二〇一八年十二月五日星期三

上午9时26分

CERT5.2中三类攻击场景离职的最早时间：

Leave\_Insiders\_Earliest\_Time:

Scene\_1: KEW0198,2010-07-29,

Scene\_2: VCF1602,2010-08-20,

Scene\_3: MPF0690,2010-06-18,

我们看最早的2010-06，主观分析是否可以从2010-06开始预测？

经过验证，MPF0690的用户恶意行为最早开始于2010-06-17

经过验证，KEW0198的用户恶意行为最早开始于2010-07-07

经过验证，VCF1620的用户恶意行为最早开始于2010-06-23

因此，我们可以将原有数据按照上述事件截断分为两段：

1. 训练集时间段：2010-01：2010-05，该时间段内将在职用户标记为+1，离职用户标记为-1，并由此提取特征训练SVM，并遍历参数空间得到最优参数，作为之后预测的分类器；抑或，建立一个分类器参数按照月份结果更新的机制，后续不断自我更新；
2. 使用训练得到的分类器对于2010-06开始预测谁会离职？重要的指标有两个（预测结果中包含当月离职Insiders的召回率；以及当月预测结果中，真正离职用户的召回率与误报率）
3. 简单的情况下，分类器依次对逐个月进行检测，输出结果；后续可以考虑对分类器进行实时进化调整，依据三个重要指标进行计算更新
4. 今天周三，我们先来实现一个朴素的主观SVM预测器；建立实验八，主体核心模块全部采用面向对象实现；

1. 朴素的主观SVM预测器特征说明：

对于CERT5.2的每个用户而言，

|  |  |
| --- | --- |
|  |  |
| **字段名称** | 注释 |
| **User\_id** |  |
| **Personality（5）** | Scores of O,C,E,A,N |
| **CPB-I/CPB-O（2）** | 由OCEAN计算得到的CPB-I/O分数 |
| **Cnt\_Late\_Days/Cnt\_Early\_Days** | 迟到与早退天数（before预测月份） |
| **Relationship with LC：同离职者的人际影响因子** | |
| **Dis\_OCEAN（1）** | 同虚拟离职综合体的人格偏差 |
| **Dis\_OS（1）** | 同虚拟离职综合体的组织距离 |
| **Cnt\_Send/Cnt\_Recv（2）** | 同虚拟离职综合体的收发邮件总数（before预测月份） |
| **Cnt\_Send\_Size/Cnt\_Recv\_Size（2）** | 同虚拟离职综合体收发邮件总字数大小（before预测月份） |
| **Cnt\_Send\_Attach/Cnt\_Recv\_Attach（2）** | 同虚拟离职综合体收发邮件中附带的邮件个数 |
| **Cnt\_Send\_Days/Cnt\_Recv\_Days（2）** | 同虚拟离职综合体收发邮件联系的天数 |
| **Cnt\_Email\_Days（1）** | 同虚拟离职综合体有邮件联系的天数（去掉了同时收发的重复天） |

1. 朴素的主观SVM预测器结构

* 类名称：JS\_SVM\_Predictor

|  |  |
| --- | --- |
| init | 构造函数：拷贝关键数据（OCEAN/LED/LC） |
| Del | 析构函数 |
| 提取用户的JS特征 | Extract\_User\_JS\_Feat（针对单个用户） |
| 标记用户类别（在职/离职） | Label\_Users\_OnJob |
| 构建SVM预测器 | Build\_SVM\_Predictor（数据scale，PCA） |
| 验证SVM预测器以确定最优参数 | Validate\_SVM\_Predictor |
| 执行预测器，检验新月份用户，输出预测结果（Recall/FPR/Alert\_Ratio） | Run\_SVM\_Predictor |

