数据挖掘大作业二：关联规则挖掘

姓名：白雪峰 学号：3120180977

# 数据说明

## 数据集

选择数据集：OaklandCrimeStatistics2011to2016

## 数据集描述

**数据集：OaklandCrimeStatistics2011to2016**

包含records-for-2011.csv到records-for-2016.csv共六个文件

属性列表：

Agency，CreateTime，Location，AreaId，Beat，Priority，IncidentTypeId，IncidentType，Description，EventNumber，ClosedTime

# 问题描述

对OaklandCrimeStatistics2011to2016数据集进行关联规则挖掘。共包含以下五个子任务：

* 对数据集进行处理，转换成适合关联规则挖掘的形式；
* 找出频繁项集；
* 导出关联规则，计算其支持度和置信度;
* 对规则进行评价，可使用Lift及其它指标,要求至少2种；
* 对挖掘结果进行可视化展示。

# 关联规则挖掘过程

## 处理数据集

第一步要将数据集处理成适合关联规则挖掘的格式，考虑到数据的完整性以及挖掘过程的复杂度和运算速度，这里选择了：

Agency

Location

Area Id

Beat

Priority

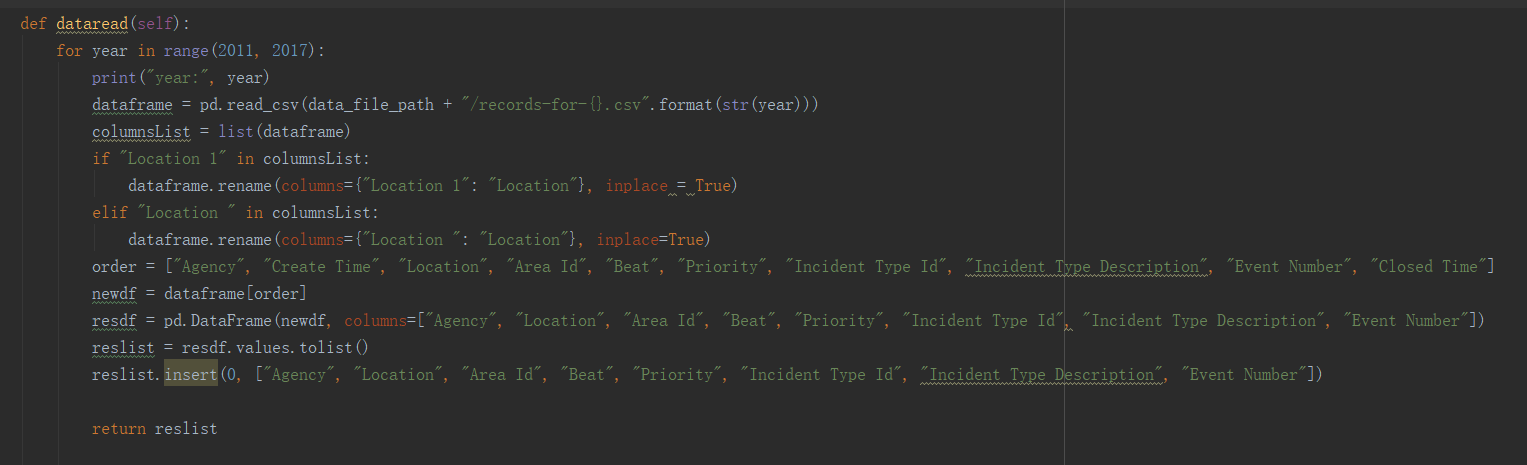
Incident Type Id

Incident Type Description

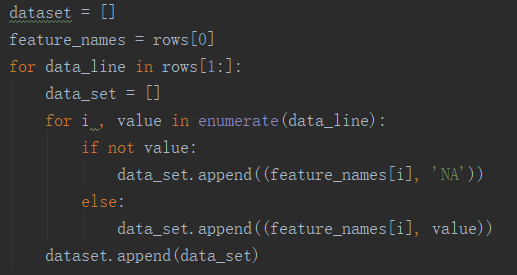
Event Number

对这些属性进行关联规则挖掘属于多维关联挖掘的问题，而单维规则分析问题仅仅针对单个属性，因此我们需要将属于不同属性的值转化为可以产生频繁项集的形式。过程如下：

首先将2011年值2016年的6个csv文件分别读取并整合为同一个DataFrame类：



然后将每一个属性名与属性值的组合表示为一个元组的形式：（属性名，属性值）。作为一个单项。并使用python 中的frozenset 类型表示项集。与set类型不同的是，frozenset 类型一经创建便不可修改，因此可以将可迭代对象转化为字典的键进行处理。

