顾客分组算法V3 -- 加权正态分布拟合法

1. 交易数据前期处理

a. 合并同一自然日内的所有消费行为——即每天的消费次数最高为一次
b. 将消费次数为一次的顾客单独分离作为新用户，不再纳入顾客分组计算
c. 计算消费间隔的时候需要去除店铺休业日——即间隔不包含店铺未营业的时长

2. 获取店铺预流失周期上下线

- 将店铺所有消费间隔排序提取上四分位点**I25**和下四分位点**I75**分别作为店铺预流失判断的下线和上线，目的在于用下四分位点限制消费频次过高的客户以防出现预流失节点太短，而上四分位点用于限制低频次消费的顾客防止出现预流失节点过长。
- 调节I25和I75至合适的大小。此处引入**上下线调节系数**outline_facotr，根据下线大小自动调整：

```
if I25 <= 1.5:
  adjust_factor = 4
elif 1.5 < I25 <= 2.5:
  adjust_factor = 3
elif 2.5 < I25 <= 3.5:
  adjust_factor = 2
elif 3.5 < I25 <= 4.5:
  adjust_factor = 1.5
else:
  adjust_factor = 1
```

I25 = I25 * adjust_factor; I75 = I75 * adjust_factor
注：此处的上下线是基于单点数据本身计算的，加入调节系数目的在于将上下线调节至直观上容易接受的值。之后的可能修改方向为：1.若有行业数据可直接用行业经验值进行限制。2. 上下线的调节系数分别独立

3. 计算店铺所有第k次到k+1次消费的平均时间间隔及其标准差

此处设置的k=2，3，4。思路：第2次消费的客户只能计算自身的均值而无法计算标准差，无法判断这个时间间隔的可信度，因此直接用所有顾客的第2到3次转化的时间间隔和标准差用于之后的预流失和流失区间计算。而对于第3，4次，当消费间隔变化很大的时候，自身的消费规律还是不可信，因此考虑用第3-4，4-5次店铺平局消费间隔和标准差替代，否则就是用自身的。
计算之前需要进行**异常值**处理，优化后的算法同时考虑过大和过小的异常值，伪代码如下：

```
消费间隔Ix ∈ [第k-k+1次的所有消费的间隔]
均值mean
标准差sd
if Ix ∈ [mean-1.25*sd*log2(1+mean/sd),  mean+1.25*sd*log2(1+mean/sd)]
  Ix is nomral, retained
else
  Ix is abnormal, removed
计算保留的所有Ix的均值k_mean和标准差k_sd作为第k-k+1次的店铺平均消费间隔
```

4. 计算所有顾客的自身的消费间隔均值和标准差

异常值处理如前所述。
此处，如果该顾客有异常大的值保留，记录该值X_max和去除异常值后的最大值X_max_remain。此值将用于后续分类的调整。

5. 计算所有顾客的流失和预流失节点

- 首先计算2-3,3-4,4-5次消费店铺水平的流失和预流失节点，此处引入**流失节点调节参数**churn_adjuct，用于调节标准差的区间（标准差过小会导致流失和预流失时间差距过小，因此放大）。伪代码如下：

```
if 1.5 <= k_sd < 2.5:
  churn_adjuct = 1.5
elif k_sd < 1.5:
  churn_adjuct = 2
else:
  churn_adjuct = 1
回购间隔均值K_mean
回购间隔标准差K_sd
原始预流失节点K_Tp = K_mean + churn_adjuct*K_sd*log2(1+X_mean/X_sd)
流失节点K_Tc = K_mean + 2 * churn_adjuct*K_sd*log2(1+X_mean/X_sd)
```


© Copyright Monday, Jul 4, 2016, 7:26 PM by COMPANYNAME

- 个人水平的流失预流失计算与之相同，只是将K_mean和K_sd换成自身的均值标准差。
- 根据店铺上下线调整流失预流失节点。伪代码如下：

```
店铺预流失上线I75
店铺预流失下线I25
顾客消费次数F
个人平均回购周期X_mean
个人回购周期标准差X_sd
个人预流失节点Tp
个人流失节点Tc
第k-k+1次店铺预流失节点K_Tp（k最大为4）
第k-k+1次店铺流失节点K_Tc
流失上线触发所需的预流失上线阈值CC
#--------------------------------------------------------------------------------
if F == (1 | 2): #若只有1or2次消费，直接使用第2-3次的店铺水平预流失和流失节点
  Tp = K_Tp
  Tc = K_Tc
  if Tp < I25:
    Tp = I25  # 若个人预流失节点小于店铺预流失下线，则用店铺预流失下线替换个人预流失；流失则为预流失的两倍
    Tc = 2 * I25
  elif Tp > I75
    Tp = I75 # 若个人预流失节点大于店铺预流失上线，则用店铺预流失上线替换个人预流失节点；流失则为自身不变
    if I75 > CC:
      Tc = 2 * I75
  else:
    pass  # 若个人预流失节点位于上下线之间，则用自身
#--------------------------------------------------------------------------------
elif F ∈ [3, 5]:
  if X_sd/X_mean > 0.6 + 0.1*(F-3): # F=3时此时两个间隔I_big:I_small > 4,  认为个人消费规律不可信，用3-4次店铺水平流失和预流失代替
    Tp = K_Tp # 后续操作同上
    Tc = K_Tc
    if Tp < I25:
      Tp = I25
      Tc = 2 * I25
    elif Tp > I75:
      Tp = I75
      if I75 > CC:
        Tc = 2 * I75
    else:
      pass
  else: # 消费间隔变化差异可以接受，用自身流失和预流失节点
    if Tp < I25: # 后续操作同上
      Tp = I25
      Tc = 2 * I25
    elif Tp > I75:
      Tp = I75
      if I75 > CC:
        Tc = 2 * I75
    else:
      pass
#--------------------------------------------------------------------------------
else: # F大于5次消费，完全考虑自身规律
  if Tp < I25:  # 后续操作同上
    Tp = I25
    Tc = 2 * I25
  elif Tp > I75:
    Tp = I75
    if I75 > CC:
      Tc = 2 * I75
  else:
    pass
```

6. 判断顾客分类

基于顾客的距离上次消费时间R判断顾客的分类，同时考虑**移除的异常值**对分类的修正。

```
去除的异常值X_max（一般情况下去除的异常值少于2个。若有2个，取最大值。part 4中有计算）
if R <= Tp: # 小于预流失节点
  cus_mark = active
elif R > Tc: # 大于流失节点
  cus_mark = churn
else:
  cus_mark = potential_churn
if R > Tc and X_max <= I75 and R <= X_max: # 在去掉异常值之后若顾客标志为流失，但是这个去掉的异常值在店铺预流失上线以下，此时认为
```


© Copyright Monday, Jul 4, 2016, 7:26 PM by COMPANYNAME

```
保留此值作为正常间隔也可，将流失状态调整问预流失
  cus_mark = potential_churn
if R > Tc and R < X_max_remain: #去除异常值后顾客标志位流失，但是去掉异常值后剩余区间仍有值大于R，将顾客状态从流失调整为预流失
  cus_mark = potential_churn
```

总结

此算法中原始输入的字段需要有距离上次消费时间R，消费时间；中间需要计算消费时间间隔，个人平均回购间隔和标准差，去除的异常值（最大值），第2-5次店铺平均消费间隔和标准差，店铺预流失上下线；最后计算个人和2-5次消费的预流失和流失节点，得到个人分组标志。

© Copyright Monday, Jul 4, 2016, 7:26 PM by COMPANYNAME