北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

数据挖掘

互评作业二

题　　目：频繁模式与关联规则挖掘

学　　院：计算机学院

专业名称：计算机科学与技术

学　　号：3220200998

姓　　名：余晓

任课教师：汤世平老师

一、 数据集说明

1.1 数据集介绍

在本次作业中，我们所要处理的是对 oakland-crime-statistics-2011-to-2016 进行频繁模式和关联规则挖掘。该数据集一共包含 6 个子数据集，分别为 2011-2016 的奥克兰犯罪情况。其 6 年属性值及前五列数据详细值如下：

2011：

```
属性值如下：
['Agency' 'Create Time' 'Location' 'Area Id' 'Beat' 'Priority'
 'Incident Type Id' 'Incident Type Description' 'Event Number'
 'Closed Time']
前五列数据如下：
```

	Agency	Create Time	Location	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time
0	OP	2011-01-01T00:00:00.000	ST&SAN PABLO AV	1.0	06X	1.0	PDOA	POSSIBLE DEAD PERSON	LOP110101000001	2011-01-01T00:28:17.000
1	OP	2011-01-01T00:01:11.000	ST&HANNAH ST	1.0	07X	1.0	415GS	415 GUNSHOTS	LOP110101000002	2011-01-01T01:12:56.000
2	OP	2011-01-01T00:01:25.000	ST&MARKET ST	1.0	10Y	2.0	415GS	415 GUNSHOTS	LOP110101000003	2011-01-01T00:07:20.000
3	OP	2011-01-01T00:01:35.000	PRENTISS ST	2.0	21Y	2.0	415GS	415 GUNSHOTS	LOP110101000005	2011-01-01T00:02:28.000
4	OP	2011-01-01T00:02:10.000	AV&FOOTHILL BLVD	2.0	20X	1.0	415GS	415 GUNSHOTS	LOP110101000004	2011-01-01T00:50:04.000

2012：

```
属性值如下：
['Agency' 'Create Time' 'Area Id' 'Beat' 'Priority' 'Incident Type Id'
 'Incident Type Description' 'Event Number' 'Closed Time' 'Location 1'
 'Zip Codes']
前五列数据如下：
```

	Agency	Create Time	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time	Location 1	Zip Codes
0	OP	2012-01-01T00:00:25.000	2.0	32Y	2.0	415GS	415 GUNSHOTS	LOP120101000004	2012-01-01T00:40:27.000	{'human_address': '{"address": "OLIVE ST", "ci...	NaN
1	OP	2012-01-01T00:00:27.000	2.0	30Y	2.0	415GS	415 GUNSHOTS	LOP120101000003	2012-01-01T01:34:31.000	{'human_address': '{"address": "AV&MACARTHUR B...	NaN
2	OP	2012-01-01T00:00:48.000	1.0	06X	2.0	949	SUSPICIOUS VEHICLE	LOP120101000005	2012-01-01T01:18:38.000	{'human_address': '{"address": "SYCAMORE ST", ...	NaN
3	OP	2012-01-01T00:00:58.000	2.0	35X	2.0	415GS	415 GUNSHOTS	LOP120101000008	2012-01-01T02:37:00.000	{'human_address': '{"address": "AV&MACARTHUR B...	NaN
4	OP	2012-01-01T00:01:14.000	1.0	02Y	2.0	415GS	415 GUNSHOTS	LOP120101000007	2012-01-01T02:12:39.000	{'human_address': '{"address": "ST&WOOD ST", "...	NaN

2013：

```
属性值如下：
['Agency' 'Create Time' 'Location ' 'Area Id' 'Beat' 'Priority'
 'Incident Type Id' 'Incident Type Description' 'Event Number'
 'Closed Time']
前五列数据如下：
```