上述函数模块为一个完整的预测类应包含的模块，但是实际测试时，逐步测试，并根据阶段注释掉重复工作即可，然而，最初的一次完美运行应完整运行上述阶段。

**实验延长一周，下周12月16日完成，然后开始关联研究部分。**

然后我们仔细明确下各个函数部分应具备的功能：

1. 构造函数：对于实验八而言，需要拷贝几个重要的文件（***OCEAN文件：psychometric-5.2.csv； Leave\_Days文件：CERT5.2-Leave-Users\_OnDays\_0.6.csv； 出勤迟到与早退文件LED：V07\_CERT5.2\_Users\_WorkOn-Off\_Time\_Team.csv； LC信息：CERT5.2\_Users\_LeaveContacts\_EmailFeats.csv***）；此外，还需要建立一个训练集（2010-01：2010-04），一个验证集2010-05，以及后续月份的预测结果目录（2010-06：2011-05）
2. 提取用户特征：需要读取CERT5.2用户的OCEAN、LED以及LC特征文件，从中组成、拼接成该用户的当前阶段的JS特征；结果将依次保存在对应的目录，需要注意的是，训练集、验证集与预测集目录中存储的都是包含当月数据的更新后的用户JS特征与预测的下月的预测结果（训练集中没有）
3. 训练模块：基于2010-01：2010-04的四个月数据，构建针对CERT5.2的用户在2010-05月离职的判断，其中应同步去掉前四个月已经离职的用户；输出一个训练好的分类器结果（附带各个参数）；
4. 验证模块：基于2010-05月份离职的用户数据，遍历SVM预测器的参数空间，从中找到最优的参数组合，并更新最优分类器参数以及重新计算排除2010-05月份离职用户的适用于下月的用户JS特征；
5. **重要原则**：每个目录下存放与该月Predictor结果相关的数据，即该月目录下应该有：

* 训练该月Predictor的所需用户JS特征；
* 该月预测器Predictor的SVM参数；
* 该月Predicotr数据的判定结果，以及该月用户实际在职与离职的标记；
* 该月Predictor的结果评价：Insiders的Recall/离职用户的Recall/以及判定可能离职用户占所有用户的比重；

下午5时6分

将2010-01：2010-04月份的出勤数据汇集到一起，验证：

1-4月：JBI1134,6.5,16.0,32.0,13.0,100.0,

1月：JBI1134,6.5,16.0,6.0,3.0,25,-1

2月：JBI1134,6.5,16.0,10.0,3.0,25,-1

3月：JBI1134,6.5,16.0,9.0,2.0,25,-1

4月：JBI1134,6.5,16.0,7.0,5.0,25,-1

验证通过！



二〇一八年十二月六日星期四

上午9时29分

今天周四，继续周三未完成的实验。

由于程序比较复杂，自己昨天晚上有些头大，不过凡事都有第一次，这次就先写一个功能达标的初始版，之余代码优化啥的，等抛出结果再说吧！

两个务必注意：

* 每个定义变量的类型、示例说明
* 每个函数方法的说明

下午2时25分

针对CERT5.2中2010-01：2010-04训练集的JS特征，先以一个用户为例：

JBI1134 Until 2010-04, js feat is like: ['JBI1134', 46.0, 37.0, 45.0, 10.0, 25.0, -4.84, -20.716, 65.8574543288075, 21.95248477626917, 48.0, 16.0, -0.5, 1.0, 3.0, 57593.0, 844698.5, 0.0, 1.0, 1.0, 3.0, 4.0]

<<JBI1134\_start>>:2011-06-30

2010-02:

2010-03:

FDS1841,-1.0,0,[],0,0,2.0,[2010-01-06; 2010-02-17],825237.5,1.0,0,2,

2010-04:

GMM1037,1.0,1.0,[2010-01-15],57593.0,0.0,0,[],0,0,1,0,

QSG1150,-1.0,0,[],0,0,1.0,[2010-03-04],19461.0,0.0,0,1,

验证：

User\_send\_days = 1

User\_recv\_days = 3

验证通过！

* 注意：
* 实际编写时，遇到一个需要考虑的问题：我们的假设是leave\_contacts应当与JSR成正相关，因此越高越好；对于没有leave\_contacts的用户，其邮件部分设置为0即可（因为正常情况下全部大于0）；
* 然而对于dis\_ocean/os则不同，我们的推测是距离越大JSR越小，因此这里似乎应当采用一个反比的关系，即使用1/exp(dis\_ocean/os)，这样似乎更正确一些
* 我们先按照默认处理dis\_ocean/os为0的方法，得到CERT5.2\_Train\_JS\_Feats-0.1.csv；根据实验再确定如何修改吧；

下午，基本写完了训练集的模块，即完成了用户JS\_Feat的提取以及SVM的训练，注意，这里我们使用了2010-01：2010-04的四个月CERT5.2用户JS\_Feats的五折交叉验证，以训练得到最优的参数，从而预测2010-05月份离职的用户，根据结果对于SVM预测器进行微调；然后逐个月预测后续的用户2010-06：2011-05，依次得到结果，

2018年12月8日星期六

上午12时48分

晚上采用新的“过程+对象”重写了JS\_Feat提取模块，并决定先队每个月生成一个新的JS\_Feat，先对2010-03月进行分析验证

* MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,8.0,11.0,23.0,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,MMK1532,17,17,16,22,28
* 2010-02:
* WMH1300,1.0,1.0,[2010-01-25],38275.0,0.0,0,[],0,0,1,0,
* 2010-03:
* MIB1265,1.0,1.0,[2010-03-05],26650.0,0.0,0,[],0,0,1,0,
* MMK1532,7.5,18.5,**8.0,11.0,23**,7,2010-03-02,2010-03-03,2010-03-05,2010-03-08,2010-03-16,2010-03-19,2010-03-25,

