# bii分析报告

1. 情景分析

本题要求在前两题的基础上，利用sldat这一时间字段，设计Sequential FP算法，进行频繁集挖掘。

1. 代码说明

sequentialFP.py文件是将序列FP算法集成的一个模块文件，将所有的过程都封装到了函数fpGrouth()中，只需传入数据字典、最小阀值两个参数即可，返回是当前阀值下的频繁集。

bii\_model.py是将数据处理、算法调用等函数集成的一个模块，只需要调用fp\_pluno()、fp\_dptno()、fp\_bndno()三个函数即可。

bii\_pluno.py、bii\_dptno.py、bii\_bndno.py分别是对三种item进行频繁集挖掘的代码，主要完成数据的整理和调用FP算法。

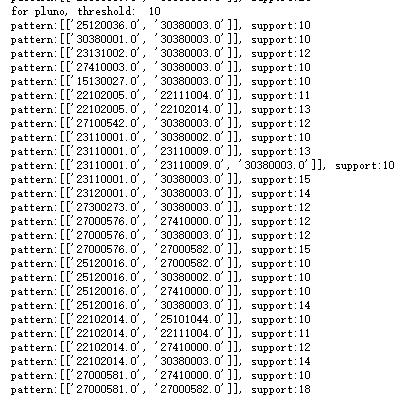
考虑到这是对购物记录做频繁项集的挖掘，我只将含有两个以上item的项集作为输出，只含有一个item的项集并没有什么意义。

1. 运行结果

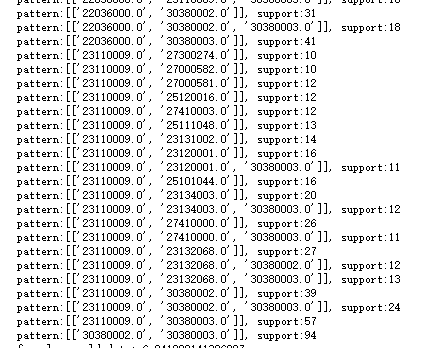
由于作业要求分别取阀值为2，4，6，8，10，所以程序的输出比较长，无法一次截取，所以分别取得到的结果的一部分进行分析：

* 1. Pluno商品

对于旧数据trade，有：



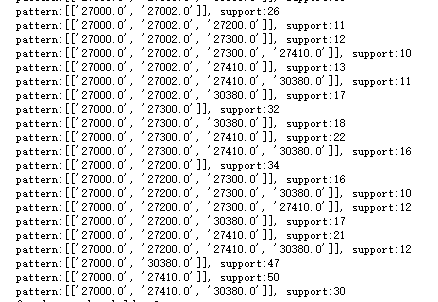
对于新数据trade\_new，有：



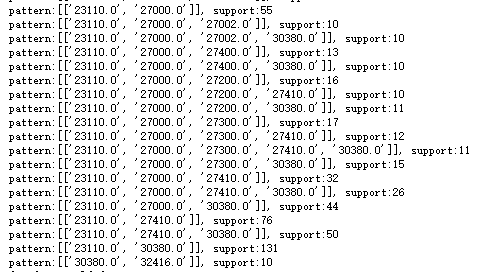
同样的，无论是新数据还是旧数据，所得到的结果和aii基本一致。

* 1. Dptno商品类型

旧数据：



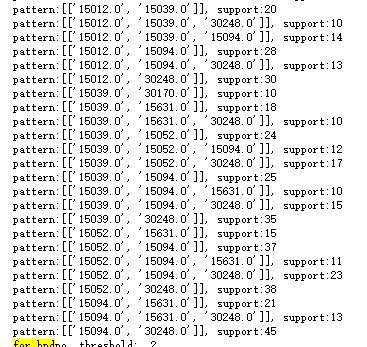
对于新数据：



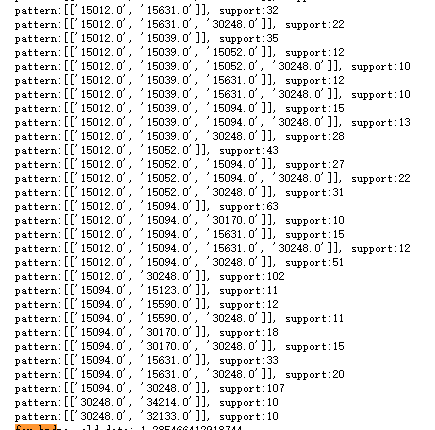
这是取自阀值为10的时候部分结果，对于dptno得到的结果也和aii一样。

* 1. Bndno品牌：

旧数据：

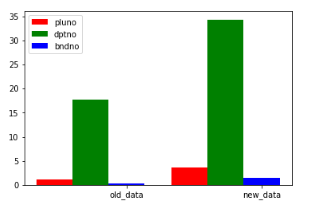
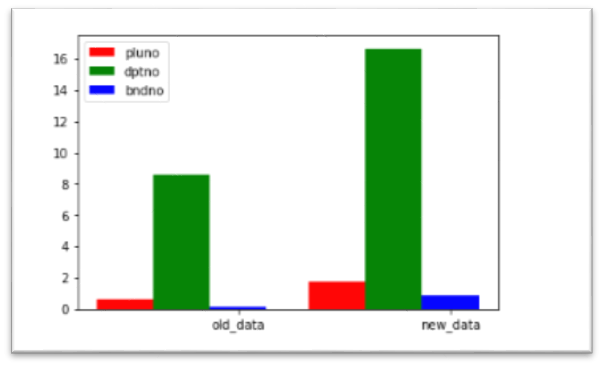


新数据：



和pluno,dptno的结果一样，bndno新旧数据的结果都和ai一致。

* 1. 性能对比



左图是aii的结果，右图是本题的结果。无论使用哪个字段，两个算法所得到的结果都惊人的一致，我觉得这是由于在aii中对数据做预处理的时候已经将数据整理成序列化的了。对于。特别的，对于字段dptno，两个算法所需的时间都远远大于另外两个字段，而且Sequential FP明显需要画更多的时间，这也是由于PrefixSpan算法在数据过大时变得缓慢的缺陷。综合考虑到三个字段使用bndno做频繁集挖掘最佳。