	Agency	Create Time	Location	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time
0	OP	2013-01-01T00:00:00.000	D ST	2.0	33X	1.0	415GS	415 GUNSHOTS	LOP130101000002	2013-01-01T00:47:51.000
1	OP	2013-01-01T00:00:05.000	ARTHUR ST	2.0	30X	2.0	415GS	415 GUNSHOTS	LOP130101000004	2013-01-01T01:30:58.000
2	OP	2013-01-01T00:00:50.000	BRIDGE AV	2.0	23X	1.0	243E	BATTERY ON CO-HABITA	LOP130101000003	2013-01-01T05:05:50.000
3	OP	2013-01-01T00:02:16.000	AV&BROOKDALE AV	2.0	29X	2.0	415GS	415 GUNSHOTS	LOP130101000005	2013-01-01T01:37:27.000
4	OP	2013-01-01T00:02:47.000	AV&SAN LEANDRO ST	2.0	26Y	2.0	415GS	415 GUNSHOTS	LOP130101000006	2013-01-01T01:33:11.000

2014：

属性值如下：
```
['Agency' 'Create Time' 'Area Id' 'Beat' 'Priority' 'Incident Type Id'
 'Incident Type Description' 'Event Number' 'Closed Time' 'Location 1'
 'Zip Codes']
```
前五列数据如下：

	Agency	Create Time	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time	Location 1	Zip Codes
0	OP	2014-01-01T00:00:00.000	1.0	02X	2	415GS	415 GUNSHOTS	LOP140101000001	2014-01-01T03:22:08.000	{'human_address': '{"address": "LINDEN ST", "c...	NaN
1	OP	2014-01-01T00:00:00.000	2.0	26Y	2	415GS	415 GUNSHOTS	LOP140101000002	2014-01-01T02:56:31.000	{'human_address': '{"address": "AV&INTERNATION...	NaN
2	OP	2014-01-01T00:00:00.000	2.0	30Y	2	415GS	415 GUNSHOTS	LOP140101000004	2014-01-01T00:49:53.000	{'human_address': '{"address": "AV&MACARTHUR B...	NaN
3	OP	2014-01-01T00:00:00.000	2.0	30Y	2	415GS	415 GUNSHOTS	LOP140101000005	2014-01-01T02:51:11.000	{'human_address': '{"address": "MACARTHUR BLVD...	NaN
4	OP	2014-01-01T00:01:04.000	2.0	35X	2	CODE7	SUBJECT ARMED WITH W	LOP140101000010	2014-01-01T05:33:22.000	{'human_address': '{"address": "AV&DOWLING ST"...	NaN

2015：

属性值如下：
```
['Agency' 'Create Time' 'Location' 'Area Id' 'Beat' 'Priority'
 'Incident Type Id' 'Incident Type Description' 'Event Number'
 'Closed Time']
```
前五列数据如下：

	Agency	Create Time	Location	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time
0	OP	2015-01-01T00:01:59.000	S ELMHURST AV	P3	31Y	2	415	DISTURBING THE PEACE	LOP150101000003	2015-01-01T06:23:08.000
1	OP	2015-01-01T00:02:02.000	AV&D ST	P3	32X	2	415GS	415 GUNSHOTS	LOP150101000007	2015-01-01T01:44:40.000
2	OP	2015-01-01T00:02:06.000	BANCROFT AV	P3	30Y	2	933R	ALARM-RINGER	LOP150101000004	2015-01-01T02:12:39.000
3	OP	2015-01-01T00:03:16.000	MACARTHUR BLVD	P3	30Y	2	415GS	415 GUNSHOTS	LOP150101000005	2015-01-01T01:53:08.000
4	OP	2015-01-01T00:03:45.000	ST&ADELINE ST	P1	02X	2	415GS	415 GUNSHOTS	LOP150101000009	2015-01-01T00:37:09.000

2016：

属性值如下：
```
['Agency' 'Create Time' 'Location' 'Area Id' 'Beat' 'Priority'
 'Incident Type Id' 'Incident Type Description' 'Event Number'
 'Closed Time']
```
前五列数据如下：

