2018年11月9日星期五

上午9时35分

在连续经历了六个阶段的实验后，自己终于来到了第七个实验。作为所剩为数不多的机会，自己计划今天一天写出整个程序框架，并试着对于前几个月的结果进行初步的预测。

自己走了很多弯路。尝试了直接使用用户邮件联系人与离职用户表匹配来作为刻画低满意度风险的重要指标，然而直接分析所有用户的上述匹配，显然是不合适的。因此自己转变了思路，继而考虑按月来分析、排序用户的满意度风险（Low Job Satisfactory Risk, LJSR），并依据下一个月的预测结果反馈修正排序选择的高危用户的阈值。

目前，自己正在跑着刻画CERT5.2中用户合理上下班时间的脚本，目标是通过2010-01的用户登录数据，建立起上下班的固定时间，最终可以分析预测器（指建立的LJSR预测器）预测结果的准确性（LJSR的表现：一是离职，二是超过一定比例的迟到早退行为）。

因此，整体的程序逻辑比较复杂，关键是一个月的数据用来建立预测器，而下一个月的登录数据与离职数据则作为现实标签反作用于预测器进行某种程度的修正。

自己初步设想了，整体的循环应包含两个大模块：

* 预测器模块：细分为数据准备模块、特征构建模块、RelationLevel计算模块、特征保存模块（特征+RL数值）以及最终的结果输出模块；
* 检验反馈模块：细分为下个月标签提取模块（离职用户+迟到早退用户）、FN+FP计算模块、预测阈值反馈模块

目录组织上，整体结果位于实验0.7目录下；

其中按照月份建立一级子目录，如利用2010-02建立预测器，则首先建立一个子目录2010-02，其中应包含预测器所需要的所有数据；而2010-03子目录在结果验证时便建立，然后其中保存着上一个预测器的检验反馈模块以及新预测器的分析数据，即每个月份目录中应包含：

1. 上一个预测器的检验反馈模块
2. 当前预测器的预测器模块

今早进行的是昨晚未完成的2010-01月份的登录统计实验，自己的目标是在统计之后建立提取固定上下班时间的方法，以此作为该用户之后是否迟到早退的依据。

由于美国实行夏令时与冬令时体制，因此其上下班时间数值上是一样的。

对于CERT5.2用户而言，大多数用户的上下班时间（系统登录登出时间）是明确的，即一个确定值+一个小方差

如：下述用户为最一般的用户：

下列用户在每天均有登录和登出，一般而言，以半小时为一个时间窗口，从早上7：30开始到9：30，然后从13：00开始到19：30，形成多个时间窗口，统计用户在不同时间窗口内登录次数与登出次数，票数最多的时间窗口上限/下限作为固定时间。

上述方法的背后假设是：用户正常应大部分准时上下班，即在上班时间前，下班时间后离开，而我们假设2010-01第一个月为正常月份；

01/02/2010 08:57:00,AAB1302,PC-5565,Logon,

01/02/2010 13:07:19,AAB1302,PC-5565,Logon,

01/02/2010 19:14:00,AAB1302,PC-5565,Logoff,

01/04/2010 09:03:00,AAB1302,PC-5565,Logon,

上述方法也能够使用下述不规范上班的用户，比如系统运维管理员，常常需要夜里工作，当然也不要求第二天按时上下班，如JBI1134即所属ITAdmin部门。此时依旧可以统计，但是最早的时间段【7:00之前】有最多的登录次数，此时该用户的上班时间确定为不适宜固定，在我们现有的分析思想中暂时不纳入考虑迟到早退的用户；而且可以看到下午14：29就下班了不是？【Before 7:00】, 【7:00 - 7:30】， 。。。

