的思想，全面考虑各个数据部分在训练与验证OCSVM中的作用，首先将训练[-1]数据随机分为五个部分（尽量均匀），然后依顺序使用其中一部分+[+1]类作为验证集，剩余4部分[-1]作为训练集训练OCSVM，然后使用对应的验证集遍历选择最优的参数，得到相应的子OCSVM；最终使用下个月数据进行预测，5个字OCSVM投票多者判定；
* 基于用户攻击倾向分层次的子OCSVM，先用自动[2,10]KMeans得到相似攻击倾向用户的群簇，然后依据不同的群簇分别建立子OCSVM，然后使用K个子OCSVM同时预测下个月数据，若有其中一个判断为[+1]，则该用户判定为下个月[+1]；训练过程中为依据上个月数据，群簇中随机选择1/5+[+1]作为验证集，然后选择得到最优的OCSVM参数；[遍历参数时可以从Recall=1.0开始变化，得到Recall<1.0的边界作为最优参数][优先确保Recall或者使用(Recall, FPR)的综合考量函数]

下午9时26分

晚上。

主要任务是由前期提取的用户CPB与LC特征合成最后的ATF。

验证：

首先是

CPBs：

user\_id,O\_Score,C\_Score,E\_Score,A\_Score,N\_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,No-JobState-in-Team,Dpt-CPB-I-mean,Dpt\_CPB-O-mean,Dpt-Less-A-mean,Dpt-Less-AC-mean,Dpt-less-C-mean,Dpt-High-N-mean,Dpt\_CPB-I-median,Dpt\_CPB-O-median,No-JobState-in-Dpt,Job State,Leader\_CPB-I,Leader\_CPB-O

MMK1532,17.0,17.0,16.0,22.0,28.0,-13.761999999999999,-24.254666666666665,16,7,12,16,-14.555000000000001,-25.135,0,-13.22647435897436,-23.644999999999996,88,38,58,90,-13.729999999999999,-23.83,1,18,-4.79,-10.23,

其次是LCE

user\_id, o, c, e, a, n, cpb-i, cpb-o, dis\_ocean, avg\_dis\_ocean, dis\_os, avg\_dis\_os, cnt\_late\_days, cnt\_early\_days, month\_work\_days, email\_ratio, cnt\_send/recv, cnt\_s/r\_size, cnt\_s/r\_attach, cnt\_s/r\_days, cnt\_email\_days

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.274789435642,0.274789435642,0.241624510319,0.241624510319,20.0,15.0,40.0,1.0,1.0,0.0,38275.0,0.0,0.0,0.0,1.0,0.0,1.0,

我们计算得到的新的ATF为：

MMK1532,17.0,17.0,16.0,22.0,28.0,-6.56,-9.51,6.76,-13.761999999999999,-24.254666666666665,16,7,12,16,-14.555000000000001,-25.135,-4.79,-10.23,0.274789435642,0.274789435642,0.241624510319,0.241624510319,20.0,15.0,40.0,1.0,1.0,0.0,38275.0,0.0,0.0,0.0,1.0,0.0,1.0,

MMK1532,17.0,17.0,16.0,22.0,28.0,-6.56,-9.51,6.76,-13.761999999999999,-24.254666666666665,16,7,12,16,-14.555000000000001,-25.135,-4.79,-10.23,0.274789435642,0.274789435642,0.241624510319,0.241624510319,1.0,1.0,0.0,38275.0,0.0,0.0,0.0,1.0,0.0,1.0,20.0,15.0,40.0,

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.0,0.0,0.0,0.0,10.0,6.0,20.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,

*MMK1532,17.0,17.0,16.0,22.0,28.0,-6.56,-9.51,6.76,-13.761999999999999,-24.254666666666665,16,7,12,16,-14.555000000000001,-25.135,-4.79,-10.23,*0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,10.0,6.0,20.0,

js\_score = 0.08 \* float(line\_lst[1]) + 0.22 \* float(line\_lst[2]) + 0.08 \* float(line\_lst[2]) + 0.31 \* float(line\_lst[4]) - 0.23 \* float(line\_lst[5])

验证通过！

今天最后为上述特征添加说明标签即可！

上述工作位按成！

二〇一八年十二月二十四日星期一

上午10时10分

今天一早重新考虑了思路，并将新的思路整理在下面，然后开始基于提取的ATF特征进行实验。

* 主要思路：整体的OCSVM刻画并不准确，需要采用较小的局部球面代替，即多OCSVM

1. 基于五折交叉验证的多OCSVM检测：基于N月预测N+1月，则将N月中所有在职[-1]类用户随机分为均等五个部分，然后依次采用其中4个部分作为训练集，剩余1个部分结合N月的离职用户[+1]类作为验证集，以此遍历OCSVM的参数空间得到最优解；最优解的判定标准为公式，在上述公式中计算最终的分数，取分数最高者；当μ=1时，选择FPR最小的；当μ=0时，选择Recall最高的；最终得到的五个子OCSVM对于N+1月的用户ATF进行分析预测，输出结果与N+1月的GroundTruth对比，得到第N+1月的预测结果（Recall, FPR, HighRiskRatio）
2. 基于相近攻击倾向聚类的思想，对于N月的[-1]类用户进行[2,10]的轮廓系数KMeans聚类，对于每一个所得的群簇而言，随机选取80%作为训练集，20%+N月的[+1]类用户作为验证，训练该群簇对应的子OCSVM；最终得到N月的K个子OCSVM，针对N+1月的ATF用户特征，初期假设若命中其中一个群簇的OCSVM判定为[+1]，则判定该用户为[+1]，与N+1月的GroundTruth比较，输出N+1月预测结果

在继续实验前，首先使用RandomForest对于特征重要性进行分析，依据N月的ATF与对应GT运行RandomForest：2010-08月份的ATF与对应的GT数据：