	Agency	Create Time	Location	Area Id	Beat	Priority	Incident Type Id	Incident Type Description	Event Number	Closed Time
0	OP	2016-01-01T00:00:57.000	ST&MARKET ST	P1	05X	2.0	415GS	415 GUNSHOTS	LOP160101000003	2016-01-01T00:32:30.000
1	OP	2016-01-01T00:01:25.000	AV&HAMILTON ST	P3	26Y	2.0	415GS	415 GUNSHOTS	LOP160101000005	2016-01-01T00:48:23.000
2	OP	2016-01-01T00:01:43.000	ST&CHESTNUT ST	P1	02X	2.0	415GS	415 GUNSHOTS	LOP160101000008	2016-01-01T00:21:24.000
3	OP	2016-01-01T00:01:48.000	WALLACE ST	P2	18Y	2.0	415GS	415 GUNSHOTS	LOP160101000007	2016-01-01T01:15:03.000
4	OP	2016-01-01T00:02:05.000	90TH AV	P3	34X	2.0	415GS	415 GUNSHOTS	LOP160101000009	2016-01-01T00:54:52.000

1.2 数据集预处理

由上面六张表分析可以得知，六张表的基本属性一致，其中可以进行分析和预处理的属性如下：Agency、Location、Area id、Beat、Incident Type Id、Incident Type Description、Event Number，其中我们需要对 2012、2014 表中的 Location1 进行处理得到 Location 列名，通过观察发现，Incident Type Id 与 Incident Type Description 一一对应，所以我们只需对 Incident Type Id 进行分析即可。

```python
#获取需要进行频繁模式和关联规则挖掘的数据
data2011_new = data2011[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]
data2012_new = data2012[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]
data2013_new = data2013[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]
data2014_new = data2014[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]
data2015_new = data2015[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]
data2016_new = data2016[["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]]

#输出查看数据
print(data2011_new)
print(data2012_new)
print(data2013_new)
print(data2014_new)
print(data2015_new)
print(data2016_new)

#获取综合数据
data_all = pd.concat([data2011_new, data2012_new, data2012_new, data2014_new, data2015_new, data2016_new],
                     axis=0)
print("综合数据集有以下属性", data_all.columns)

print(data_all)
#删除空值
data_all = data_all.dropna(how='any')

print(data_all)

data_all_mining = data_all.head(50000)
print(data_all_mining)
```

对六年的数据进行合并，对于有缺失值的数据进行删除，删除前的数据为1045767 条，删除后的数据为 860353 条数，由于实验条件有限，无法大规模的数据挖掘，我们选取其前 50000 行数据进行数据挖掘。

二、 频繁模式挖掘

2.1 算法选择

本次频繁模式挖掘我们选用 Apriori 算法。Apriori 算法的步骤为首先找出所有的频繁项集，然后由频繁项集产生强关联关系（这些写规则必须满足最小支持度和置信度），其中有这样的规则：1：一个频繁项集它的子项集一定是频繁项集；2：若一个子集不是频繁项集那它的父集一定不是频繁项集。

在本实验中，我们使用 Apriori 算法构建频繁项集。其中我们设置支持度的阈值为 10%，置信度的阈值为 50%。

```python
min_sup = 0.1
min_conf = 0.5
```

2.2 代码实现

构造相应的关联规则类，算法的主体如下，生成单元数候选项集，当候选项元素大于 2 时，合并检测是否子项集满足频繁，过滤支持度低于阈值的项集。

```
min_sup = 0.1
min_conf = 0.5
Property_list = ['location', 'Area Id', 'beat', 'Priority', 'Incident Type Id', 'Event Number']
class Association_rules():
    def __init__(self):
        self.min_sup = min_sup
        self.min_conf = min_conf

    def apriori(self, dataset):                  #算法主体
        C1 = self.C1_generation(dataset)         #生成单元数候选项集
        dataset = [set(data) for data in dataset]
        F1, sup_rata = self.Ck_low_support_filtering(dataset, C1)
        F = [F1]
        k = 2
        while len(F[k-2]) > 0:
            Ck = self.apriori_gen(F[k-2], k)         #当候选项元素大于2时，合并时检测是否子项集满足频繁
            Fk, support_k = self.Ck_low_support_filtering(dataset, Ck)      #过滤支持度低于阈值的项集
            sup_rata.update(support_k)
            F.append(Fk)
            k += 1
        return F, sup_rata
```

