pdf

2021 年 6 月 1 日

1 数据挖掘第二次作业

1.1 题目描述

1.1.1 选择 1 个数据集进行频繁模式和关联规则挖掘。

选择数据集：WineReview

1.1.2 具体要求：

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。

1.1.3 仓库地址

https://github.com/annwfsly/DataMiningHomework2

[12]:
```
import os
import json
import pandas as pd
from matplotlib import pyplot as plt

# ==========   HOMEWORK2:频繁模式及关联规则挖掘 ========
```

```
# ----- 选取的属性 -------
priorities = ["country", "designation", "points", "price", "province",␣
↪"region_1", "region_2", "variety", "winery"]
```

[13]:
```
class WineDataProc:
    def __init__(self, csv1_path, csv2_path, prior, min_s, min_c,␣
↪out_path=None):
        self.csv1_path = csv1_path
        self.csv2_path = csv2_path
        self.out_path = out_path
        self.prior = prior
        self.min_s = min_s
        self.min_c = min_c
        pass

    # ======= 数据预处理：删除缺失值 ========
    def preprocess(self):
        f130 = pd.read_csv(self.csv1_path)
        f150 = pd.read_csv(self.csv2_path)
        # ===== 缺失值剔除 =======
        f130 = f130.dropna(inplace=False)
        f150 = f150.dropna(inplace=False)
        # ===== 选取合适属性 ======
        prior_130 = f130[self.prior]
        prior_150 = f150[self.prior]

        datasets = pd.concat([prior_130, prior_150], axis=0)
        return datasets

    # ===== 数据的频繁模式及关联规则挖掘 ========
    def freq_assoc_mining(self):
        # ====== 数据格式处理 =======
        data = self.preprocess()
        data_list = data.values.tolist()
        dataset = []
        for row in data_list:
```

```
            dic = []
            for idx, elem in enumerate(row):
                dic.append((priorities[idx], elem))
            dataset.append(dic)

        # ========= 频繁项集生成 ========
        assoc = AssocRule(self.min_s, self.min_c)
        freq_set, supports = assoc.apriori(dataset)
        supports_output = sorted(supports.items(), key=lambda d: d[1],␣
↪reverse=True)
        print("supports_output ", supports_output)
        # ======== 生成关联规则 =======
        rules_list = assoc.assoc_rule_gene(freq_set, supports)
        rules_list = sorted(rules_list, key=lambda x: x[3], reverse=True)
        print("strong_rules_list ", rules_list)
        self.file_out(supports_output, rules_list)

    # ======= 将结果输出到文件 =======
    def file_out(self, supp_out, rules_list):
        # ===== 未指定输出路径，则自动定位当前代码文件夹 ======
        if self.out_path is None:
            out_path = "./"
        else:
            out_path = self.out_path
        pass
        # ====== 频繁项集写入 ========
        freq_file = open(os.path.join(out_path, 'freq_items.json'), 'w')
        for (key, value) in supp_out:
            result_dict = {'freq_set': None, 'supp': None}
            set_result = list(key)
            sup_result = value
            if sup_result < self.min_s:
                continue
            result_dict['freq_set'] = set_result
            result_dict['supp'] = sup_result
            json_str = json.dumps(result_dict, ensure_ascii=False)
```

```
            freq_file.write(json_str + '\n')
        freq_file.close()

        # ====== 关联规则写入 ========
        rules_file = open(os.path.join(out_path, 'rule_items.json'), 'w')
        for result in rules_list:
            result_dict = {'set1': None, 'set2': None, 'supp': None, 'conf':␣
↪None, 'lift': None, 'jacc': None}
            set1, set2, sup, conf, lift, jaccard = result
            result_dict['set1'] = list(set1)
            result_dict['set2'] = list(set2)
            result_dict['supp'] = sup
            result_dict['conf'] = conf
            result_dict['lift'] = lift
            result_dict['jacc'] = jaccard
            json_str = json.dumps(result_dict, ensure_ascii=False)
            rules_file.write(json_str + '\n')
        rules_file.close()
```