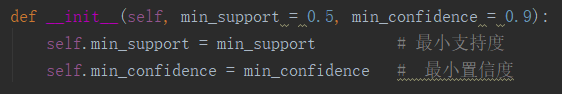


对于数据进行读取和处理的代码分别位于data\_read.py和data.py文件中。

## 产生频繁项集

使用Apriori算法在经过预处理的数据集上构建频繁项集。频繁项集表示在数据集中共现频率高的属性项的集合。关联规则的强度主要可以使用两个指标来衡量：支持度和置信度。支持度表示一个项集或者规则在所有事物中出现的频率，确定规则可以用于给定数据集的频繁程度，X->Y的支持度指的就是X和Y共同出现的概率。X->Y的置信度表示Y在包含X的事务中出现的频繁程度，也就是在出现了X的项中，Y出现的概率有多大。

首先为这两个超参数设定阙值，设定最小支持度为0.01（将支持度设得偏小是因为数据集中有几个属性的取值较为分散，如果支持度太高则会忽略掉这些属性），将最小置信度设为0.5。



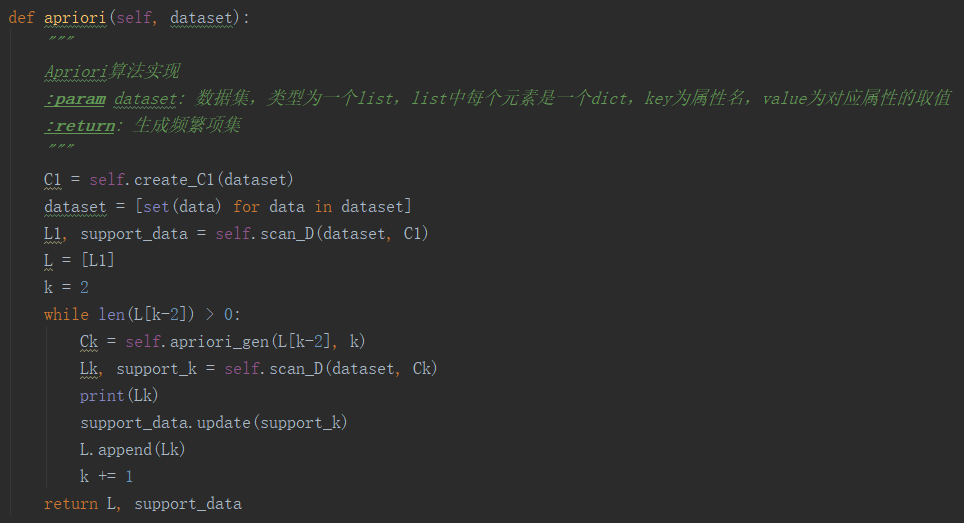
Apriori算法使用频繁项集的先验知识，使用一种称作逐层搜索的迭代方法，k项集用于探索(k+1)项集（如果事件A中包含k个元素，那么称这个事件A为k项集，并且事件A满足最小支持度阈值的事件称为频繁k项集）。首先，通过扫描事务（交易）记录，找出所有的频繁1项集，该集合记做L1，然后利用L1找频繁2项集的集合L2，L2找L3，如此下去，直到不能再找到任何频繁k项集。

Apriori算法流程如下：

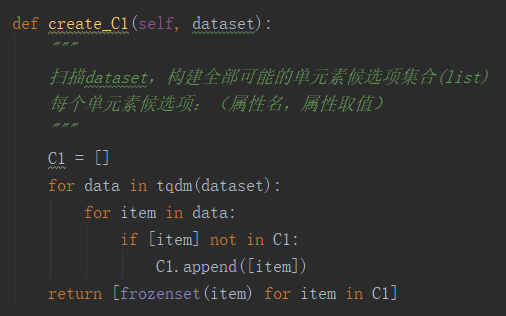
* 扫描一次数据库D；计算出各个1项集的支持度，得到频繁1项集的集合。
* 从2项集开始循环，进行由频繁k-1项集生成频繁频繁k项集。
  + 连接步：将2个只有一个项不同的属于的频集做一个（k-2）JOIN运算得到。
  + 剪枝步：因为是超集，所以可能有些元素不是频繁的。舍弃掉子集不是频繁项集即不在频繁k-1项集中的项集
  + 扫描数据库，计算2.3步中过滤后的k项集的支持度，舍弃掉支持度小于阈值的项集，生成频繁k项集。
* 当生成的频繁k项集中只有一个项集时循环结束