0 0.01781494418700184

1 0.016573088635490382

2 0.017589945081491164

3 0.015103697337465388

4 0.014630546777728626

5 0.029660107627069953

6 0.021602759627784268

7 0.020982485245475253

8 0.015170140203745089

9 0.01726868430099518

10 0.012396651314275085

11 0.014714870762005023

12 0.010673549050020908

13 0.009590237942022758

14 0.015580165800832558

15 0.0148538731865902

16 0.018809482729850134

17 0.014917092820170367

18 0.02357486591709682

19 0.018519108340318013

20 0.009922924854033266

21 0.02528746439179336

22 0.012976142133437904

23 0.014663787599310378

24 0.014995783303174786

25 0.012670073237938602

26 0.012430352928379806

27 0.010774340676526698

28 0.014625389064035564

29 0.014224984964922499

30 0.01574639552861443

31 0.017376694435349595

32 0.013233204886256967

33 0.011641012593859001

**34 0.4594051525149367**

如果采用MinMax后：结果一样

如果先采用自动PCA然后采用MinMax：

0 0.019856919840874333

1 0.014128367585787052

2 0.023773071799295963

3 0.043091909993568454

4 0.03648815655145728

5 **0.2166858191516953**

6 0.02148872739250789

7 0.015712383412839072

8 0.02771200793373519

9 0.02667375865772601

10 0.025394531006235583

11 0.02773853509930786

12 0.025832319516907093

13 0.022641930265949205

14 0.02701336847602065

15 0.021828921535488342

16 0.02628761673369687

17 0.027496023849168164

18 0.01671271851669812

19 0.013569363648701824

20 0.03460833928195251

21 0.023354881745060004

22 0.016643962272956236

23 0.020401352088739555

24 0.029103316132259197

25 0.01390912993706016

26 0.014932370572791981

27 0.015175339100674534

28 0.03552602883096532

29 0.023435664522997234

30 0.02112469326868398

31 0.02608950608520348

32 0.014185095791543029

33 0.013867425495809303

34 0.017516443905643487

如果人为设定PCA=17

0 0.03419659571463046

1 0.022759246746475448

2 0.040749148363361035

3 0.07120173585965466

4 0.05424916899886134

5 **0.3351108090116051**

6 0.037843734817181146

7 0.02660177488277424

8 0.04487085548499092

9 0.0413896981783078

10 0.03808374988790266

11 0.04290819586615704

12 0.04372839927027642

13 0.037559857594063634

14 0.04294827829060923

15 0.03695846266290073

16 0.04884028837024891

对应的每个部分的重要性都提升了，可以感性理解为：更多的特征分散了重要性

如果先自动pca然后scale（OCSVM的处理流程）

0 0.019856919840874333

1 0.014128367585787052

2 0.023773070304018526

3 0.043095429005914135

4 0.03648815656621065

5 **0.21668574395485948**

6 0.02148872692548495

7 0.015707211174245433

8 0.02771200793373519

9 0.026673832385975663

10 0.025394530210252772

11 0.02773156169165851

12 0.02583426554347932

13 0.022641930265949205

14 0.02701336847602065

15 0.021828917561901095

16 0.026287613196823975

17 0.027496022383672488

18 0.01671271851669812

19 0.013569363648701824

20 0.03460833928195251

21 0.023354881745060004

22 0.016643961975519896

23 0.020401349835572194

24 0.029103316132259197

25 0.01390912993706016

26 0.014929542795014206

27 0.015175339100674534

28 0.03552602883096532

29 0.023435664522997234

30 0.021131900164912107

31 0.02608950608520348

32 0.014195330090456712

33 0.013859508424445874

34 0.017516443905643487

如果先做scale，然后再PCA，结果维度没有改变，

0 0.029271391874431806

1 0.031749287860081216

2 0.03789560553084402

3 0.0513941885432056

4 0.043043822691812575

5 0.035355340293826275

6 0.023014466979458567

7 0.02852678042587996

8 0.024768263821610664

9 0.024954752399392915

10 0.0485723830711371

11 0.023023734525841454

12 0.10936742280735387

13 0.03762370258656122

14 0.02069729065981083

15 0.01994951400887

16 0.016639528687606694

17 0.018548217976454708

18 0.039960667696867555

19 0.026028726055247257

20 0.020227789595246213

21 0.029032065282161703

22 0.02137112933612393

23 0.035799603200437

24 0.024903297095341403

25 0.02604418045983718

26 0.022872018572822485

27 0.04398652111142298

28 0.02437871201468143

29 0.018789552430381953

30 0.021422720290005057

31 0.02078732211524457

32 0.0 33 0.0 34 0.0

下午2时16分

通过对2010-08月的ATF与GT数据分析，发现如果直接使用随机森林计算特征重要性，则35个特征基本都不明显；当先做了自动PCA后再使用MinMax/Scale后，发现CPB-I表现出异常的特点，其特征重要性大于CPB-O；

单纯使用A与C计算CPB的数据公式尾标为0.11

而使用OCEAN全部计算CPB/JS的数据公式为0.1

如使用OCEAN的26JS特征文件名为：CERT5.2-200912-New-26JS-0.9.csv

而单独使用A与C计算的26JS数据文件名为：CERT5.2-200912-New-26JS-0.9-0.11.csv

从已有效果来看，单独使用A与C的CPB区分度不如同时使用OCEAN

接下来由于需要使用OCSVM，我们采用PCA+scale的组合方式处理数据。

下午10时9分

需要训练OCSVM的基本数据已经完成，之前的调参形式为：

clf = OneClassSVM(kernel='rbf', tol=0.01, nu=0.35, gamma='auto')

Nu:训练错误的上界，(0,1]，默认0.5

Tol: 停止的精度

print 'OCSVM开始训练...\n'

clf = OneClassSVM(kernel='rbf', tol=0.01, nu=0.35, gamma='auto')

Train\_array = np.array(Train\_lst)

Test\_array = np.array(Test\_lst)

clf.fit(Train\_array)

pred = clf.predict(Test\_array)

print 'ACM2278 is ', clf.predict(Users\_26JS\_pca\_nor[2839]), '\t', clf.decision\_function(Users\_26JS\_pca\_nor[2839]), '\n'

print 'CMP2946 is ', clf.predict(Users\_26JS\_pca\_nor[2329]), '\n', clf.decision\_function(Users\_26JS\_pca\_nor[2329]), '\n'

print 'PLJ1771 is ', clf.predict(Users\_26JS\_pca\_nor[1281]), '\t', clf.decision\_function(Users\_26JS\_pca\_nor[1281]), '\n'

print 'CDE1846 is ', clf.predict(Users\_26JS\_pca\_nor[654]), '\n', clf.decision\_function(Users\_26JS\_pca\_nor[654]), '\n'

print 'MBG3183 is ', clf.predict(Users\_26JS\_pca\_nor[1493]), '\n', clf.decision\_function(Users\_26JS\_pca\_nor[1493]), '\n'

趁热打铁

写出了2010-08月的五个OCSVM分类器，其结果如下;

2010-08 五个最优OCSVM依次为：

召回率尚可的情况下，误报率实在太高了！

当然，我们还需要继续使用上述五个OCSVM clfs来预测N+1月份的数据，然后得出最终结论；

No: 0 [0.05, 0.9, 0.6482630847614637, **0.8235294117647058**, **0.9291338582677166**, 0.9246231155778895, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 1 [0.05, 0.9, 0.6490504863362668, **0.8235294117647058, 0.9212598425196851**, 0.9170854271356784, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 2 [0.05, 0.9, 0.5926972363748649, **0.7647058823529411, 0.9553805774278216**, 0.9472361809045227, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 3 [0.05, 0.9, 0.6503628222942721, **0.8235294117647058, 0.9081364829396326**, 0.9045226130653267, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 4 [0.05, 0.9, 0.6520855614973262, **0.8235294117647058, 0.8909090909090909**, 0.8880597014925373, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

