数据挖掘作业2　　关联规则挖掘

姓名：董永银

学号：2120171007

日期：2018.4.23

Building_Permits 数据的关联规则分析报告

1.数据源

选取数据挖掘作业 1 中的数据集 2（Building_Permits.csv）进行关联规则挖掘。

2.分析过程

1、对数据进行处理

```
start = time.clock() #计时开始
print(u'\n转换原始数据至0-1矩阵...')
ct = lambda x : pd.Series(1, index = x[pd.notnull(x)]) #转换0-1矩阵的过渡函数
b = list(map(ct, data.as_matrix())) #用map方式执行
data = pd.DataFrame([b]).fillna(0) #实现矩阵转换，空值用0填充
end = time.clock() #计时结束
print(u'\n转换完毕，用时：%0.2f秒' %(end-start))
del b #删除中间变量b，节省内存
```

首先数据方面，要把数据转换为 0，1 的形式，1 代表有，0 代表无。

2、找频繁项集

Apriori 算法-频繁项集挖掘

Apriori 算法是一种发现频繁项集的基本算法。其使用一种称

为逐层搜索的迭代方法，其中 k 项集用于探索（k+1)项集。首先，通过扫描数据库，累计每个项的计数，并收集满足最小支持度的项，找出频繁 1 项集的集合。该集合记为 L1。然后。使用 L1 找出频繁 2 项集的集合 L2，使用 L2 找出 L3，如此下去，直到不能再找到频繁 k 项集。为了压缩搜索空间，提出先验性质：频繁项集的所有非空子集也一定是频繁的。（如果一个集合不能通过测试，则他的所有超集也都不能通过相同的测试）。

```
# 频繁规则的产生
# 用于实现L_{k-1}到C_k的连接
def connect_string(x, ms):
    x = list(map(lambda i:sorted(i.split(ms)), x))
    l = len(x[0])
    r = []
    for i in range(len(x)):
        for j in range(i,len(x)):
            if x[i][:l-1] == x[j][:l-1] and x[i][l-1] != x[j][l-1]:
                r.append(x[i][:l-1]+sorted([x[j][l-1],x[i][l-1]]))
    return r
```

x 传入的参数是 L_{K-1},即所有频繁 K-1 项集的集合。

循环的思路：两两判断，如果两个项集的前 K-1 项相同，但第 K 项不同，则把两者拼接起来，组成备选项集，继而形成 C_K。

3、导出关联规则，计算其支持度和置信度

```
def find_rule(data, support, confidence):
    result = pd.DataFrame(index=['support', 'confidence'])  # 定义输出结果

    support_series = round(1.0 * data.sum() / len(data), 6)  # 支持度序列

    column = list(support_series[support_series > support].index)  # 初步根据支持度筛选
```

d 为数据，support 为支持度阈值，confiden 为置信度阈值。

（1）关联规则的产生：

```
column_new = []

for i in column:
    i = i.split('-')
    for j in range(len(i)):
        column_new.append(i[:j] + i[j + 1:] + i[j:j+1])

# 先行定义置信度序列，节约计算时间
cofidence_series = pd.Series(index=['-'.join(i) for i in column_new])

for i in column_new:  # 计算置信度序列
    cofidence_series['-'.join(i)] = support_series['-'.join(sorted(i))] /
    support_series['-'.join(i[:len(i) - 1])]
```

（2）支持度

```
# 计算占比（支持度）
support_series_new = round(data_new[['-'.join(i) for i in column]].sum() / len(data),6)

# 通过支持度剪枝
column = list(support_series_new[support_series_new > support].index)
support_series = support_series.append(support_series_new)
```

（3）置信度筛选

```
for i in cofidence_series[cofidence_series > confidence].index:  # 置信度筛选
    result[i] = 0.0
    result[i]['confidence'] = cofidence_series[i]
    result[i]['support'] = support_series['-'.join(sorted(i.split('-')))]
```

4、对规则进行评价

运行时间太长，没有计算出结果