生成单元数候选项集：

```
def C1_generation(self, dataset):        #生成单元数候选项集
    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：

```
def Ck_low_support_filtering(self, dataset, Ck):        #过滤支持度低于阈值的项集
    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()
    # 过滤非频繁项集
    for key in Ck_count:
        support  = Ck_count[key] / num_items
        if support >= self.min_sup:
            return_list.insert(0, key)
        sup_rata[key] = support
    return return_list, sup_rata
```

当候选项元素大于 2 时，合并时检测是否子项集满足频繁：

```
def apriori_gen(self, Fk, k):       #当候选项元素大于2时，合并时检测是否子项集满足频繁
    return_list = []
    len_Fk = len(Fk)

    for i in range(len_Fk):
        for j in range(i+1, len_Fk):
            # 第k-2个项相同时，将两个集合合并
            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
```

产生强关联规则算法实现，这里，我们计算关联规则以及使用的评价指标为 Lift 和 Jaccard:

其中计算的公式为：

支持度：

$$Sup(X) = \frac{count(X)}{all\_data}$$

置信度：

$$conf(X \to Y) = \frac{Sup(X \cup Y)}{Sup(X)}$$

Lift:

$$left(X \to Y) = \frac{Sup(X \cup Y)}{Sup(X) * Sup(Y)}$$

Jaccard:

$$\text{Jaccard}(X \to Y) = \frac{Sup(X \cup Y)}{Sup(X) + Sup(Y) - Sup(X \cup Y)}$$

```python
def generate_rules(self, F, sup_rata):
    """
    产生强关联规则算法实现
    基于Apriori算法，首先从一个频繁项集开始，接着创建一个规则列表，
    其中规则右部只包含一个元素，然后对这些规则进行测试。
    接下来合并所有的剩余规则列表来创建一个新的规则列表，
    其中规则右部包含两个元素。这种方法称作分级法。
    :param F: 频繁项集
    :param sup_rata: 频繁项集对应的支持度
    :return: 强关联规则列表
    """
    strong_rules_list = []
    for i in range(1, len(F)):
        for freq_set in F[i]:
            H1 = [frozenset([item]) for item in freq_set]
            # 只获取有两个或更多元素的集合
            if i > 1:
                self.rules_from_reasoned_item(freq_set, H1, sup_rata, strong_rules_list)
            else:
                self.cal_conf(freq_set, H1, sup_rata, strong_rules_list)
    return strong_rules_list
```

从推理项计算规则，计算置信度：

```python
def rules_from_reasoned_item(self, freq_set, H, sup_rata, strong_rules_list):
    """
    H->出现在规则右部的元素列表
    """
    m = len(H[0])
    if len(freq_set) > (m+1):
        Hmp1 = self.apriori_gen(H, m+1)
        Hmp1 = self.cal_conf(freq_set, Hmp1, sup_rata, strong_rules_list)
        if len(Hmp1) > 1:
            self.rules_from_reasoned_item(freq_set, Hmp1, sup_rata, strong_rules_list)