01/02/2010 02:24:51,JBI1134,PC-0168,Logon,

01/02/2010 02:38:28,JBI1134,PC-0168,Logoff,

01/02/2010 04:55:52,JBI1134,PC-2320,Logon,

01/02/2010 05:02:28,JBI1134,PC-2320,Logoff,

01/02/2010 06:36:55,JBI1134,PC-5439,Logon,

01/02/2010 06:38:42,JBI1134,PC-5439,Logoff,

01/02/2010 06:50:00,JBI1134,PC-7235,Logon,

01/02/2010 07:13:59,JBI1134,PC-2226,Logon,

01/02/2010 07:16:08,JBI1134,PC-2226,Logoff,

01/02/2010 10:41:45,JBI1134,PC-7235,Logon,

01/02/2010 14:10:33,JBI1134,PC-7872,Logon,

01/02/2010 14:29:38,JBI1134,PC-7872,Logoff,

试着再看一个用户，该用户的上班时间更早

01/02/2010 06:35:00,HMI1448,PC-9352,Logon,

01/02/2010 11:02:11,HMI1448,PC-9352,Logon,

01/02/2010 16:47:00,HMI1448,PC-9352,Logoff,

因此上班时间窗口应定在【before 6:00】【6：00-6：00】【6：30-7：00】

因此，我们实验七的第一个模块用于从2010-01用户登录/登出数据中确定该用户的固定上下班时间，且保存到指定的目录。

12时6分0秒

利用2个小时的时间写好了上下班时间模块，开始验证

AAB1302，

WorkOn is like: [[6.5, 0.0], [7.0, 0.0], [7.5, 0.0], [8.0, 0.0], [8.5, 0.0], [9.0, 14.0], [9.5, 10.0]]

WorkOff is like: [[16.0, 0.0], [16.5, 0.0], [17.0, 0.0], [17.5, 0.0], [18.0, 0.0], [18.5, 8.0], [19.0, 16.0]]

基本验证通过

2018年11月10日星期六

下午2时47分

今天利用周六下午半天的时间，写完实验系列7的主体程序初版。

昨晚已经完成了实验的准备部分，即：

* 初始化了分析所用的数据源
* 初始化了预测与验证结果的输出目录
* 初始化了月份目录，并尝试写入了当月组织内的离职用户名单

接下来，我们开始进行实验的主体设计实现。

第一个是预测器模块，该模块又主要分为两个模块：构建预测器（Build\_Predictor）与运行预测器（Run\_Predictor）；而验证分析模块则包括结果度量与反馈更新两个模块，结果度量（Predictor\_Messure）又可以分为自动标记低满意度用户（Auto\_Label）与度量计算（Cal\_Messurement）两个子模块。

接下来，我们简要列举说明下各个功能模块的核心功能：

预测器模块：

1. 构建预测器模块：

* 分析当月CERT5.2所有用户的邮件通信情况，与当月离职情况交叉匹配后得到该月该用户需分析的离职联系人；
* 从当月及以前的邮件通讯中提取该用户的邮件9元特征，
* 提取该用户的离职联系人的OCEAN特征；
* 提取该用户与离职联系人的OS距离（四位OS异或码的十进制）
* 将上述三类特诊按照人格、OS距离、邮件的形式拼接得到该用户的离职联系人Relationship Level特征，并按照公式计算其对应的RL值，写入到文件中user-id\_RL\_feat-version.csv

1. 运行预测器模块：

* 依据该用户的离职联系人的RL数值，求其和作为该用户的JS\_Risk值；
* 重复上述步骤，计算完毕CERT5.2所有用户（除去CEO），并保存到本次预测器的结果文件（JS\_Risk由高到低）【两个文件，一个是排序的全部JS\_Risk文件，一个是按照既定比例输出的低满意度嫌疑用户：month\_ratio\_LJSR.csv】

检验器模块：

1. 自动标记模块

* 读取CERT5.2中所有用户最初的上下班时间
* 提取检测月CERT5.2所有用户上班迟到与下班早退的天数以及占当月工作天数的比例；
* 当月用户出勤有问题的用户（迟到+早退天数所有用户排序50%以上）与当月离职用户统统作为低JS用户标记

