数据挖掘第二次作业项目文档(c)

学号：1552635　　姓名：胡嘉鑫

1. 项目说明

1.1 问题描述

根据前两问，我对 trade.csv (trade_new.csv) 文件进行了用户购买记录的频繁项集的挖掘。经过挖掘可以发现，threshold 值越大，挖掘出的频繁集个数越多，那么预测某用户将来可能购买商品的概率也就越大。

而根据前两问的性能分析可以看出，对 vipno 进行分组的话得到的频繁项集要远远多于对 uid 进行分组后得到的频繁项集，因此选择利用 vipno 进行分组后得到的频繁项集进行预测更为合适。另外，由于 pluno、bndno 和 dptno 三个属性中只有 pluno 细致到了商品的编号，那么如果对 pluno 进行预测的话意义更大。

fp-growth 算法更适合做预测还是 PrefixSpan 算法更加适合，将进行比较以后得出结论。

同样，将对候选的预测集个数、minsupport 的最佳选取做探究。

1.2 算法介绍

我的预测算法是：

1. 将 trade.csv 中的数据先根据 vipno 分组

2. 将分组后的数据分别提取前 60% 作为生成频繁项集的训练集，再分别提取后 40%作为判断预测是否准确的测试集。

3. 分别选取不同的算法{fp-growth，PrefixSpan}，不同的属性{pluno，bndno，dptno}对测试集进行挖掘。

4. 提取所有存在的 vipno，针对每个 vipno 对应的训练集中出现的数据进行频繁集的查找，选出出现次数前 5 的项目作为预测集——这个用户在 test 集合中可能会购买的项目。

