频繁模式与关联规则挖掘

姓名：惠子杨 学号：3120201028

一、数据分析要求

1.对数据集进行处理，转换成适合进行关联规则挖掘的形式

2.找出频繁模式

3.导出关联规则，计算其支持度和置信度

4.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

5.对挖掘结果进行分析

6.可视化展示

二、数据说明

选择 Oakland Crime Statistics 2011 to 2016 数据集，详情见 GitHub：https://github.com/Hzy66666/Data_mining2。

三、数据分析过程

1.数据预处理

在 Oakland Crime Statistics 2011 to 2016 数据集中，一共包含 6 个数据文件，分别为 records-for-2011.csv、records-for-2012.csv、records-for-2013.csv 、 records-for-2014.csv 、 records-for-2015.csv 、 records-for-2016.csv。通过上一次作业中对此数据集的分析得知，每个小数据都有十条属性，除了 Location 这条属性之外其余属性均相同，因此将 2012 年、2013 年和 2014 年的“Location ”和“Location 1”属性处理为“Location”。

另外，通过观察，选择 Agency、Location、Area Id、Beat、Incident Type Id、Incident Type Description、Event Number 这几条属性进行处理。对于部分数据存在缺失值的情况，使用上次预处理中剔除含有缺失值的行的方法，得到 859906 条数据。

2.生成频繁项集

（1）Apriori 算法

对于频繁模式的挖掘，本次实验中所使用的算法是 Apriori 算法。Apriori 算法是第一个关联规则挖掘算法，也是最经典的算法。它利用逐层搜索的迭代方法找出数据库中项集的关系，以形成规则，其过程由连接（类矩阵运算）与剪枝（去掉那些没必要的中间结果）组成。该算法中项集的概念即为项的集合。包含 K 个项的集合为 k 项集。项集出现的频率是包含项集的事务数，称为项集的频率。如果某项集满足最小支持度，则称它为频繁项集。

该算法的基本思想是：首先找出所有的频集，这些项集出现的频繁性至少和预定义的最小支持度一样。然后由频集产生强关联规则，这些规则必须满足最小支持度和最小置信度。然后使用第 1 步找到的频集产生期望的规则，产生只包含集合的项的所有规则，其中每一条规则的右部只有一项，这里采用的是中规则的定义。一旦这些规则被生成，那么只有那些大于用户给定的最小置信度的规则才被留下来。为了生成所有频集，使用了递归的方法。

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: $C_k = \text{apriori-gen}(F_{k-1})$. {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: $C_t = \text{subset}(C_k, t)$. {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

基于以上对 Apriori 算法的介绍，本实验规定最小支持度为 0.1，最小置信度为 0.5。

（2）算法流程

生成单元素候选项集 C1：

```python
def C1_generation(dataset):
    C1 = []
    progress = ProgressBar()
    for data in progress(dataset):
        for item in data:
            if [item] not in C1:
                C1.append([item])
    return [frozenset(item) for item in C1]
```

过滤支持度低于阈值的项集：

```python
def Ck_low_support_filtering(dataset, Ck):
    Ck_count = dict()
    for data in dataset:
        for cand in Ck:
            if cand.issubset(data):
                if cand not in Ck_count:
                    Ck_count[cand] = 1
                else:
                    Ck_count[cand] += 1

    num_items = float(len(dataset))
    return_list = []
    sup_rata = dict()
    for key in Ck_count:
        support  = Ck_count[key] / num_items
        if support >= min_sup:
            return_list.insert(0, key)
        sup_rata[key] = support
    return return_list, sup_rata
```

若候选项大于 2，合并时需检测子项集是否满足频繁规则：

```python
def apriori_gen(Fk, k):
    return_list = []
    len_Fk = len(Fk)

    for i in range(len_Fk):
        for j in range(i+1, len_Fk):
            F1 = list(Fk[i])[:k-2]
            F2 = list(Fk[j])[:k-2]
            F1.sort()
            F2.sort()
            if F1 == F2:
                return_list.append(Fk[i] | Fk[j])
    return return_list
```

