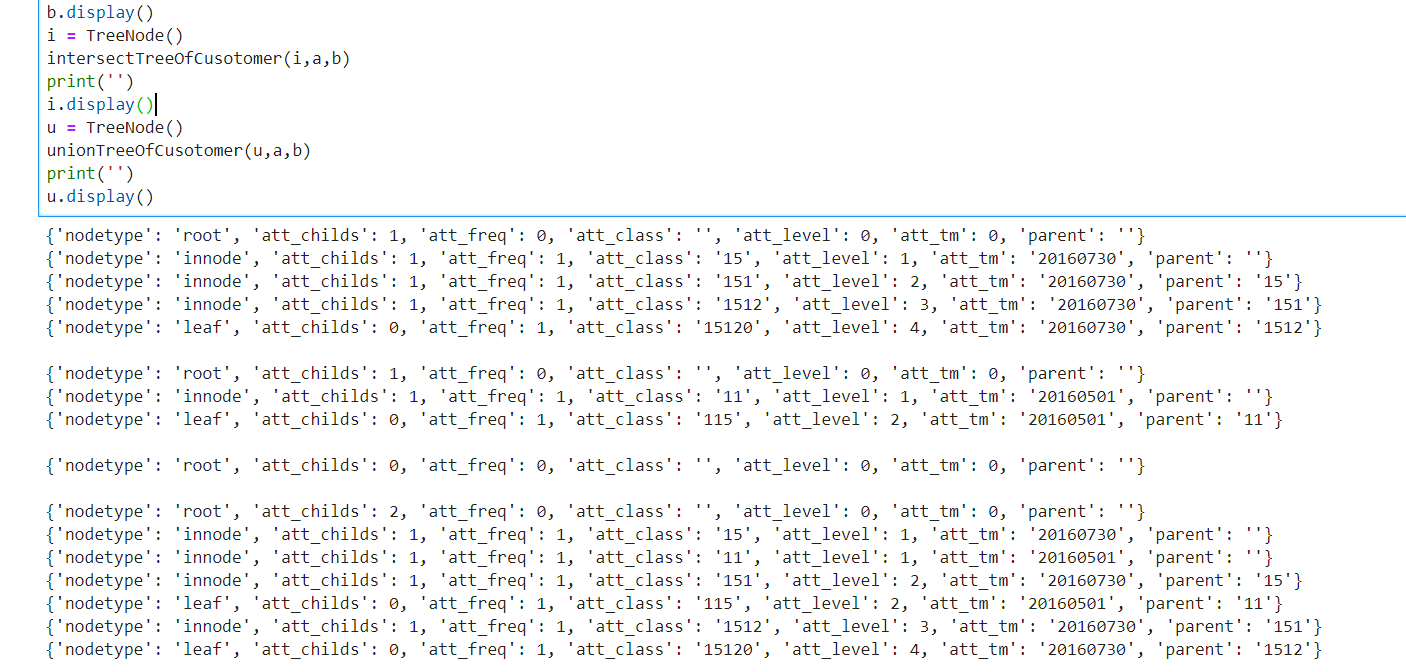
### 整个算法框架

* 对每一条交易记录建立一棵根据time level退化到指定层的树
* 对每个用户的所有交易记录所建立的树，都Union成一棵树
* 将所有用户作为初始cluster
* 循环直到没有cluster能够分裂
  + 对当前的cluster计算centroid tree
  + 选择和centroid tree距离最近的topK子树，并选择其中相互之间距离最大的两棵树，最为2-means算法的初始centroid
    - 两棵树的距离由他们的交集树、并集树之间的distance决定
  + 根据2-means算法对这个待分裂的cluster进行聚类计算
    - 当2-means算法结束时，parent cluster分成了两个child cluster
  + 计算BIC准则，如果增加就分裂，反之将这个cluster固定