5. 将生成的预测集和测试集根据 vipno 一一对应起来，判断预测成功的个数，然后得出预测准确率

1.3 数据说明

分别根据 FP-growth 或者 PrefixSpan 生成不同类型的 transaction：前者不考虑先后次序，后者考虑数据的先后次序。

数据的种类是一样的：

- uid：订单编号
- sldat：购买时间
- pluno：商品编号
- dptno：商品类型编号
- bndno：品牌编号

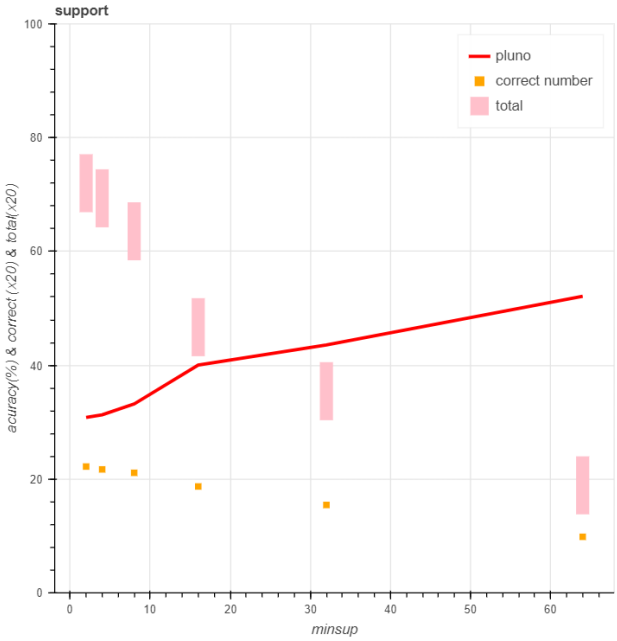
2. 代码部分

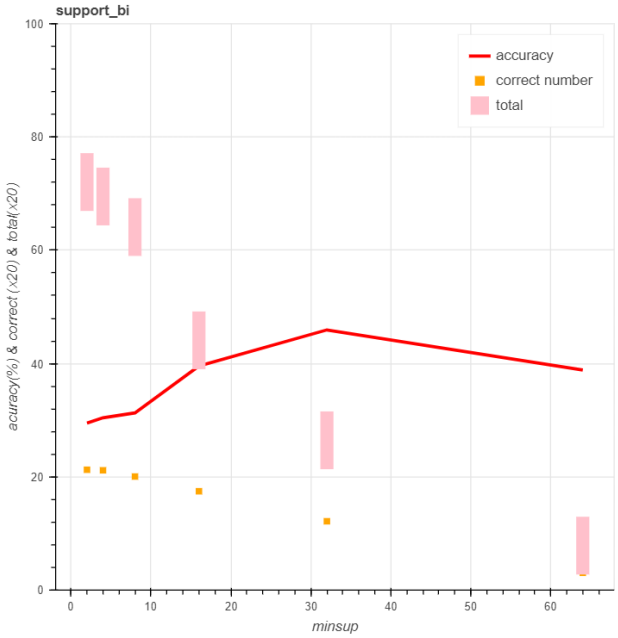
2.1 代码说明

- anticipate.py 文件是预测文件，其中含有 Anticipate 类。该类下的 anticipate()函数可以预测频繁项集挖掘的准确性。其中含有的参数是：
    - property：选择测试的 item 项
    - min_support：最小阈值
    - k：用来预测的项目的个数
    - func：测试的方法函数名（ai，aii，bi 或者 bii）
    - file：选择测试的文件
- 以下是文件内其它主要函数及作用：