1. 度量计算模块

* 根据自动的标签作为认为的真实标签，对于预测器的结果进行FPR/TPR的检测，同时还计算FN
* 根据FN与FP的大小关系，更新预测器中的选择比例阈值（最初设定为5%，小概率事件）

2018年11月11日星期日

上午12时15分

初步写完了Predictor\_Module中的邮件特征提取部分，这部分首先验证：

RMB1821,1.0,10.0,[2010-01-13; 2010-01-21; 2010-01-25; 2010-01-26; 2010-01-28; 2010-02-08; 2010-02-09],462898.5,5.0,0,[],0,0,7,1

满足BYO1846中与RMB1821的关系

首先通过程序自动验证，截止到2010-02，BYO1846离职的联系人是否只有RMB一个用户？

继续手动验证是否有遗漏：

BYO1846,2010-12-15,

Insider\_LaidOff\_0,RMB1821,FDS1841,TRC1838,

Insider\_LaidOff\_1,NWP1609,ZAD1621,TAG1610,MAR1075,JXH1061,KEW0198,LSM1382,VCF1602,KBC1390,TNB1616,IVS1411,SLL0193,MFM1400,CTT0639,MZO1066,

Insider\_LaidOff\_2,WSW1091,JHP1654,JDM0208,HSF1115,UAM1108,CIM1095,ZHB1104,PTM1432,CIF1430,WDT1634,DHS0204,DDR1649,HMK0653,MIB0203,

经过程序验证，以BYO1846为例，2010-02月该用户离职的邮件联系人仅有

**离职联系人: RMB1821**

而BYO1846的联系人为下列用户，只有RMB出现在2010-02，验证通过

BYO1846,

Cluster0,

**RMB1821**,0.28571428571449997,0.0425531914893617,0.005187972305548536,0.0,0.0819672131147541,0.016017337873492615,0.075,0.595974689854,

**FDS1841**,0.3333333333335,0.0851063829787234,0.01938484458272191,0.027777777777777776,0.13114754098360656,0.053807737409929106,0.109375,0.626952130714,

**TRC1838**,0.3650793650795,0.48936170212765956,0.033426060894124456,0.033816425120777774,0.6557377049180328,0.003353036762244741,0.0,0.646521851626,

Cluster1,

**KBC1390**,1.0,0.02127659574468085,0.004174853102605735,0.0,0.0,0.0,0.0,0.474132273636,

Cluster2,

Cluster3,

**BRM0126**,1.0,0.02127659574468085,0.0028435066418784384,0.0,0.0,0.0,0.0,0.474114642349,

**OJC0930**,1.0,0.02127659574468085,0.4073085174754199,0.3333333333333333,0.0,0.0,0.0,0.483838113043,

**HXP0976**,0.0,0.0,0.0,0.0,0.01639344262295082,0.0018571578950494598,0.0,0.474095934906,

Cluster4,

**TCP0380**,1.0,0.02127659574468085,0.004310310575377622,0.0,0.0,0.0,0.0,0.474134067509,

**DTB0722**,1.0,0.02127659574468085,0.003073710993168567,0.0,0.0,0.0,0.0,0.474117691013,

**RPJ1159**,1.0,0.02127659574468085,0.004011521709443966,0.0,0.0,0.0,0.0,0.474130110622,

**ZVW1475**,1.0,0.02127659574468085,0.0035138255256404005,0.0,0.0,0.0,0.0,0.474123519554,

2018年11月11日星期日

下午3时57分

按照我们最初的假设，每个月份的用户邮件特征应该包括全部1999个CERT5.2用户，考虑到每个用户有联系人与特征两个文件，因此应是3998个文件。而实际运算却只有3100+个用户，明显不符合我们的假设。故重新检查。