def cal_conf(self, freq_set, H, sup_rata, strong_rules_list):          #评估规则
    prunedH = []
    for reasoned_item in H:
        sup = sup_rata[freq_set]
        conf = sup / sup_rata[freq_set - reasoned_item]
        lift = conf / sup_rata[reasoned_item]
        jaccard = sup / (sup_rata[freq_set - reasoned_item] + sup_rata[reasoned_item] - sup)
        if conf >= self.min_conf:
            strong_rules_list.append((freq_set-reasoned_item, reasoned_item, sup, conf, lift, jaccard))
            prunedH.append(reasoned_item)
    return prunedH
```

至此，整个数据挖掘的流程如下：

1. 将数据转为字典存储
2. 获取频繁项集
3. 获取强关联规则列表
4. 将频繁项集输出到结果文件
5. 将关联规则输出到结果文件

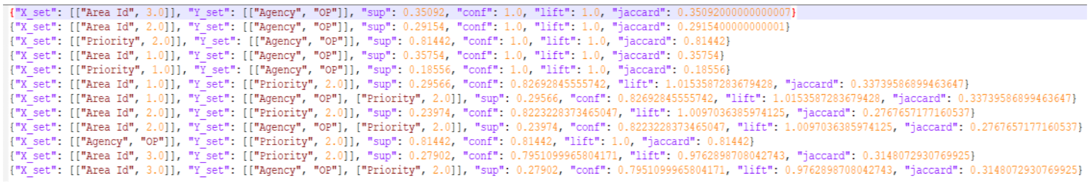
```python
def mining():
    out_path = result_path
    association = Association_rules()
    rows = data_all_mining.values.tolist()

    # 将数据转为数据字典存储
    dataset = []
    feature_names = ["Agency", "Location", "Area Id", "Beat", "Priority", "Incident Type Id", "Event Number"]
    for data_line in rows:
        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)

    # 获取频繁项集
    freq_set, sup_rata = association.apriori(dataset)
    sup_rata_out = sorted(sup_rata.items(), key=lambda d: d[1], reverse=True)
    print("sup_rata ", sup_rata)
    # 获取强关联规则列表
    strong_rules_list = association.generate_rules(freq_set, sup_rata)
    strong_rules_list = sorted(strong_rules_list, key=lambda x: x[3], reverse=True)
    print("strong_rules_list ", strong_rules_list)

    # 将频繁项集输出到结果文件
    freq_set_file = open(os.path.join(out_path, 'frequent_item.json'), 'w')
    for (key, value) in sup_rata_out:
        result_dict = {'set': None, 'sup': None}
        set_result = list(key)
        sup_result = value
        if sup_result < min_sup:
            continue
        result_dict['set'] = set_result
        result_dict['sup'] = sup_result
        json_str = json.dumps(result_dict, ensure_ascii=False)
        freq_set_file.write(json_str + '\n')
    freq_set_file.close()

    # 将关联规则输出到结果文件
    rules_file = open(os.path.join(out_path, 'related_rule.json'), 'w')
    for result in strong_rules_list:
```

三、 结果分析

我们将得到的频繁项集放到了./results/ frequent_item.json 文件中，按照支持度由大到小排列，形式如下图所示：