    - get_data(file)：根据文件类型获得数据
    - get_test_by_property()：根据 property 选择测试集
    - show_information()：显示预测过后的信息

2.2 代码运行截图

1. 首先，选出合适的 property 和 min support 作为预测的选项值和阈值。以 ai 为待测函数。由图（右图）可以看出，随着 minsup 逐渐增高，其准确率也不断升高，原因有二：
    - Min support 增加，对应频繁项集出现的次数也就增加，意味着频繁集中 item 的相关性更强，预测的准确率也就越高。
    - 随着 minsup 的增长，生成的预测集合的个数和正确命中的个数会减少，因此每次命中后对总体正确率的提升的权重就大。



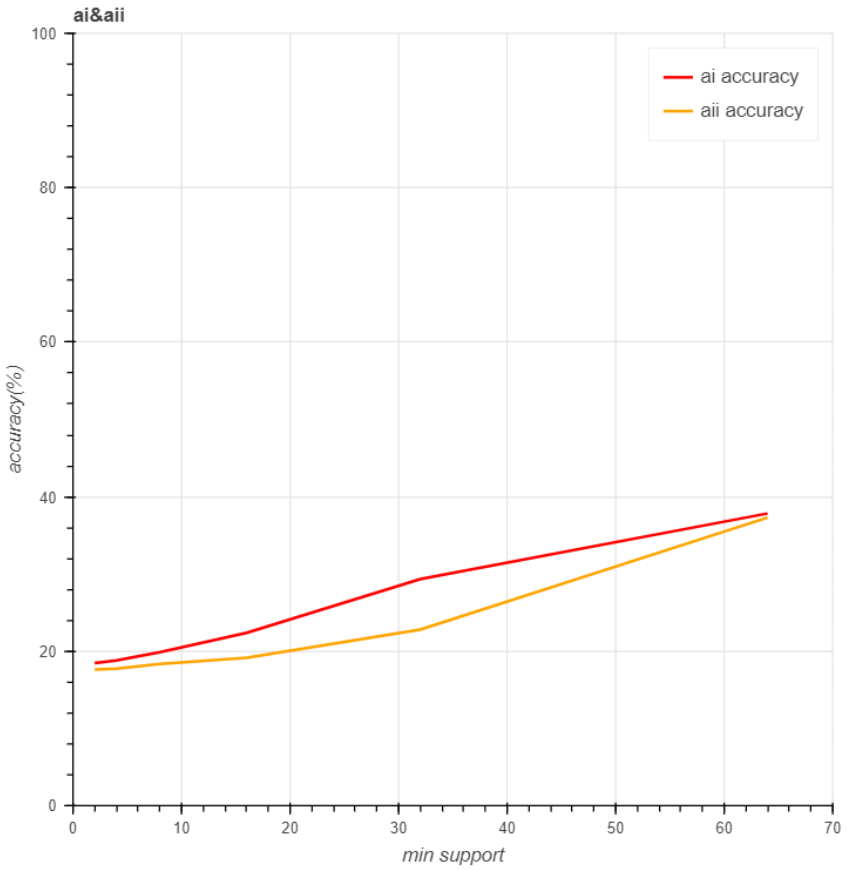


同样，对 bi 进行相同的预测，可以得到右图

由图可以看出，随着 minsup 的逐渐增大，其预测的效率在 minsup < 32 的时候不断升高，但随着 minsup 进一步扩大，其预测值不升反降。其原因是：数据量太少，具有较强的偶然性，因此测算出的预测值就会变化幅度大。