最后按照 Apriori 算法生成频繁项集：

```python
def apriori(dataset):
    C1 = C1_generation(dataset)
    dataset = [set(data) for data in dataset]
    F1, sup_rata = Ck_low_support_filtering(dataset, C1)
    F = [F1]
    k = 2
    while len(F[k-2]) > 0:
        Ck = apriori_gen(F[k-2], k)
        Fk, support_k = Ck_low_support_filtering(dataset, Ck)
        sup_rata.update(support_k)
        F.append(Fk)
        k += 1
    return F, sup_rata
```

support_k: dict

3.关联规则挖掘

基于以上的频繁项集，实验使用 Lift 和 Jaccard 两种评价指标来对其进行评价。计算评价指标的公式如下：

$Lift(A,B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$
$Jaccard(A,B)$	$\frac{s(A \cup B)}{s(A)+s(B)-s(A \cup B)}$

计算支持度、置信度的公式如下：

$$Support(X,Y) = P(XY) = \frac{number(XY)}{num(AllSamples)}$$

$$Confidence(X \Leftarrow Y) = P(X|Y) = P(XY)/P(Y)$$

代码实现如下：

```python
def generate_rules(F, sup_rata):

    strong_rules_list = []
    for i in range(1, len(F)):
        for freq_set in F[i]:
            H1 = [frozenset([item]) for item in freq_set]
            if i > 1:
                rules_from_reasoned_item(freq_set, H1, sup_rata, strong_rules_list)
            else:
                cal_conf(freq_set, H1, sup_rata, strong_rules_list)
    return strong_rules_list

def rules_from_reasoned_item(freq_set, H, sup_rata, strong_rules_list):
    m = len(H[0])
    if len(freq_set) > (m+1):
        Hmp1 = apriori_gen(H, m+1)
        Hmp1 = cal_conf(freq_set, Hmp1, sup_rata, strong_rules_list)
        if len(Hmp1) > 1:
            rules_from_reasoned_item(freq_set, Hmp1, sup_rata, strong_rules_list)
```

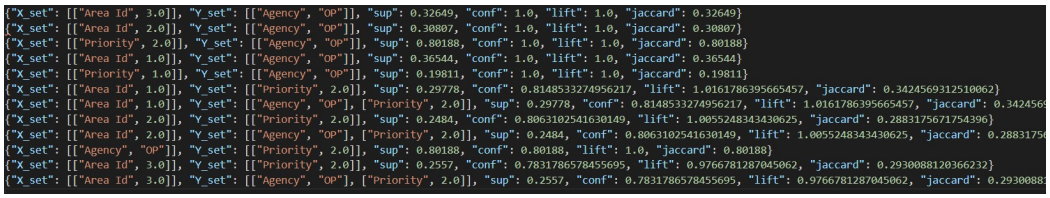
```python
def cal_conf(freq_set, H, sup_rata, strong_rules_list):
    prunedH = []
    for reasoned_item in H:
        sup = sup_rata[freq_set]
        conf = sup / sup_rata[f reasoned_item ned_item]
        lift = conf / sup_rata[reasoned_item]
        jaccard = sup / (sup_rata[freq_set - reasoned_item] + sup_rata[reasoned_item] - sup)
        if conf >= min_conf:
            strong_rules_list.append((freq_set-reasoned_item, reasoned_item, sup, conf, lift, jaccard))
            prunedH.append(reasoned_item)
    return prunedH
```