```
{"set": [["Agency", "OP"]], "sup": 1.0}
{"set": [["Priority", 2.0]], "sup": 0.81442}
{"set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.81442}
{"set": [["Area Id", 1.0]], "sup": 0.35754}
{"set": [["Area Id", 1.0], ["Agency", "OP"]], "sup": 0.35754}
{"set": [["Area Id", 3.0]], "sup": 0.35092}
{"set": [["Agency", "OP"], ["Area Id", 3.0]], "sup": 0.35092}
{"set": [["Area Id", 1.0], ["Priority", 2.0]], "sup": 0.29566}
{"set": [["Area Id", 1.0], ["Agency", "OP"], ["Priority", 2.0]], "sup": 0.29566}
{"set": [["Area Id", 2.0]], "sup": 0.29154}
{"set": [["Area Id", 2.0], ["Agency", "OP"]], "sup": 0.29154}
{"set": [["Area Id", 3.0], ["Priority", 2.0]], "sup": 0.27902}
{"set": [["Agency", "OP"], ["Area Id", 3.0], ["Priority", 2.0]], "sup": 0.27902}
{"set": [["Area Id", 2.0], ["Priority", 2.0]], "sup": 0.23974}
{"set": [["Area Id", 2.0], ["Agency", "OP"], ["Priority", 2.0]], "sup": 0.23974}
{"set": [["Priority", 1.0]], "sup": 0.18556}
{"set": [["Agency", "OP"], ["Priority", 1.0]], "sup": 0.18556}
```

将得到的关联规则以及评价结果放到了./results/related_rule.json 文件中，按照置信度由大到小排列，形式如下图所示：



{"X_set": [["Area Id", 3.0]], "Y_set": [["Agency", "OP"]], "sup": 0.35092, "conf": 1.0, "lift": 1.0, "jaccard": 0.35092000000000007}
{"X_set": [["Area Id", 2.0]], "Y_set": [["Agency", "OP"]], "sup": 0.29154, "conf": 1.0, "lift": 1.0, "jaccard": 0.29154000000000001}
{"X_set": [["Priority", 2.0]], "Y_set": [["Agency", "OP"]], "sup": 0.81442, "conf": 1.0, "lift": 1.0, "jaccard": 0.81442}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Agency", "OP"]], "sup": 0.35754, "conf": 1.0, "lift": 1.0, "jaccard": 0.35754}
{"X_set": [["Priority", 1.0]], "Y_set": [["Agency", "OP"]], "sup": 0.18556, "conf": 1.0, "lift": 1.0, "jaccard": 0.18556}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Priority", 2.0]], "sup": 0.29566, "conf": 0.82692845555742, "lift": 1.0153587283679428, "jaccard": 0.33739586899463647}
{"X_set": [["Area Id", 1.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.29566, "conf": 0.82692845555742, "lift": 1.0153587283679428, "jaccard": 0.33739586899463647}
{"X_set": [["Area Id", 2.0]], "Y_set": [["Priority", 2.0]], "sup": 0.23974, "conf": 0.8223228373465047, "lift": 1.0097036385974125, "jaccard": 0.2767657177160537}
{"X_set": [["Area Id", 2.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.23974, "conf": 0.8223228373465047, "lift": 1.0097036385974125, "jaccard": 0.2767657177160537}
{"X_set": [["Agency", "OP"]], "Y_set": [["Priority", 2.0]], "sup": 0.81442, "conf": 0.81442, "lift": 1.0, "jaccard": 0.81442}
{"X_set": [["Area Id", 3.0]], "Y_set": [["Priority", 2.0]], "sup": 0.27902, "conf": 0.7951099965804171, "lift": 0.9762898708042743, "jaccard": 0.3148072930769925}
{"X_set": [["Area Id", 3.0]], "Y_set": [["Agency", "OP"], ["Priority", 2.0]], "sup": 0.27902, "conf": 0.7951099965804171, "lift": 0.9762898708042743, "jaccard": 0.3148072930769925}

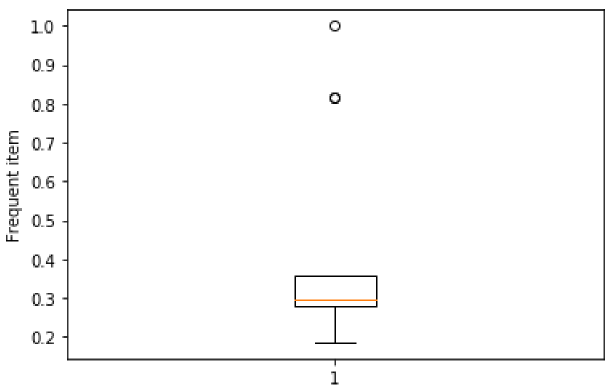
由于所有的 Agency 属性的值都是 OP，所以对其分析没有实际意义，我们跳过包含 Agency 属性的频繁项集与规则进行分析。

我们可以由频繁项集.json 得知，Area Id 为 1.0 时支持度最高，也就是说在该地区的犯罪事实出现最多。而且 Area Id 和 Priority 的关联度较高。

我们可以由规则.json 得知，["Area Id", 1.0]与["Priority", 2.0]的置信度较高，这说明犯罪的严重性与所在地有着较强联系。

四、 可视化结果

我们对频繁项集使用盒图进行可视化得如下结果：



对规则的支持度和置信度使用散点图进行可视化得如下：