结合上两幅图的信息，在选择合适的 min sup 时，既不能让正确率太低，也不能让集合的个数太少。因此，我选择折中的 16，而且综合比较而言 PrefixSpan 的预测结果比 FP-growth 的预测结果微微偏差一些，所以选择 FP-growth 作为预测算法。

2. 接下来选择按照 vipno 分组和按照 uid 分组哪个较优：



由图可以看出，在 min_support 很小或者很大的时候二者准确率相差不多，但 min support 在 30 – 40 间取的时候，其差别较大，其中 ai 的准确率更好一些。这是因为：ai 是根据 uid 来取的，更关注于每一次订单之间的联系。而用户的所有购买记录虽然生成的频繁项集较多，但也具有订单之间种类的差别，其偶然性较大。

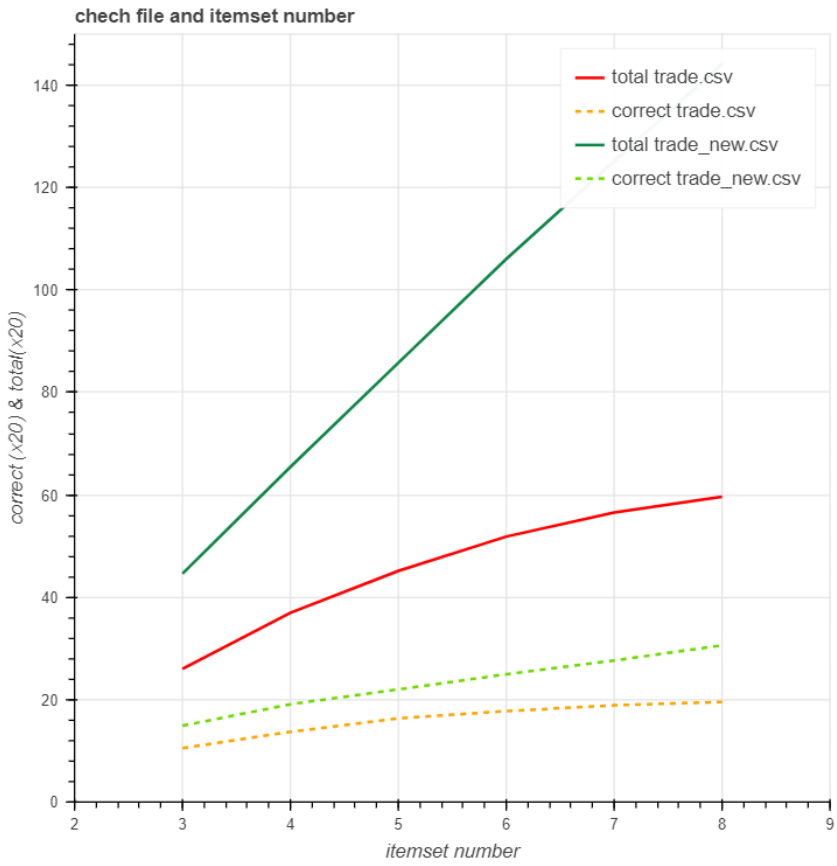
因此，选择按照 uid 进行分组的方式更符合数据挖掘的要求。

3. 下面选择合适的数据集做为预测。合适的数据集应该具有两个特点：

1）数据量大

2）预测的结果在容忍的范围之内

下面对两个数据集跑出的结果进行对比：



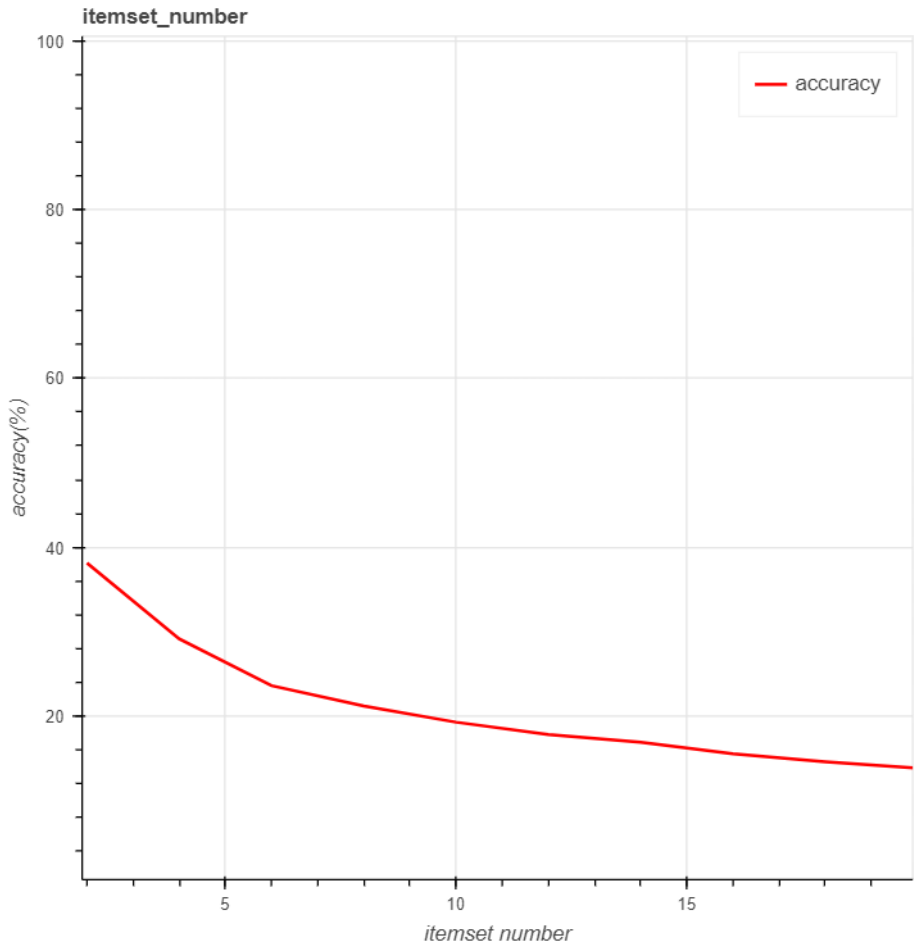
根据图可以看出两点：

1）trade_new.csv 的数据集较大，其生成的频繁项集预测集的个数也就更多

2）trade_new.csv 的预测效果并没有 trade.csv 好，当然这也是正常的，因为随着预测集中含有项目个数不断增多，其出现偏差的可能性较高，更加符合实际的预测值。

其实上述两点就常理而言也是可以理解的，而且因为 trade_new.csv 的预测准确率大概在 20%上下，是可以接受的预测概率值，因此可以选 trade_new.csv 作为生成频繁项集的测试集合。