验证通过！

等2010-01的LED\_Feats统计完毕，即可

2018年12月9日星期日

下午7时39分

周六泡温泉时突然想到，为什么不单独列一个可以提取JS\_Feats的类呢？该类负责为2010-01：2011：05的每个月份生成对应的月JS\_Feat，当然，LC部分是累积特征。然后再依据

训练集多数据集合并验证：

Train:

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,1.79405672302,0.299009453836,0.266338042847,0.0443896738078,**45.0,40.0,106**.0,0.2,3.0,3.0,99311.0,69220.0,0.0,0.0,3.0,2.0,4.0,

2010-01:

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,

2010-02:

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.274789435642,0.274789435642,0.241624510319,0.241624510319,**10.0,9.0,20.0**,1.0,1.0,0.0,38275.0,0.0,0.0,0.0,1.0,0.0,1.0,

2010-03

32,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,**8.0,11.0,23.0**,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

2010-04

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,**8.0,7.0,21.0**,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

2010-05

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,1.79405672302,0.299009453836,0.266338042847,0.0443896738078,**9.0,7.0,22.0**,0.2,3.0,3.0,99311.0,69220.0,0.0,0.0,3.0,2.0,4.0,

虽然这里的2010-05月份数据因为忘记计算而采用的2010-06月份数据，但是JS\_Feats更新验证通过！



2018年12月10日星期一

上午9时48分

2018年12月第二周

继续上周的实验，今天的目标是实现一个简单的SVM Predictor，需要注意：

1. 由于我们的预测模式是：基于前N个月来预测第N+1个月，因而训练时不再使用分隔交叉验证，而是直接由训练集遍历参数空间得到最优的SVM分类器，然后输出分类器参数和得到的分类器对象；
2. 采用上个阶段训练得到的分类器对象与参数，预测下一个月的用户离职情况；
3. 根据预测结果重新训练SVM Predictor，输出新的最优分类器
4. SVM训练前必须进行scale
5. 最重要的参数是C与赐嘎玛，C越大对于非线性拟合能力越强，赐个吗越大越平滑，非线性拟合越低，对噪声越不敏感；
6. 我们筛选的指标有：Recall/FPR/Accuracy

2018年12月10日星期一

下午3时7分

中午重新整理了思路，发现，为什么不把每个月的JS\_Feats文件都设置成当月SVM可以直接进行训练呢？

于是，重写了之前的JS\_Feats提取模块，确保每个月份目录下的JS\_Feats的特征已经是截止到该月的累积和（主要涉及LED与LC特征）。

然后从2010-01:2010-02进行简单验证：

2010-01：

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,

MMK1532,7.5,18.5,10.0,6.0,20,5,2010-01-05,2010-01-07,2010-01-13,2010-01-15,2010-01-21,

2010-02：

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.274789435642,0.274789435642,0.241624510319,0.241624510319,**20.0,18.0,40.0**,1.0,1.0,0.0,38275.0,0.0,0.0,0.0,1.0,0.0,1.0,

MMK1532,7.5,18.5,**10.0,9.0,20**,8,2010-02-02,2010-02-03,2010-02-04,2010-02-09,2010-02-12,2010-02-16,2010-02-24,2010-02-25,

下午9时47分

关于面向对象编程的一点规范：

1. 一般将数据处理与分析过程尽量分开，先进行完全的数据处理，然后再进行训练分析；
2. 分析过程从最细的方式展开，后期可以将多个具体部分综合在一起；

我们开始进行验证：

Train\_JSF: (2010-01:2010-04)

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,47.0,40.0,104.0,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

Train\_Lables:

MMK1532,-1,

2010-03:

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,28.0,26.0,63.0,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

2010-04:

Validate\_JSF:

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,47.0,40.0,104.0,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

2010-05:

MMK1532,17.0,17.0,16.0,22.0,28.0,-10.648,-12.0872,0.596827474047,0.298413737024,0.254693593253,0.127346796627,47.0,40.0,104.0,1.0,2.0,0.0,64925.0,0.0,0.0,0.0,2.0,0.0,2.0,

Validate\_Lables:

MMK1532,-1,

验证通过！

再看RMB（2010-02离职）是否也正确？

RMB1821,22.0,19.0,48.0,14.0,32.0,-6.776,-11.9464,0.0,0.0,0.0,0.0,0.0,1.0,31.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,

同实际JS特征

RMB1821,22.0,19.0,48.0,14.0,32.0,-6.776,-11.9464,0.0,0.0,0.0,0.0,0.0,1.0,31.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,

通过！