四、实验结果及分析

本次实验由于数据量过大，因此取前 10 万条数据进行实验，得到频繁项集如下：


```
{"set": [["Agency", "OP"]], "sup": 1.0}
{"set": [["Priority", 2.0]], "sup": 0.80188}
{"set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.80188}
{"set": [["Area Id", 1.0]], "sup": 0.36544}
{"set": [["Agency", "OP"], ["Area Id", 1.0]], "sup": 0.36544}
{"set": [["Area Id", 3.0]], "sup": 0.32649}
{"set": [["Area Id", 3.0], ["Agency", "OP"]], "sup": 0.32649}
{"set": [["Area Id", 2.0]], "sup": 0.30807}
{"set": [["Agency", "OP"], ["Area Id", 2.0]], "sup": 0.30807}
{"set": [["Area Id", 1.0], ["Priority", 2.0]], "sup": 0.29778}
{"set": [["Agency", "OP"], ["Area Id", 1.0], ["Priority", 2.0]], "sup": 0.29778}
{"set": [["Area Id", 3.0], ["Priority", 2.0]], "sup": 0.2557}
{"set": [["Area Id", 3.0], ["Agency", "OP"], ["Priority", 2.0]], "sup": 0.2557}
{"set": [["Area Id", 2.0], ["Priority", 2.0]], "sup": 0.2484}
{"set": [["Agency", "OP"], ["Area Id", 2.0], ["Priority", 2.0]], "sup": 0.2484}
{"set": [["Priority", 1.0]], "sup": 0.19811}
{"set": [["Agency", "OP"], ["Priority", 1.0]], "sup": 0.19811}
```


关联规则如下：



```
{"X_set": [["Area Id", 3.0]], "Y_set": [["Agency", "OP"]], "sup": 0.32649, "conf": 1.0, "lift": 1.0, "jaccard": 0.32649}
{"X_set": [["Area Id", 2.0]], "Y_set": [["Agency", "OP"]], "sup": 0.30807, "conf": 1.0, "lift": 1.0, "jaccard": 0.30807}
{"X_set": [["Priority", 2.0]], "Y_set": [["Agency", "OP"]], "sup": 0.80188, "conf": 1.0, "lift": 1.0, "jaccard": 0.80188}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Agency", "OP"]], "sup": 0.36544, "conf": 1.0, "lift": 1.0, "jaccard": 0.36544}
{"X_set": [["Priority", 1.0]], "Y_set": [["Agency", "OP"]], "sup": 0.19811, "conf": 1.0, "lift": 1.0, "jaccard": 0.19811}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Priority", 2.0]], "sup": 0.29778, "conf": 0.8148533274956217, "lift": 1.0161786395665457, "jaccard": 0.3424569312510062}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.29778, "conf": 0.8148533274956217, "lift": 1.0161786395665457, "jaccard": 0.3424569
{"X_set": [["Area Id", 2.0]], "Y_set": [["Priority", 2.0]], "sup": 0.2484, "conf": 0.8063102541630149, "lift": 1.0055248343430625, "jaccard": 0.2883175671754396}
{"X_set": [["Area Id", 2.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.2484, "conf": 0.8063102541630149, "lift": 1.0055248343430625, "jaccard": 0.2883175
{"X_set": [["Agency", "OP"]], "Y_set": [["Priority", 2.0]], "sup": 0.80188, "conf": 0.80188, "lift": 1.0, "jaccard": 0.80188}
{"X_set": [["Area Id", 3.0]], "Y_set": [["Priority", 2.0]], "sup": 0.2557, "conf": 0.7831786578455695, "lift": 0.9766781287045062, "jaccard": 0.293008812036232}
{"X_set": [["Area Id", 3.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.2557, "conf": 0.7831786578455695, "lift": 0.9766781287045062, "jaccard": 0.2930088
```


可以发现 Agency 的值均为 OP，因此不对其进行分析。观察频繁项集可以看出，Area Id 为 1.0 时支持度最高，说明在这一区域的犯罪率较高。另外 Area Id 为 2.0、3.0 时，支持度同样较高，并且 Area Id 与 Priority 的关联度也很高。

进一步观察关联规则，发现 Area Id=1.0 与 Priority=2.0 的置信度较高，并且 Area Id=2.0 或 3.0 时同样较为显著，因此可以推断出犯罪情况与所在地有

着很强的联系。

最后对频繁项集与关联规则可视化：