二〇一八年十二月二十七日星期四

下午3时39分

在时隔多日之后，终于重新开始上述实验的分析。

之前实验发现了召回率尚可的情况下，误报率较高，于是以一个单折OCSVM检查nu参数的结果：

Bug for:

[0.05, 0.0, -0.8976377952755905, **0.8235294117647058**, **0.8976377952755905,** 0.8944723618090452, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.1, 0.0, -0.8740157480314961, **0.7058823529411765, 0.8740157480314961,** 0.8668341708542714, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.1, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.15, 0.0, -0.8057742782152231, **0.7058823529411765, 0.8057742782152231**, 0.8015075376884422, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.15, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.2, 0.0, -0.7664041994750657, **0.6470588235294118, 0.7664041994750657**, 0.7613065326633166, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.2, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.25, 0.0, -0.7322834645669292, **0.6470588235294118, 0.7322834645669292**, 0.7286432160804021, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.25, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.3, 0.0, -0.7034120734908137, **0.5882352941176471, 0.7034120734908137**, 0.6984924623115578, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.3, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.35, 0.0, -0.6614173228346457, **0.5882352941176471, 0.6614173228346457**, 0.6582914572864321, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.35, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.4, 0.0, -0.6272965879265092, 0.5882352941176471, 0.6272965879265092, 0.6256281407035176, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.45, 0.0, -0.5800524934383202, 0.5294117647058824, 0.5800524934383202, 0.5778894472361809, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.45, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.5, 0.0, -0.5328083989501312, 0.5294117647058824, 0.5328083989501312, 0.5326633165829145, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.5, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.55, 0.0, -0.4776902887139108, 0.4117647058823529, 0.4776902887139108, 0.4748743718592965, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.55, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.6, 0.0, -0.4199475065616798, 0.4117647058823529, 0.4199475065616798, 0.41959798994974873, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.6, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.65, 0.0, -0.3700787401574803, 0.4117647058823529, 0.3700787401574803, 0.37185929648241206, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.65, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.7, 0.0, -0.31496062992125984, 0.35294117647058826, 0.31496062992125984, 0.3165829145728643, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.7, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.75, 0.0, -0.24146981627296588, 0.35294117647058826, 0.24146981627296588, 0.24623115577889448, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.75, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.8, 0.0, -0.2178477690288714, 0.17647058823529413, 0.2178477690288714, 0.21608040201005024, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.8, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.85, 0.0, -0.1732283464566929, **0.17647058823529413, 0.1732283464566929**, 0.17336683417085427, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.85, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.9, 0.0, -0.13648293963254593, **0.11764705882352941, 0.13648293963254593**, 0.135678391959799, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.9, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

[0.95, 0.0, -0.07086614173228346, 0.0, 0.07086614173228346, 0.0678391959798995, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.95, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

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

max\_iter=-1, nu=1.0, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

由上述实验可以看出，随着nu的逐渐变化增大，球面逐渐扩张，导致OCSVM的Recall逐渐降低，同时FPR也逐渐降低，在nu=0.95时recall=0，而FPR=7%

上述实验采用的默认PCA，使用了全部35个维度特征，那么如果修改PCA的维度呢？

PCA=17

[0.95, 0.0, -0.05511811023622047, **0.058823529411764705, 0.05511811023622047**, 0.05527638190954774, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.95, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

PCA=8

[0.95, 0.0, -0.06036745406824147, 0.0, **0.06036745406824147, 0.05778894472361809**, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.95, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

PCA=5

[0.95, 0.0, -0.08398950131233596, 0.058823529411764705, 0.08398950131233596, 0.0829145728643216, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.95, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

PCA=3

[0.95, 0.0, -0.06561679790026247, 0.11764705882352941, 0.06561679790026247, 0.0678391959798995, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.95, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

从结果上看，特征与结果的相关性比较差，那么是否因为CPB的计算关系呢？

在继续深入研究前，我们先来看看如果单纯使用A与C计算CPB：

No: 0 [0.1, 0.8, 0.4348618187432454, 0.7647058823529411, 0.884514435695538, 0.8793969849246231, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.1, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 1 [0.05, 0.8, 0.4264628686120118, 0.7647058823529411, 0.926509186351706, 0.9195979899497487, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 2 [0.05, 0.8, 0.427512737378416, 0.7647058823529411, 0.9212598425196851, 0.914572864321608, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 3 [0.05, 0.8, 0.4269878029952139, 0.7647058823529411, 0.9238845144356955, 0.9170854271356784, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.05, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

No: 4 [0.1, 0.8, 0.43150496562261276, 0.7647058823529411, 0.9012987012987013, 0.8955223880597015, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.1, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

最好的FPR

0.0292110545005 [0.9, 0.8, **0.029211054500540377, 0.058823529411764705**, 0.08923884514435695, 0.08793969849246232, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.9, random\_state=None, shrinking=True, tol=0.01,

verbose=False)]

上述结果中虽然FPR只有5%，但是召回率却低的只有2%，该A/C决定的CPB中，连高召回率都达不到，最多只有40%左右的召回率。

二〇一八年十二月二十八日

上午10时3分

由于之前的模型中检测的标签打在了离职[+1]与在职[-1]上，结果最终的recall十分低；那么是否可以仅针对Insiders来进行检测呢？这里，我们将与满意度看似有关的三类场景纳入考虑（该三类场景的用户均离职）

需要在类初始化时添加一个Insiders的索引列表，而后后续五折分类时也需要对是否是Insiders进行考量（Insiders\_1, Insiders\_2, Insiders\_3, Insiders\_4）

做一个简单的假设，即除了insiders中的用户外，其余离职用户均假设为解雇用户，那么，可以试着对已有的离职用户进行区分，此时新的特征中都添加了’laidoff’关键词，而带有’null’关键词的文件则是无效错误文件，不应被使用。

二〇一八年十二月二十八日星期五

下午2时6分

在继续试验前，我们先来明确下目前为止0.9实验的重要结果说明。

之前实验时使用了元素索引作为样本分划的依据，便于操作，但是容易混淆中间的处理过程。

对于我们的ATF特征，使用的全OCEAN五元数据的CPB计算方式，得到的：

CPB特征：CERT5.2-200912-New-26JS-0.9.csv

使用单纯的A、C计算的CPB特征：CERT5.2-200912-New-26JS-0.9-0.11.csv

每个月份下，

将用户截止到该月的Leave Contacts通讯合并到一起的，LC特征，

考虑所有离职用户（含Insiders联系）：CERT5.2\_Month\_AvgLC\_JS\_Feats\_v01.csv

此时计算CPB默认使用的单纯A/C模式，但是无关紧要，因为后期合并ATF时对CPB进行了重新计算：CERT5.2\_ATF\_0.1.csv（我们实际分析的用户数据）