修改的方法是：

将最后调用邮件特征生成文件的模块中，文件生成放在了函数调用模块初始的位置，不再后续的遍历循环中进行，从而得到了3998个用户。

而后期为了检查容易方便，应对CERT\_Users进行排序再送入分析

此时单独看email\_contacts或者email\_feats数据，都是各1999个，符合数据分布了。

采用之前循环内部建立文件的方法重新运行分析模块，查看是否得到全部3999个文件？

下午7时1分

根据家中的实验，对于CERT52\_Users进行了排序，然后将文件创建代码移到了循环内部，结果如此得到的结果又变得十分不稳定，成了只有3000个文件，约1500个用户，丢失了499个用户。

因此可以定位问题的原因就在最后邮件特征提取模块中的循环部分。

对比实验发现丢失了499个用户2010-02的邮件信息

Lost Users : 499

0 AAB1302

1 AAC0904

2 AAG1136

3 AAO1919

4 ABM0890

5 ACB0220

6 ACC0950

7 ACH1831

8 ACH1840

9 ACJ1797

下面就以缺失的用户 AAB1302为例进行分析

问题终于明确：

因为最初分析的用户是user\_id

而中途循环遍历时不小心又使用了user\_id，导致名称发生了改变，因此错误

修正后问题解决。

下午9时52分

但是在实际分析时，发现了一个特征提取的问题，即出现了接收邮件特征的遗漏，如

Rose.Maisie.Blackwell

[WMH1300,1.0,2.0,[2010-01-04; 2010-01-30],26533.5,0.0,0,[],0,0,2,1]

RMB同样有上述问题

RMB1821,1.0,10.0,[2010-01-13; 2010-01-21; 2010-01-25; 2010-01-26; 2010-01-28; 2010-02-08; 2010-02-09],462898.5,5.0,0,[],0,0,7,1

下午11时3分

关于离职时间的确认，LDAP中如果在2月离职，则2月的LDAP中该用户即已经不存在；但是该用户可以存在2月份的邮件等行为记录；

因此，我们关于离职时间的统计是以第一次不存在LDAP的月份为记录的，即该月份用户不在LDAP中，但是上个月份还在。

在这个意义上，RMB1821的离职时间确定为2010-02，因为该月LDAP中不再存在RMB用户，却存在到2-9日的邮件通信

同理WMH1300也存在2月2日的邮件通信

经过重新一步一步验证（需要同时考虑to/cc/bcc）

email\_feat is ['WMH1300', 0.0, 4.0, ['2010-01-04', '2010-01-06', '2010-01-15', '2010-01-30'], 163363.0, 0.0, 4.0, ['2010-01-18', '2010-01-19', '2010-01-27', '2010-01-29'], 11427627.0, 8.0]

同时发送接收的邮件联系人 WMH1300 特征提取完毕, 其邮件特征为： ['WMH1300', 0.0, 4.0, '[2010-01-04; 2010-01-06; 2010-01-15; 2010-01-30]', 40840.75, 0.0, 4.0, '[2010-01-18; 2010-01-19; 2010-01-27; 2010-01-29]', 2856906.75, 8.0]

同时发送接收的邮件联系人 WMH1300 特征提取完毕, 其邮件特征为： ['WMH1300', 0.0, 4.0, "['2010-01-04'; '2010-01-06'; '2010-01-15'; '2010-01-30']", 40840.75, 0.0, 4.0, '[2010-01-18; 2010-01-19; 2010-01-27; 2010-01-29]', 2856906.75, 8.0]

同时发送接收的邮件联系人 WMH1300 特征提取完毕, 其邮件特征为： ['WMH1300', 0.0, 4.0, "['2010-01-04'; '2010-01-06'; '2010-01-15'; '2010-01-30']", 40840.75, 0.0, 4.0, "['2010-01-18'; '2010-01-19'; '2010-01-27'; '2010-01-29']", 2856906.75, 8.0]