4. 下面选择合适的 itemset number 作为预测的数字：

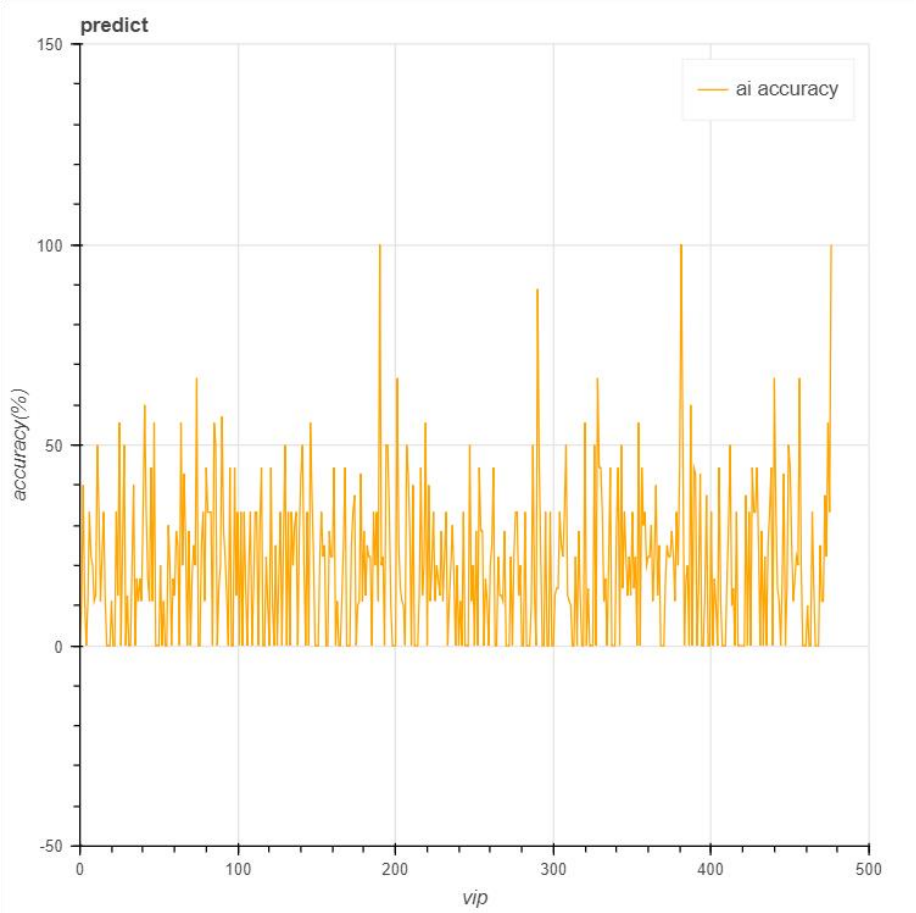


可以看出来，随着候选集合个数逐渐增加，其命中率不断降低。

这是因为用户本身购买的商品具有偶然性，候选集合中元素个数越多，出错的可能性就越大；而且每个用户购买的商品数量都是一定的，太多个数的预测集在命中用户购买记录之后继续增加只会使命中概率降低。

因此，即为了保持不错的预测效率，又为了不让预测的个数过少使得预测出来的结果不够使用，应选取折中的方法：选择 10 比较合适

因此得出结论：最合适的预测方法是：选择 trade_new.csv 做数据集，min_support 为 16，10 为预测集合个数，fp_growth 作为预测算法，按照 uid 进行分组来做预测，得到的预测集是较优的预测集合。下面就按照这样的条件进行以西预测：

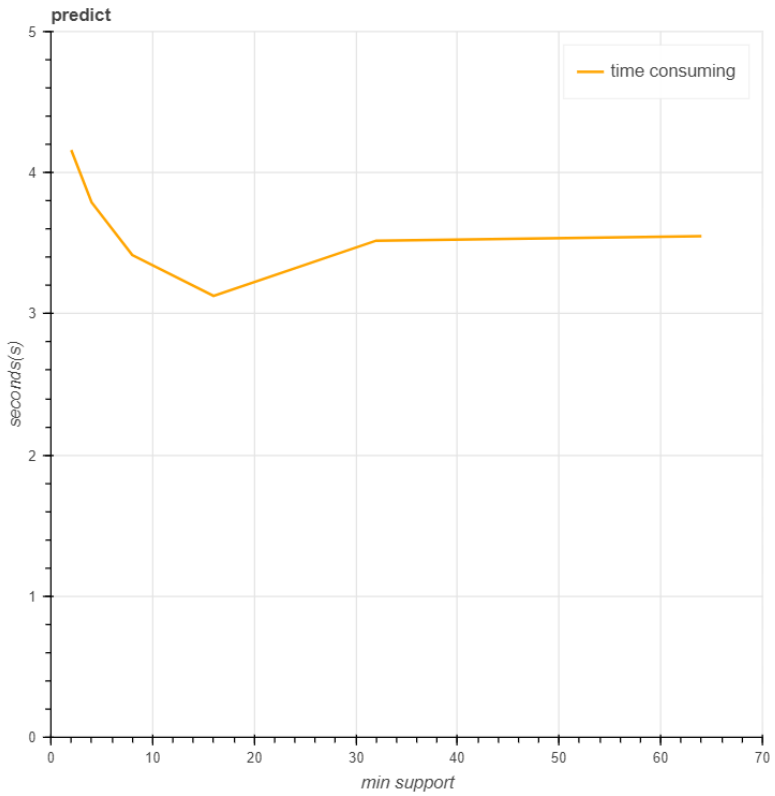



```
the total anticipated data is: 3030
the correct anticipated data number is: 676
the accuracy is: 22.31%
using time: 2.9188s
```

根据上述图片可以看出，预测的准确率为 22.31%，用时 2.9188s，而且根据折线图显示，绝大部分的数据都是 10% ~ 30% 的命中概率，比较稳定。

3. 运行效率分析

以下是根据 min support 值的变化做出的性能分析图表：



由图可以看出，min support 与整个代码的预测效率有一定关系但关系不大。在 minsupport 较小的时候，效率随着 min support 的增大而增大，这是因为生成频繁集的个数变少，所用的时间在缩短，预测效率提升；但随着 min_support 超过 16 左右后，效率开始趋于稳定，这是因为生成的频繁集个数趋于稳定，其效率也就趋于稳定。

由此可见，预测的效率高低很大程度上取决于生成频繁项集时的效率。