如果所有CPB都是用A/C模式：

~~CERT5.2\_ATF\_0.11.csv（实际检测效果不如0.1，已经弃用）~~

仅考虑解雇用户（假设Insiders之外的离职用户均是解雇）：

CERT5.2\_Month\_AvgLC\_LaidOff\_JS\_Feats\_v01.csv与

CERT5.2\_LaidOff\_ATF\_0.1.csv

未来的实验将围绕CERT5.2\_ATF\_0.1.csv与CERT5.2\_LaidOff\_ATF\_0.1.csv进行；

对应的2011-05\_CERT5.2\_Users\_GroundTruth.csv\_v01.csv中将当月在职人员标记为[-1]，离职人员标记为[+1]，其中未区分Insiders与解雇用户；

接下来，可以考虑使用纯碎的在职人员建立OCSVM，或者使用非Insiders建立OCSVM，取决于两类Labels方式。

在进行五折OCSVM实验分隔时，索引关系为：

ATF\_index<-->GroundTruth-->Train\_Index/Validate\_Index

摸清了整体思路，开始就2010-08月份的离职结果进行分析验证：

2010-08 [+1]类用户索引为: [110, 163, 176, 187, 501, 745, 748, 795, 829, 1000, 1006, 1071, 1102, 1235, 1319, 1477, 1685]

1. 使用2010-08月份所有用户ATF进行分隔训练，以8月份离职用户与部分在职用户作为验证集，五个OCSVM的结果为：

0：

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

MMR1458,True Positive

PBC0077,True Positive

JIB1258,True Positive,Recall

SCO1719,True Positive,Recall

VRP0267,True Positive

ZJN1492,True Positive,Recall

SAF1942,True Positive,Recall

PTV0067,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

DTB0722,True Positive,Recall

CEW1960,True Positive,Recall

ZHB1104,True Positive,Recall

EPG1196,True Positive,Recall

PTM1432,True Positive,Recall

1：

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

MMR1458,True Positive

PBC0077,True Positive

JIB1258,True Positive,Recall

SCO1719,True Positive,Recall

VRP0267,True Positive

ZJN1492,True Positive,Recall

SAF1942,True Positive,Recall

PTV0067,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

DTB0722,True Positive,Recall

CEW1960,True Positive,Recall

ZHB1104,True Positive,Recall

EPG1196,True Positive,Recall

PTM1432,True Positive,Recall

2：

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

MMR1458,True Positive

PBC0077,True Positive

JIB1258,True Positive,Recall

SCO1719,True Positive,Recall

VRP0267,True Positive

ZJN1492,True Positive,Recall

SAF1942,True Positive,Recall

PTV0067,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

DTB0722,True Positive,Recall

CEW1960,True Positive,Recall

ZHB1104,True Positive,Recall

EPG1196,True Positive,Recall

PTM1432,True Positive,Recall

3：

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

MMR1458,True Positive

PBC0077,True Positive

JIB1258,True Positive,Recall

SCO1719,True Positive,Recall

VRP0267,True Positive

ZJN1492,True Positive,Recall

SAF1942,True Positive,Recall

PTV0067,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

DTB0722,True Positive,Recall

CEW1960,True Positive,Recall

ZHB1104,True Positive,Recall

EPG1196,True Positive,Recall

PTM1432,True Positive,Recall

4：

VCF1602,True Positive,Recall

CKP0630,True Positive

MMR1458,True Positive

PBC0077,True Positive

JIB1258,True Positive,Recall

SCO1719,True Positive,Recall

VRP0267,True Positive

ZJN1492,True Positive,Recall

SAF1942,True Positive,Recall

PTV0067,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

DTB0722,True Positive,Recall

CEW1960,True Positive,Recall

ZHB1104,True Positive,Recall

EPG1196,True Positive,Recall

PTM1432,True Positive,Recall

于此同时，Insiders方面：（非常理想，8个Recall了7个，只有Insiders\_3漏了）

Insider\_1:

EPG1196,2010-08-05,

KBC1390,2010-08-10,

PBC0077,2010-08-13,

SAF1942,2010-08-13,

Insider\_2:

VCF1602,2010-08-20,

CKP0630,2010-08-26,

ZIE0741,2010-08-27,

Insider\_3:

VRP0267,2010-08-08,

验证发现，上述Insiders为什么都被判定为了FP呢？看来程序存在错误

最终发现错误来自于分析tp\_set,recall\_set,fp\_set时颠倒了顺序，因此出现了错误，重新调整后，每月份TP 用户都在验证集的最后末尾，很容易识别。

既然自己开始的实验是以在职[-1]与[+1]离职区分的，那么如果直接以Insiders作为标识计算Recall，那么想必FPR应该不会有如此之低，换句话而言，nu值会相应提升才对；

然后在Insiders标识的基础上，进一步重新训练五折OCSVM，然后对N+1月份的原始ATF用户进行预测。

投票阈值最低为3，最高为5；

新的机遇Insiders的Labels后，

一般解雇用户：PTM1432,-1

Insiders：PBC0077,1

验证通过！

然后尝试重新使用Insiders作为标记实验：miu = 0.8

0 OCSVM：

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive

VRP0267,True Positive

SAF1942,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

1 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive

VRP0267,True Positive

SAF1942,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

2 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive

VRP0267,True Positive

SAF1942,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

3 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive

VRP0267,True Positive

SAF1942,True Positive,Recall

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

4 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive

VRP0267,True Positive

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

可以发现，8个Insiders中识别出了6个，考虑是由于解雇用户混淆了OCSVM导致出现了失误。

如果选择完全由解雇用户刻画的ATF，那么miu=0.9时，

0 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive,Recall

VRP0267,True Positive,Recall

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

1 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive,Recall

VRP0267,True Positive,Recall

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

2 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive,Recall

VRP0267,True Positive,Recall

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

3 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive,Recall

VRP0267,True Positive,Recall

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

4 OCSVM

VCF1602,True Positive,Recall

CKP0630,True Positive,Recall

PBC0077,True Positive,Recall

VRP0267,True Positive,Recall

SAF1942,True Positive

KBC1390,True Positive,Recall

ZIE0741,True Positive,Recall

EPG1196,True Positive,Recall

问题是为啥五个OCSVM的结果完全一样！这样还怎么区分Insiders呢？难道要设置OCSVM投票阈值为5么？

一共383个初始验证用户，+8个Insiders，一共391个用户全部识别为Positive了

0\_OCSVM结果：

383中误报76个

Recall=2/8

1\_OCSVM:

2010-08 五个最优OCSVM依次为：

No: 0 [0.8, 0.5, 0.02578328981723238, **0.25**, **0.19843342036553524**, **0.19948849104859334**, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.8, random\_state=None, shrinking=True, tol=0.01,

verbose=False), [383, 384, 385, 386, 387, 388, 389, 390], [385, 389], [0, 3, 5, 6, 12, 14, 19, 31, 35, 38, 40, 53, 58, 63, 69, 79, 80, 82, 85, 86, 90, 95, 101, 104, 107, 118, 127, 136, 145, 150, 151, 153, 161, 166, 176, 177, 179, 180, 183, 206, 215, 216, 234, 239, 241, 244, 249, 257, 262, 264, 266, 268, 279, 280, 288, 289, 295, 301, 309, 310, 311, 327, 328, 334, 339, 340, 351, 353, 358, 359, 362, 368, 370, 377, 380, 382]]

No: 1 [0.25, 0.5, 0.10460182767624021, **0.875**, **0.6657963446475196**, 0.670076726342711, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.25, random\_state=None, shrinking=True, tol=0.01,

verbose=False), [383, 384, 385, 386, 387, 388, 389, 390], [383, 384, 385, 386, 388, 389, 390], [0, 1, 2, 3, 4, 5, 8, 9, 10, 11, 14, 17, 19, 20, 21, 22, 24, 25, 28, 31, 32, 33, 34, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 49, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 72, 73, 74, 75, 76, 77, 80, 81, 82, 83, 84, 85, 87, 88, 89, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 103, 105, 107, 108, 109, 112, 114, 117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 133, 134, 136, 139, 141, 143, 144, 146, 148, 149, 151, 152, 155, 157, 158, 161, 162, 163, 164, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 179, 180, 181, 182, 184, 186, 187, 188, 189, 191, 192, 193, 195, 197, 198, 199, 200, 201, 202, 204, 205, 207, 208, 209, 210, 211, 215, 216, 217, 219, 223, 225, 226, 227, 229, 230, 232, 236, 237, 238, 239, 243, 245, 249, 250, 251, 252, 253, 254, 255, 257, 259, 260, 262, 263, 264, 265, 266, 267, 268, 269, 270, 272, 273, 274, 276, 277, 279, 280, 283, 284, 286, 287, 288, 289, 291, 294, 295, 297, 299, 300, 301, 302, 305, 307, 308, 309, 310, 312, 313, 315, 316, 317, 318, 319, 324, 325, 326, 328, 329, 331, 332, 333, 334, 335, 336, 337, 338, 339, 341, 342, 343, 346, 347, 348, 349, 350, 351, 352, 353, 355, 356, 358, 359, 360, 361, 363, 365, 366, 367, 371, 372, 373, 379, 380, 381]]

No: 2 [0.25, 0.5, 0.045855091383812, **0.875**, **0.783289817232376**, 0.7851662404092071, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.25, random\_state=None, shrinking=True, tol=0.01,

verbose=False), [383, 384, 385, 386, 387, 388, 389, 390], [383, 384, 385, 386, 388, 389, 390], [0, 1, 2, 4, 5, 6, 8, 10, 13, 14, 15, 16, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 41, 42, 43, 44, 46, 48, 50, 51, 52, 54, 55, 56, 57, 59, 60, 61, 64, 66, 67, 68, 69, 71, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 98, 99, 100, 101, 102, 105, 107, 108, 109, 110, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 128, 129, 130, 131, 132, 134, 135, 136, 137, 138, 140, 141, 142, 143, 144, 145, 147, 148, 149, 150, 151, 152, 153, 154, 156, 157, 158, 159, 160, 161, 162, 163, 164, 166, 167, 168, 169, 170, 171, 174, 175, 176, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 191, 192, 194, 195, 196, 197, 198, 200, 201, 202, 204, 205, 206, 207, 208, 209, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 222, 224, 225, 226, 227, 228, 231, 232, 234, 236, 237, 238, 241, 243, 244, 245, 246, 247, 248, 251, 252, 253, 255, 256, 257, 258, 259, 261, 262, 264, 266, 267, 268, 269, 270, 271, 272, 274, 275, 277, 279, 280, 281, 282, 283, 284, 285, 286, 288, 289, 290, 291, 293, 294, 295, 299, 301, 302, 303, 304, 305, 306, 307, 308, 310, 311, 312, 313, 317, 318, 319, 323, 324, 325, 326, 327, 328, 329, 330, 332, 333, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 357, 358, 359, 362, 363, 364, 365, 366, 367, 368, 369, 372, 373, 375, 376, 377, 378, 379, 380, 382]]

No: 3 [0.4, 0.5, 0.0799608355091384, **0.75**, **0.5900783289817232,** 0.5933503836317136, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False), [383, 384, 385, 386, 387, 388, 389, 390], [384, 385, 386, 388, 389, 390], [0, 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 18, 21, 22, 23, 24, 26, 27, 34, 35, 36, 37, 38, 41, 43, 44, 47, 50, 52, 56, 58, 60, 62, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 81, 82, 84, 87, 88, 89, 90, 92, 93, 94, 95, 97, 100, 103, 104, 105, 106, 107, 109, 112, 113, 114, 115, 117, 119, 120, 121, 122, 123, 125, 127, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 144, 145, 147, 148, 149, 152, 155, 156, 158, 160, 161, 162, 165, 166, 168, 169, 170, 171, 172, 173, 174, 177, 179, 181, 182, 188, 190, 191, 193, 195, 196, 197, 202, 204, 205, 206, 209, 210, 212, 216, 218, 219, 220, 221, 226, 227, 228, 230, 232, 233, 236, 237, 240, 242, 243, 245, 247, 249, 252, 254, 256, 261, 262, 264, 265, 266, 267, 268, 269, 270, 276, 277, 279, 280, 281, 282, 283, 284, 285, 286, 288, 292, 294, 295, 296, 297, 298, 300, 301, 302, 307, 308, 309, 313, 315, 316, 317, 319, 321, 325, 326, 327, 328, 329, 330, 331, 332, 334, 335, 336, 338, 339, 341, 342, 343, 344, 345, 348, 349, 350, 352, 353, 354, 355, 357, 358, 366, 368, 369, 371, 372, 374, 375, 378, 380, 382]]