同时发送接收的邮件联系人 WMH1300 特征提取完毕, 其邮件特征为： ['WMH1300', 0.0, 4.0, "['2010-01-04'; '2010-01-06'; '2010-01-15'; '2010-01-30']", 40840.75, 0.0, 4.0, "'2010-01-18'; '2010-01-19'; '2010-01-27'; '2010-01-29'", 2856906.75, 8.0]

2018年11月12日星期一

0时23分7秒

最终关于邮件特征拼写通过验证，预计明天一个上午的时间写完Predictor与Check模块，下午就可以进行多次实验了。

2018年11月12日星期一

上午9时43分

早上继续完成周末的实验程序。

这次第一次运行AAB用户的一个离职用户，得到rl特征

WMH1300 rl\_feat is ['WMH1300', 39.0, 36.0, 20.0, 40.0, 20.0, 13.92838827718412, 0, 0, 0, 0, 0.0, 0.0, 4.0, 40840.75, 0.0, 4.0, 2856906.75, 8.0]

OS验证通过！

2018年11月13日星期二

上午10时32分

昨天基本完成了Predictor\_Module，该模块主要输出结果有：

* 当月离职用户列表及其LDAP信息：Leave\_Users\_*Month*.csv
* 与当月离职用户关联的CERT5.2中用户的邮件特征目录：CERT5.2\_Users\_EmailFeats-0.7
* 拼装组成当前待分析用户（与当月离职用户关联）的Relation Level特征：*Month*\_CERT5.2\_Users\_RL\_Feats.csv
* 由上述RL\_Feats文件进一步提取得到的中间变量文件，利于直接归一化：RLF\_Process.csv
* 结合当月以及之前的JSR数值，得到该月的CERT5.2在职用户的JSR，其中补全了本月与离职用户无关的其他在职用户：Current\_Month\_JSR.csv
* 从当月JSR文件中，依据JSR的高低排序，输出Risk\_Ratio所划定的高危用户列表；

2018年11月13日星期二

下午2时40分

在分析登录登出数据时发现，离职用户数据范围为2010-02 to 2011-05，登录数据却持续到了2011-06。由于2011年6月份没有用户离职，因此对于刻画我们所说的JSR而言意义不大，故对于我们而言，重点还是审视2010-03 到2011-05， 2011-06只能使用出勤率来进行标注，对于检测其他攻击用户而言意义不明。

因此，编程实现时顺便分析上2011年6月，但是计算FP/FN时需要区分对待

下午10时18分

白天基本完成了每个月用户登录系统的迟到与早退的情况，由于用户多用时长，故先使用一个月的个别用户进行验证：

我们以2010-01月份的用户数据，具体以

HRE1950,7.0,16.0,4.0,0.0,30,

为例，该用户当月所有登录登出数据为：

验证通过；

2018年11月14日星期三

上午9时7分

后悔没有早日升级内存，拿着一个4G的内存吃屎呢？

由于logon数据暂时还未有结果，虽然只需要跑一次，但是跑的实在太慢了！

先来简单验证下JS\_Riks的计算

按照我们的想法，每月JSR是当月JSR+之前的JSR之和

先看2010-02的当月JSR（Current\_Month\_JSR.csv代表截止到当月用户的JSR）

AAB1302,1.14218527305

AAC0904,2.64098618562

AAG1136,1.39001900276

而2010-02的当月JSR为（Current\_Related\_Leave\_JSR.csv）

继续看2010-03

当月JSR

AAB1302,1.23029298274

AAB1762,1.12460580953

AAC1489,1.22344543699

则预测下月的JSR为

AAB13021.23029298274

AAB17621.12460580953

AAC14891.22344543699

2018年11月14日星期三

上午11时14分

经过修改，发现是JSR特征写入时，忘记添加字段间的‘,’导致后续读取分割失败

修正后可以得到正确结果

line\_lst is ['AAB1302', '1.14218527305']

+ AAB1302,1.23029298274

= jsr\_lst is [['AAB1302', 2.3724782557878394],

中午开始运行全部15个月的JSR数据分析，下午即可以分析Insiders的命中情况。