Apriori 算法的实现位于“associasion.py”，程序代码如下：

* Apriori 主函数：



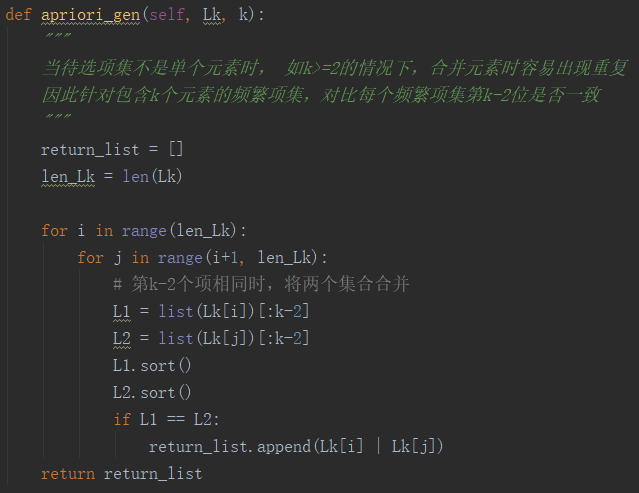
* create\_C1()函数用于生成初始的单个项项集集合：



* Scan\_D()函数用于扫描项集集合，并过滤掉小于最小支持度的项集：



* Apriori\_gen()函数用于非重复地合并两个项集：



## 导出关联规则

使用上面基于Apriori算法产生的频繁项集，进行强关联规则的挖掘。过程如下：

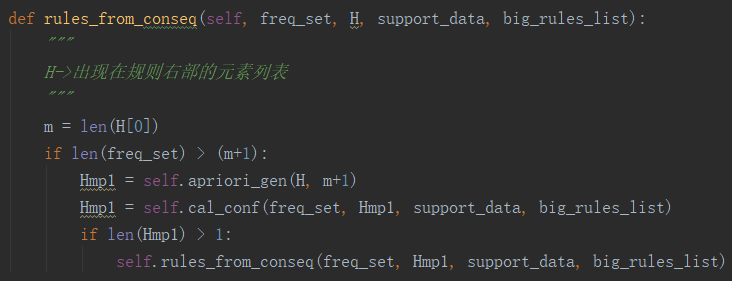
* 根据选定的频繁项集，找到它所有的非空子集。然后建立规则列表。
* 强关联规则需要满足最小支持度和最小置信度，对每一条规则计算指标：
  + 支持度：
* 置信度：
* 对每一条规则还需要计算提升度指标：

表示“包含 A 的事务中同时包含 B 的事务的比例”与“包含 B 的事务的比例”的比值。用来判断规则X ⇒ Y 中的X 和Y 是否独立，如果独立，那么这个规则是无效的。提升度反映了关联规则中的 A 与 B 的相关性，提升度 > 1 且越高表明正相关性越高，提升度 < 1 且越低表明负相关性越高，提升度 = 1 表明没有相关性。但是在具体的应用之中，我们认为提升度 > 3 才算作值得认可的关联。