No: 4 [0.25, 0.5, 0.0579663212435233, 0.875, 0.7590673575129534, 0.7614213197969543, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.25, random\_state=None, shrinking=True, tol=0.01,

verbose=False), [386, 387, 388, 389, 390, 391, 392, 393], [386, 387, 388, 389, 391, 392, 393], [0, 1, 2, 3, 4, 8, 9, 12, 13, 14, 15, 17, 19, 22, 23, 24, 25, 29, 31, 32, 34, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 82, 84, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 100, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127, 129, 130, 132, 133, 134, 135, 136, 137, 138, 140, 141, 142, 143, 144, 145, 147, 148, 149, 151, 152, 154, 160, 161, 162, 163, 169, 170, 173, 174, 176, 177, 178, 180, 181, 182, 183, 184, 187, 188, 189, 190, 191, 192, 193, 195, 196, 197, 198, 199, 200, 201, 203, 206, 208, 209, 210, 211, 212, 213, 215, 216, 217, 219, 220, 221, 224, 226, 227, 228, 229, 231, 232, 233, 234, 235, 236, 237, 239, 240, 242, 243, 244, 245, 246, 247, 249, 250, 251, 252, 253, 254, 255, 257, 258, 259, 260, 262, 263, 264, 265, 266, 267, 270, 271, 272, 274, 275, 276, 277, 278, 279, 280, 281, 282, 284, 285, 288, 289, 290, 293, 294, 296, 297, 298, 299, 300, 301, 302, 303, 304, 306, 307, 308, 309, 311, 312, 313, 316, 317, 318, 319, 320, 321, 323, 324, 325, 326, 327, 328, 329, 330, 332, 333, 334, 335, 336, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 372, 373, 374, 375, 377, 378, 379, 380, 381, 382, 383, 384]]

下午11时48分

接下来，似乎需要对结果进行过滤：

1. 过滤OCSVM的结果，从中减少FP；
2. 聚类分析过滤OCSVM的训练集合，使得

今天最后的2010-08的五折OCSVM实验表明：

对于LaidOff\_ATF而言，当默认miu=0.5，即recall与fpr同样重要时，我们得到了相对不同的五个OCSVM分类器（以Insiders作为[+1]标签）

明天需要基于2010-08的学习结果预测2010-09，并分析结果的recall/fpr；

考虑可能误报比较高，需要设置五个OCSVM投票的阈值，一般要求[3,5]

可能的思路：

事先对要分析的2010-09的ATF进行筛选，从中去掉大量的普通用户，对于剩余用户进行预测投票，选择K>3的用户；

为此，需要先对每个月的ATF数据进行聚类，查看当月的10多个insiders的分布，若分布于特定的群簇，则没问题；否则需要另寻他法。

二〇一八年十二月二十九日星期六

上午11时21分

之前的ATF特征中包含着用户的OCEAN数据，由于CPB也是由OCEAN计算，那么是否可以省略掉原始的OCEAN数据呢？

修改ATF数据的提取规则：

如果仅适用ATF[:, 5:]的部分计算，得到的OCSVM0.75的召回率，即8个用户里面2个丢失，但是相应误报率降低到230左右，几乎降低了1/3

然而上述问题的关键是结果基本一致，不足以构成差异分类器

No: 0 [0.4, 0.5, 0.060378590078328964, 0.75, 0.6292428198433421, 0.6317135549872123, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False),

No: 1 [0.4, 0.5, 0.08387728459530025, 0.75, 0.5822454308093995, 0.5856777493606138, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False),

No: 2 [0.4, 0.5, 0.09693211488250653, 0.75, 0.556135770234987, 0.5601023017902813, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False),

No: 3 [0.4, 0.5, 0.09040469973890342, 0.75, 0.5691906005221932, 0.5728900255754475, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.4, random\_state=None, shrinking=True, tol=0.01,

verbose=False),

No: 4 [0.45, 0.5, 0.08613989637305697, 0.75, 0.5777202072538861, 0.5812182741116751, OneClassSVM(cache\_size=200, coef0=0.0, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, nu=0.45, random\_state=None, shrinking=True, tol=0.01,

verbose=False),

试着依据2010-08预测2010-09月份三类Insiders离职情况：（K\_Vote = 5）

2010-09 Predicotr Results: recall: 0.8 fpr: 0.556082148499 risk\_ratio: 0.55735987428

Insiders分布：

Insiders-1

EPG1196,2010-08-05,

KBC1390,2010-08-10,

PBC0077,2010-08-13,

SAF1942,2010-08-13,

**ALT1465,2010-09-08,**

**SLL0193,2010-09-23,**

**IHC0561,2010-09-28,**

JKB0287,2010-10-04,

DNJ0740,2010-10-26,

Insiders\_2

VCF1602,2010-08-20,

CKP0630,2010-08-26,

ZIE0741,2010-08-27,

**SIS0042,2010-09-02,**

**TNB1616,2010-09-10,**

**TRC1838,2010-09-15,**

**MDS0680,2010-09-17,**

**WDT1634,2010-09-20,**

**OSS1463,2010-09-21,**

**CIF1430,2010-09-23,**

MCP0611,2010-10-06,

CHP1711,2010-10-13,

GWG0497,2010-10-15,

KSS1005,2010-10-16,

Insiders\_3

MPF0690,2010-06-18,

CRD0272,2010-07-16,

VRP0267,2010-08-08,

ELM1123,2010-10-15,

GKW0043,2010-11-05,

ACA1126,2010-12-03,

KCM0466,2011-02-11,

ZEH0685,2011-04-01,

LAH0463,2011-05-06,

CWW1120,2011-05-13,

继续分析：

Insiders-1

**ALT1465,2010-09-08, ALT1465,1,1,1,1,1,5,1,**

**SLL0193,2010-09-23, SLL0193,-1,-1,-1,-1,-1,0,-1,**

**IHC0561,2010-09-28, IHC0561,-1,-1,-1,-1,-1,0,-1,**

Insiders\_2

**SIS0042,2010-09-02, SIS0042,1,1,1,1,1,5,1,**

**TNB1616,2010-09-10, TNB1616,-1,-1,-1,-1,-1,0,-1,**

**TRC1838,2010-09-15, TRC1838,1,1,1,1,1,5,1,**

**MDS0680,2010-09-17, MDS0680,1,1,1,1,1,5,1,**

**WDT1634,2010-09-20, WDT1634,-1,-1,-1,-1,-1,0,-1,**

**OSS1463,2010-09-21, OSS1463,-1,-1,-1,-1,-1,0,-1,**

**CIF1430,2010-09-23, CIF1430,1,1,1,1,1,5,1,**

在继续修改之前，先来解决目前recall计算错误的问题，为啥10个recall里识别出5个recall还是0.8呢？

问题解决：

预测月数据应为self.Next\_Cert52\_Crt\_Users

Cal\_Messures函数的cnt\_recall未使用，应使用recall替换

由于当miu=0.5的时候，漏报率将近0.5，故提高阈值，采取优先考虑recall的情况，得到了新的结果：

Miu = 1.0

K\_Vote = 5

2010-09 Predicotr Results: recall: 8.0 0.8 fpr: 0.865192206424 risk\_ratio: 0.864850707177

