# 数据挖掘互评作业 2：频繁模式与关联规则挖掘

姓名：曹健　　学号：3120190978

一．数据

1. 数据集选择：**Oakland Crime Statistics 2011 to 2016**
2. 数据集描述：

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

   属性列表：

   Agency，Create Time，Location，Area Id，Beat，Priority，Incident Type Id，Incident Type，Description，Event Number，Closed Time

二．问题

- 对数据集进行处理，转换成适合进行关联规则挖掘的形式；
- 找出频繁模式；
- 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、卡方和其它教材中提及的指标，至少 2 种；
- 对挖掘结果进行分析；
- 可视化展示。

三、关联规则挖掘

1. 数据集处理

将数据集处理成适合关联规则挖掘的格式，对 Agency、Location、Area Id、Beat、Priority、Incident Type Id、Incident Type Description、Event Number 这些属性进行关联规则挖掘。

将不同属性值转化为可以产生频繁项集的形式：

```
def dataread(self):
    for year in range(2011, 2017):
        print("year:", year)
        dataframe = pd.read_csv(data_file_path + "/records-for-{}.csv".format(str(year)))
        columnsList = list(dataframe)
        if "Location 1" in columnsList:
            dataframe.rename(columns={"Location 1": "Location"}, inplace = True)
        elif "Location " in columnsList:
            dataframe.rename(columns={"Location ": "Location"}, inplace=True)
        order = ["Agency", "Create Time", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Incident Type Description",
        newdf = dataframe[order]
        resdf = pd.DataFrame(newdf, columns=["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Incident Type
        reslist = resdf.values.tolist()
        reslist.insert(0, ["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Incident Type Description", "Eve

        return reslist
```

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。

```
# 将数据转为数据字典存储

dataset = []
feature_names = rows[0]
for data_line in rows[1:]:
    data_set = []
    for i , value in enumerate(data_line):
        if not value:
            data_set.append((feature_names[i], 'NA'))
        else:
            data_set.append((feature_names[i], value))
    dataset.append(data_set)
```

2. 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。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。

首先为这两个超参数设定阈值，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 算法实现代码细节见 associasion.py 文件**

3. 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。
- 根据以上指标，找到所有可能的关联规则

**实现代码细节见 associasion.py 文件**

4. 结果分析

**产生的频繁项集结果见 results/freq_set.json 文件**

**产生的关联规则结果见 results/rules.json 文件**

results > {} freq_set.json > ...

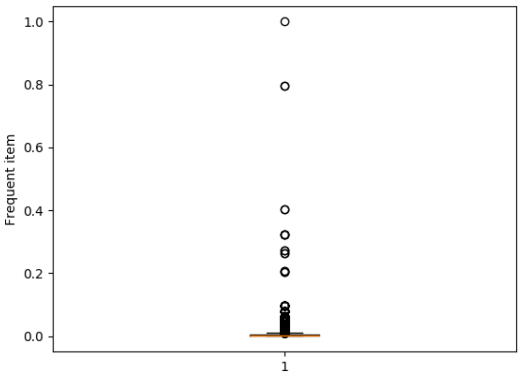

```
{"set": [["Agency", "OP"]], "sup": 1.0}
{"set": [["Priority", 2.0]], "sup": 0.7947947947947948}
{"set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.7947947947947948}
{"set": [["Area Id", 3.0]], "sup": 0.4044044044044044}
{"set": [["Area Id", 3.0], ["Agency", "OP"]], "sup": 0.4044044044044044}
{"set": [["Area Id", 3.0], ["Priority", 2.0]], "sup": 0.32432432432432434}
{"set": [["Area Id", 3.0], ["Agency", "OP"], ["Priority", 2.0]], "sup": 0.32432432432432434}
{"set": [["Area Id", 1.0]], "sup": 0.3233233233233233}
{"set": [["Agency", "OP"], ["Area Id", 1.0]], "sup": 0.3233233233233233}
{"set": [["Area Id", 2.0]], "sup": 0.2722722722722723}
```

```
sults > {} rules.json > ...
1  [{"X_set": [["Incident Type Id", "WELCK"]], "Y_set": [["Agency", "OP"]], "sup": 0.012012012012012012, "conf": 1.0, "lift": 83.25},
2  {"X_set": [["Incident Type Description", "WELFARE CHECK -- CHE"]], "Y_set": [["Agency", "OP"]], "sup": 0.012012012012012012, "conf": 1.0
3  {"X_set": [["Incident Type Description", "WELFARE CHECK -- CHE"]], "Y_set": [["Incident Type Id", "WELCK"]], "sup": 0.012012012012012012
4  {"X_set": [["Incident Type Id", "WELCK"]], "Y_set": [["Incident Type Description", "WELFARE CHECK -- CHE"]], "sup": 0.012012012012012012
5  {"X_set": [["Beat", "31X"]], "Y_set": [["Agency", "OP"]], "sup": 0.011011011011011011, "conf": 1.0, "lift": 90.81818181818181}
6  {"X_set": [["Beat", "31X"]], "Y_set": [["Priority", 2.0]], "sup": 0.011011011011011011, "conf": 1.0, "lift": 90.81818181818181}
7  {"X_set": [["Beat", "31X"]], "Y_set": [["Area Id", 3.0]], "sup": 0.011011011011011011, "conf": 1.0, "lift": 90.81818181818181}
8  {"X_set": [["Location", " FOOTHILL BLVD "]], "Y_set": [["Agency", "OP"]], "sup": 0.011011011011011011, "conf": 1.0, "lift": 90.818181818
9  {"X_set": [["Incident Type Id", "975"]], "Y_set": [["Agency", "OP"]], "sup": 0.01001001001001001, "conf": 1.0, "lift": 99.9}
10 {"X_set": [["Incident Type Id", "975"]], "Y_set": [["Priority", 2.0]], "sup": 0.01001001001001001, "conf": 1.0, "lift": 99.9}
```


上图中是对应的前十条结果。

5. 结果可视化

对频繁项集结果做可视化：




对关联规则结果做可视化：