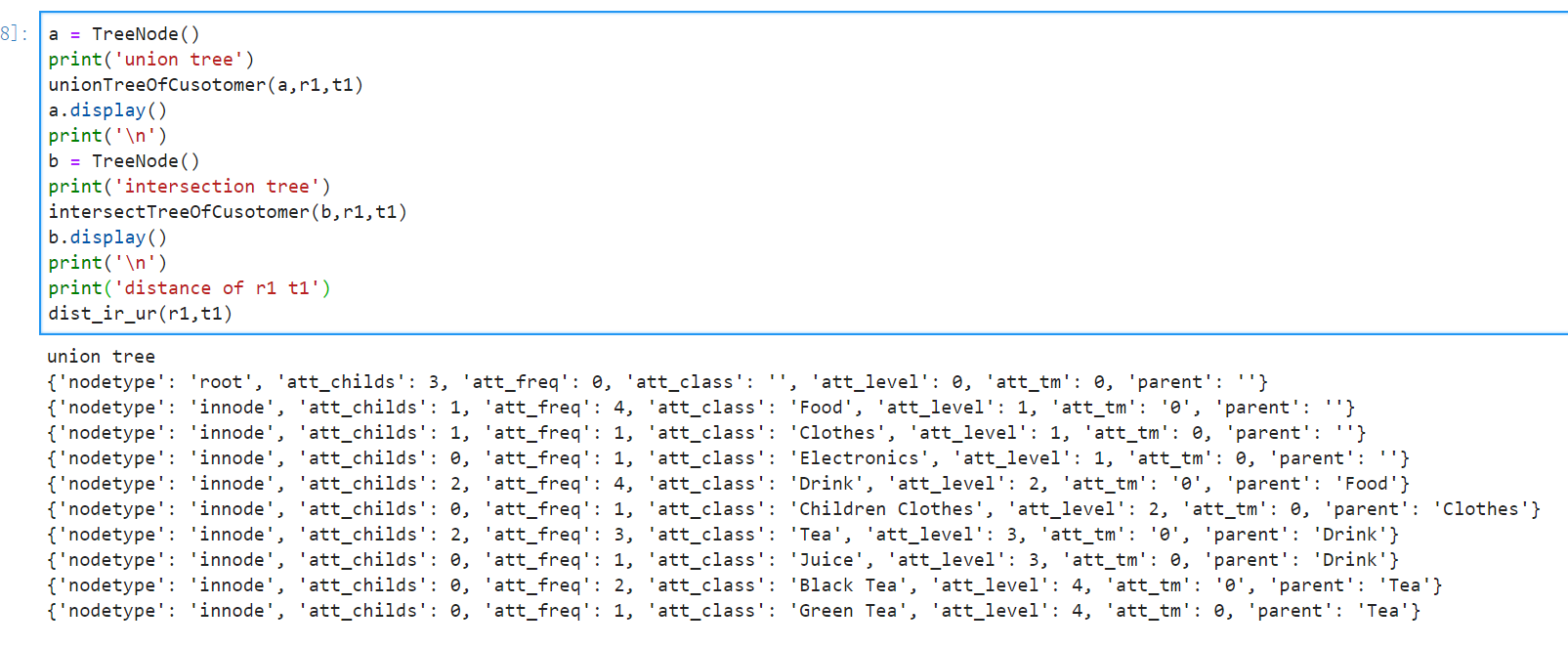
### 用户交易树的构建

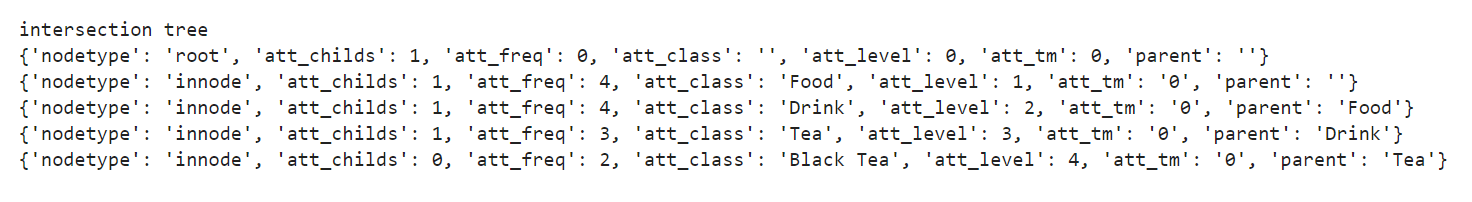


这里以两条交易为例，分别构建交易树，并计算交集数和并集树

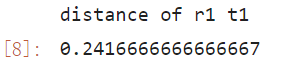
### 交集树与并集树的计算

这里用论文中的树作为测试用例，计算他们的交集树、并集树、中心树



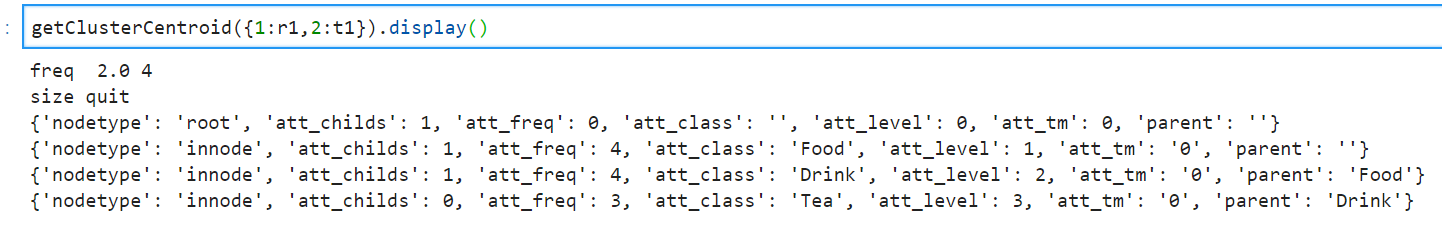


### 距离的计算



### 中心树的计算

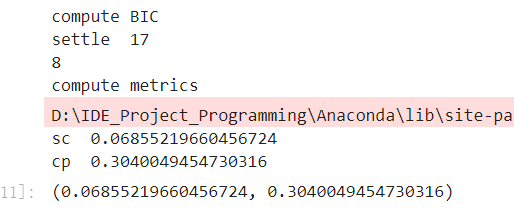
以2为freqStep，并在中心树小于平均节点数时退出



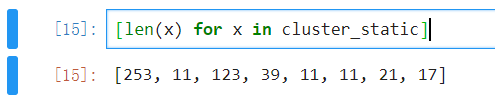
### 最终聚类的结果

解释：

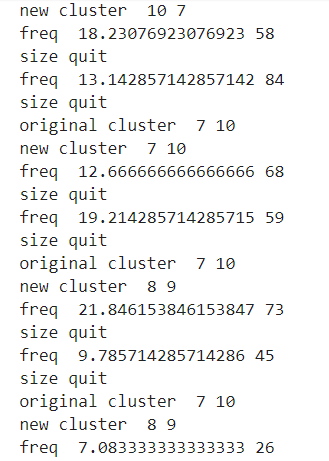
* 最终得到8个cluster
* Settle表示这个cluster不再分裂，并且大小为17
* Sc表示silhouette score
* Cp 表示 compactness score



1最终结果



2最终cluster的分布



3中间过程

中间过程的解释：

* Freq 表示 计算中心树时的 freqstep & freqEnd
* Size quit代表 中心树的计算因为结点数足够少而退出
* Original cluster 表示 分裂前的 同一个 parent cluster 的两个 child cluster 的分布
* New cluster 表示 进行 2-means 算法时的 cluster 的当前分布
* Split 代表 一个 parent cluster 分裂为两个 child cluster

讨论：

* 相对于论文中的效果差一点，sc和cp都不够好，因为这个算法跑一轮要比较长的时间，没有足够的时间去做优化处理
* 目前做到的足够的加速是，利用hash table去存储tree，但是还是有很多地方避免不了去做遍历；算法仍然可以进一步加速，但是代码量较大，需要重构很多函数
* FTCTree算法的结果比Jaccard算法好很多，比较适合这个应用场景
* 做了部分的调参，包括设置了分裂的条件，计算topK的范围等，但是没有在所有范围都测试一遍