如果降低K\_Vote=4

结果不变；

上述实验依靠LaidOff的ATF，且PCA自动降维模式，与ATF特征模式未变

PCA=10/5的结果都要好于PCA()的33个维度，说明特征中有一些事干扰特征；

若PCA=3(2010-08预测2010-09最优)

**Miu=0.8 : 2010-09 Predicotr Results: recall: 9.0 0.9 fpr: 0.68299104792 risk\_ratio: 0.68412781561**

**Miu=1.0 : 2010-09 Predicotr Results: recall: 10.0 1.0 fpr: 0.810953133228 risk\_ratio: 0.811943425877**

若PCA=10

Miu-0.8 : 2010-09 Predicotr Results: recall: 9.0 0.9 fpr: 0.784623486045 risk\_ratio: 0.785227867994

Miu=1.0 :2010-09 Predicotr Results: recall: 10.0 1.0 fpr: 0.867825171143 risk\_ratio: 0.868517548455

若PCA=5

Miu=1.0: 2010-09 Predicotr Results: recall: 9.0 0.9 fpr: 0.876250658241 risk\_ratio: 0.876375065479

Miu=0.5 : 2010-09 Predicotr Results: recall: 7.0 0.7 fpr: 0.570300157978 risk\_ratio: 0.570979570456

Miu=0.8 : 2010-09 Predicotr Results: recall: 8.0 0.8 fpr: 0.71037388099 risk\_ratio: 0.710843373494

上述方案先予以保留，受到更少PCA表现更好的启发，考虑一个替代方案：

若将原先的ATF特征按照Personality(5+2+1), CPBs（8），LED（8），WorkDays（3）分别PCA-->3或者1，然后对于新的10维度特征进行现有的五折OCSVM？

ATF\_10D-0.1.csv进行处理

下午11时19分

明确两个思路：

1. 使用中期时的研究思路，即先对用户ATF进行KMeans，若Insidrs聚集在某些确定群簇中可以区分开，则可使用不含Insiders的群簇训练OCSVM，Insiders训练Recall的OCSVM参数；然后预测N+1月的KMeans结果；
2. 从ATF特征优化入手，选择PCA-->10/12，然后运行五折OCSVM查看结果；或分析所有误报中的特点，继续进一步筛选；

二〇一八年十二月三十日星期日

上午8时40分

实验一：

将原始ATF数据降维到10维度（3+3+3+1），之后使用ATF\_10D进行后续五折OCSVM训练，当然，后续的训练不需要再做PCA，但是需要做scale

经过实验，发现同样情况不如pca(3)

2010-09 Predicotr Results: recall: 9.0 0.9 fpr: 0.893101632438 risk\_ratio: 0.893137768465

于是决定放弃该条路线，转而考虑已有方法的改进，即KMeans的上的应用

在进行新的实验前，先把所有具有Insiders的月份结果按照上述实验PCA(3),Miu=0.8/1.0进行验证：

结果有一点达到自己的预期，即召回率针对Insiders\_2还是可以的，有波动：

1. 初始Insiders数据少，分类训练结果一般；
2. 但是随着Insiders数据增多，recall效果越来越好；
3. 最后当Insiders\_2少的时候，针对Insiders\_1/3效果并不好，因此重点检测insiders\_2
4. 标签依据：是否主动离职；默认Insiders主动离职，其余用户均为解雇

PCA=3， Miu=0.8，且laidoff\_ATF

2010-07 Predicotr Results: recall: 1.0 0.25 fpr: 0.031007751938 risk\_ratio: 0.031459515214

2010-08 Predicotr Results: recall: 4.0 0.5 fpr: 0.50417101147 risk\_ratio: 0.504153686397

2010-09 Predicotr Results: recall: 9.0 0.9 fpr: 0.686150605582 risk\_ratio: 0.68727082242

2010-10 Predicotr Results: recall: 6.0 0.857142857143 fpr: 0.701595744681 risk\_ratio: 0.702172760996

2010-11 Predicotr Results: recall: 5.0 0.833333333333 fpr: 0.751472951259 risk\_ratio: 0.751735184196

2010-12 Predicotr Results: recall: 7.0 0.875 fpr: 0.808879263671 risk\_ratio: 0.809164420485

2011-01 Predicotr Results: recall: 3.0 1.0 fpr: 0.827322404372 risk\_ratio: 0.827605019094

2011-02 Predicotr Results: recall: 5.0 0.714285714286 fpr: 0.48401323043 risk\_ratio: 0.484898407468

2011-03 Predicotr Results: recall: 2.0 0.333333333333 fpr: 0.623123957754 risk\_ratio: 0.62216066482

2011-04 Predicotr Results: recall: 5.0 1.0 fpr: 0.717245240761 risk\_ratio: 0.718034617532

2011-05 Predicotr Results: recall: 3.0 0.75 fpr: 0.724040632054 risk\_ratio: 0.724099099099

上午10时0分

接下来，首先从结果中看是否可以依据Desicion\_Function筛选结果降低误报率

我们先来看看2010-08月预测2010-09月的结果，重点是三类Insiders的结果

Insiders\_1

ALT1465,2010-09-08,

ALT1465,1,0.8472619512494189,1,0.852544700635724,1,0.8469712852231985,1,0.850611950587242,1,0.8465903575011787,5,1,**0.8489046110995775,0.8510934082766286**,(最后两列为五个DF均值、中位数的MinMax结果)

SLL0193,2010-09-23,

SLL0193,1,0.8602877272903818,1,0.8595726902036225,1,0.8495381992810126,1,0.8583800142083563,1,0.8585219521134356,5,1,**0.8573678314396683,0.8589212009368236,**

IHC0561,2010-09-28,

IHC0561,1,0.9077589368363876,1,0.9067784783385355,1,0.8958733223675464,1,0.9058177192956487,1,0.9063289007903358,5,1,**0.9046246422195092,0.9054525155068499**,

JKB0287,2010-10-04,

JKB0287,1,0.9237485273768457,1,0.9351289183604443,1,0.9336901643586957,1,0.9364424351678176,1,0.9337255881805755,5,1,

DNJ0740,2010-10-26,

DNJ0740,-1,0.8103116354443715,-1,0.8067223306838843,-1,0.8200902430338903,-1,0.807790533794939,-1,0.8046761820498947,0,-1,