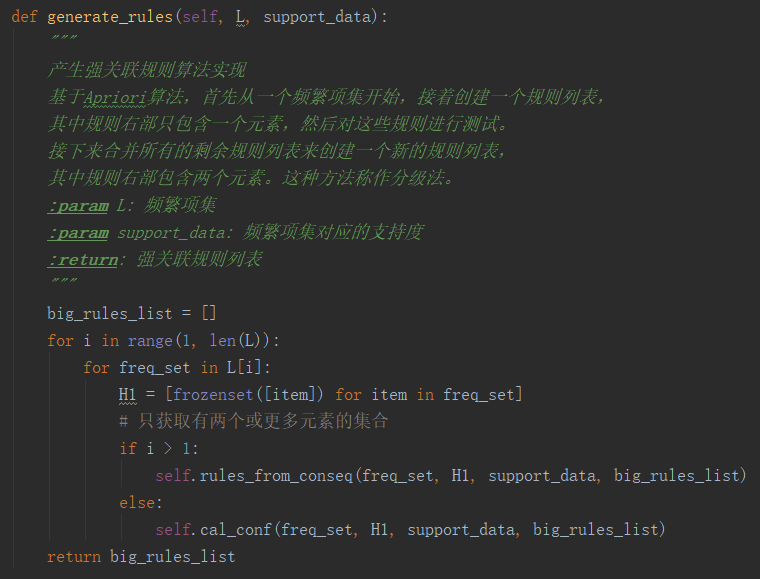
* 根据以上指标，找到所有可能的关联规则。

产生强关联规则的算法同样位于“associasion.py”的Associasion 类中，程序代码如下：

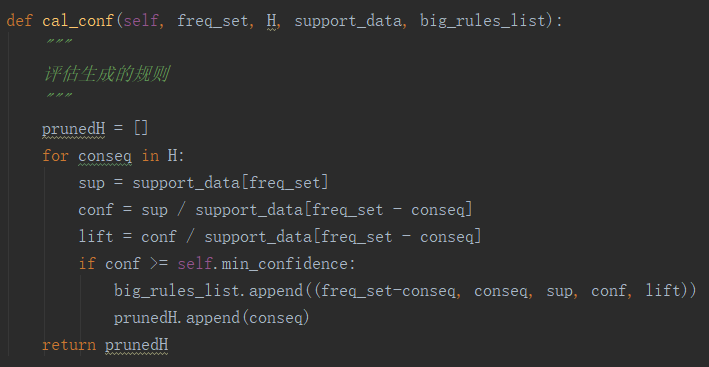
* Rules\_from\_conseq()函数，用于递归地产生规则右部的结果项集：



* Generate\_rules()函数，用于产生强关联规则：

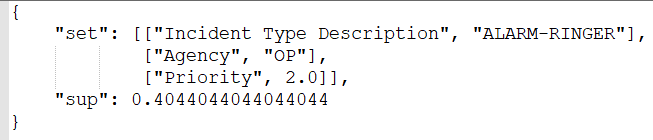


* Cal\_conf()函数，用于评价生成的规则，并计算支持度、置信度、lift 指标：

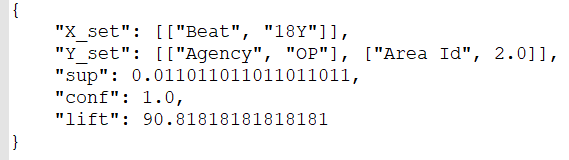


## 挖掘结果及分析

挖掘出的频繁项集按照每一项一行的格式存储在“./results/freq\_set.json”文件下，并且将频繁项集按照支持度从大到小进行排列，以下图为例说明输出结果格式：



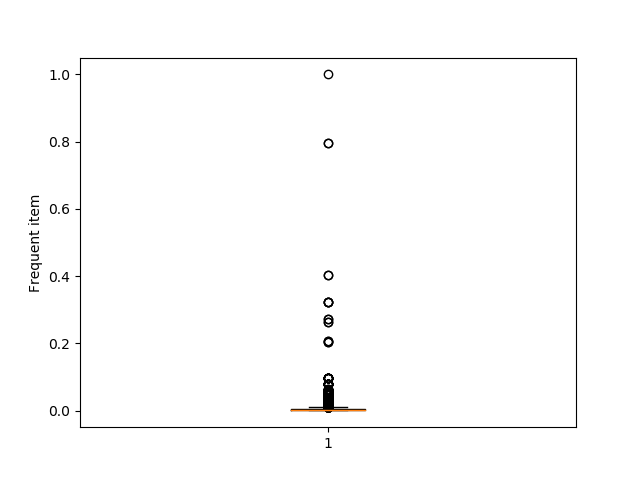
挖掘出的符合条件的关联规则



通过对最终挖掘得到的关联规则进行分析我们可以得到，“Beat”属性与“Agency”属性的属性关联度极高，并且“Area Id-> Incident Type Id”规则的置信度极高，这说明大部分犯罪事件的类型与发生的地点有着很大的联系，犯罪事件有很强的地域特征。

# 结果可视化

对频繁项集结果做可视化



对关联规则挖掘结果做可视化