Insiders\_2: 若使用中位数作为指标，则可以设置阈值0.89从而得到提升，7个Insiders检出5个，3个Insiders检出1个；

如果使用[+1]与DF中位数>0.89作为筛选，那么最终得到760个用户，约占全体用户的0.4

虽然已经从0.68降低到了0.4，但是依然不够啊

743 0.389209009953

SIS0042,2010-09-02,

SIS0042,1,0.9646773389119544,1,0.9689238523996756,1,0.9627975099063372,1,0.9684522177925159,1,0.9700904098605688,5,1,**0.9671049691450816,0.9681302563024174**,

TNB1616,2010-09-10,

TNB1616,1,0.8926803071776833,1,0.8874965029609507,1,0.8865377449518428,1,0.8917638463155525,1,0.8959681910165238,5,1,**0.8910150245432509,0.892764792934502**,

TRC1838,2010-09-15,

TRC1838,1,0.966565593913755,1,0.962991065929729,1,0.9643453344050328,1,0.9642685083284237,1,0.9651595730283199,5,1,**0.9648063297337668,0.9661847315972283,**

MDS0680,2010-09-17,

MDS0680,1,0.9021641837276868,1,0.8991588887681458,1,0.8946713920884201,1,0.9001395510589849,1,0.9031472546489359,5,1**,0.8999760368587467,0.9012072511786339,**

WDT1634,2010-09-20,

WDT1634,-1,0.7034404886425611,-1,0.7181716224873425,-1,0.6918849918496169,-1,0.6836050411120146,-1,0.7027700620929525,0,-1**,0.6999658700477165,0.7005398387582766,**

OSS1463,2010-09-21,

OSS1463,1,0.8607291230615078,1,0.860174715226664,1,0.8480132819250041,1,0.8587840924295579,1,0.8599994041296558,5,1,**0.8576431696318099,0.8593669140019274,**

CIF1430,2010-09-23,

CIF1430,1,0.9786690021170041,1,0.9728963240312256,1,0.9770075797582711,1,0.9763120608083835,1,0.9744039747691906,5,1,

CIF1430,1,0.9749627147803382,1,0.976182443208186,1,0.97291699154207,1,0.9759402063989802,1,0.9771341318287192,5,1**,0.975554454794431,0.9751206723249098,**

MCP0611,2010-10-06,

CHP1711,2010-10-13,

今天上午的结论：

1. 在PCA=3，miu=0.8的LaidOff-ATF分析上，获得了2010-06月开始最好召回率（针对Insiders\_2）；
2. 然而误报率较高，故采取OCSVM的DF函数进行筛选，使用的是五个DF函数和的中位数作为指标，从2010-08-->2010-09来看，使用[+1]与DF\_Median>0.89可以有效筛选出743个Risk用户，占到全体用户的0.39左右；（已经比开始的误报率0.78-0.82降低了很多了）

GWG0497,2010-10-15,

KSS1005,2010-10-16,

Insiders\_3

VRP0267,2010-08-08,

二〇一八年十二月三十日星期日

下午10时5分

总结下今天上午的工作。

1. 通过重组PCA的逐月预测实验发现，虽然PCA=3时原始的ATF有了最好的结果表现，针对Insiders\_2的Recall较高，且FPR已经降到0.81附近；后期根据2010-08-->2010-09的实验，利用Decision Function作为进一步筛选的结果，在保证Recall不变的情况下，FPR降到0.39附近，然而，已经很难继续降低提高精度了；
2. 于是自己决定采用原有方案的改进，即假设已经获取了2010-01:2011-05的全年数据，从最初的阶段开始（基于CPB特征）来预测，只不过将用户根据是否离职区分为[-1]与[+1]用户，仅用所有在职用户[-1]来预测识别[+1]用户中的Insiders\_1/2/3;或者采用全局ATF特征，即使用所有用户最终阶段的包含了LCE均值的新ATF特征作为分析的全局依据，依然利用在职坐KMeans聚类选择中上满意度建模，检测离职用户中的Insiders。
3. 最终输出可疑Insiders的Decision Function数值作为后续分析的依据。

二〇一八年十二月三十一日星期一

上午9时16分

2018年的最后一天，在科研中度过，晚上记得去写一篇2018年最后的日志吧！

昨天经过实验，基本放弃了现有环境下考虑逐月、实时预测High Risk用户的方案，因为复杂度太高，自己即便获得了较好的Recall，但是无法将FPR降低到0.4以下。出于毕业论文的考虑，转而考虑次要的目标：

假设：已经获得了2010-01:2011-05的CERT5.2用户数据，那么如何结合用户的主客观特征分析得到其中Insiders呢？

既然有了全局数据假设，那么我们可以进而提出一个基于满意度的假设：

1. 如果一个用户工作满意度低-->CPBs行为倾向高-->消极怠工或者寻求跳槽-->离职（解雇/跳槽），但是离职未必说明工作满意度低（必要不充分条件）
2. 反之，如果一个用户一直工作，起码说明工作满意度不低
3. 为此，我们将全体用户按照在职与离职做第一次分组，将所有离职用户全部划归到待检测集合1（Test\_1）
4. 即便工作满意度不低，但是中低满意度用户都有可能转变为Insidiers，因此，我们实际上是想选择中上满意度的用户作为模型训练OCSVM，识别出所有可能的潜在Insiders
5. 具体方案依旧是对于所有在职用户做KMeans，然后计算不同群簇中各个特征部分的定性高低，从中选择出满意度高的群簇，进而训练OCSVM，剩余部分作为Test\_2；
6. 然后用训练得到的OCSVM去检测

核心思想是：

基于在职[-1]与离职[+1]对用户数据做初始划分，确保了所有因工作满意度低而离职的用户都在[+1]中，而还有一些工作满意度度低却还没有离职的，也是我们检测的目标；

故需要对[+1]用户做KMeans聚类，从中发现工作满意度不低的用户，训练出正常用户的工作满意度OCSVM模型，最后对于筛选出的[-1]类用户+[+1]类用户一起做OCSVM，最终判定其是否属于工作满意度低的高危用户；

上述方案起码在KMeans阶段不会损失Insiders，而关键可能还在后续分类的准确性上。

实验准备从两个数据集入手：

1. 原始的只考虑CPB的26维度JS特征；
2. 新的考虑了LCE的33维度的ATF特征；

分别建立上述模型分类算法，不同的区别在于KMeans群簇选择的依据；

建立新的实验目录：

KMeans\_OCSVM\_ATF\_Predictor

两类满意度特征：JS + ATF

自动KMeans聚类算法以及保存的最好的K值分类群簇结果（KMeans\_JS/KMeans\_ATF）

然后筛选群簇，并生成对应的OCSVM的训练集与测试集，并仅标注场景1-2-3的Insiders为[+1]，其余均为[-1]；

自动遍历参数nu，选择最好的OCSVM结果[miu \* recall - (1 - miu) \* fpr]

输出最好预测结果的recall/fpr/risk\_ratio