[14]:
```
class AssocRule:
    def __init__(self, min_s, min_c):
        self.min_s = min_s
        self.min_c = min_c
        pass

    # ======= 生成候选项集 =========
    def generate_candi(self, dataset):
        cand_1 = []
        for transac in dataset:
            for item in transac:
                if [item] not in cand_1:
                    cand_1.append([item])
        cand_1.sort()
        return [frozenset(var) for var in cand_1]

    # ========= 筛选频繁项集，计算支持度 ==========
```

```
    def freq_supports(self, d_set, cand_set):
        freq_set = set()
        item_count = {}
        supports = {}
        # ==== 计数项集元素是否出现在数据集中 ====
        for record in d_set:
            for cand in cand_set:
                if cand.issubset(record):
                    if cand not in item_count:
                        item_count[cand] = 1
                    else:
                        item_count[cand] += 1
        data_len = float(len(d_set))
        # ========== 频繁项集支持度计算 ===========
        for item in item_count:
            if (item_count[item] / data_len) >= self.min_s:
                freq_set.add(item)
                supports[item] = item_count[item] / data_len
        return freq_set, supports

    # =========== 生成新组合项集（候选项元素大于 2，合并项集） ===========
    def generate_combi(self, freq_set, k):
        new_combis = set()
        sets_len = len(freq_set)
        freq_set_list = list(freq_set)
        for i in range(sets_len):
            for j in range(i + 1, sets_len):
                l1 = list(freq_set_list[i])
                l2 = list(freq_set_list[j])
                l1.sort()
                l2.sort()
                # ========= 合并具有相同父集的项集 ========
                if l1[0:k - 2] == l2[0:k - 2]:
                    freq_item = freq_set_list[i] | freq_set_list[j]
                    new_combis.add(freq_item)
        return new_combis
```

```
# ===== aprior 算法（参考 CSDN:RinnyLu) ========
def apriori(self, dataset):
    cand = self.generate_candi(dataset)
    # ===== dataset 集合化 ======
    d_set = list(map(set, dataset))
    # ====== 生成只有一个元素的频繁项集 ========
    freq, supports = self.freq_supports(d_set, cand)
    freq_set = [freq]
    k = 2
    # ===== 循环调用 gene_comb 和 freq_sup 产生频繁项集及支持度 ======
    while len(freq_set[k - 2]) > 0:
        Ck = self.generate_combi(freq_set[k - 2], k)
        Lk, sup = self.freq_supports(d_set, Ck)
        supports.update(sup)
        freq_set.append(Lk)
        k += 1
    return freq_set, supports

# ======== 生成满足 min_support 和 min_conf 的关联规则 =========
def assoc_rule_gene(self, freq_set, supports):
    rule_list = []
    for i in range(1, len(freq_set)):
        for freq in freq_set[i]:
            # ===== H 为频繁项集中所有的元素集合 =======
            H = [frozenset([item])for item in freq]
            if i > 1:
                self.rules_conseq(freq, H, supports, rule_list)
            else:
                self.eval(freq, H, supports, rule_list)
    return rule_list

# ======= 计算置信度 conf, 提升度 lift,jaccard 系数指标 ======
def eval(self, freq_set, H, supports, rule_list):
    prunedH = []
    for conseq in H:
```

```
            sup = supports[freq_set]
            conf = sup / supports[freq_set - conseq]
            lift = conf / supports[conseq]
            jaccard = sup / (supports[freq_set - conseq] + supports[conseq] - sup)
            if conf >= self.min_c:
                print(freq_set - conseq, '-->', conseq, 'conf:', conf)
                rule_list.append((freq_set-conseq, conseq, sup, conf, lift, jaccard))
                prunedH.append(conseq)
        return prunedH

    def rules_conseq(self, freq_set, H, supports, rule_list):
        m = len(H[0])
        if len(freq_set) > (m + 1):
            Hmp1 = self.generate_combi(H, m + 1)
            Hmp1 = self.eval(freq_set, Hmp1, supports, rule_list)
            # 然后调用 calcConf() 计算 Hmp1 能够满足最小支持度来做规则后件的元素
            # 如果 Hmp1 中有值存在，则递归调用
            if len(Hmp1) > 1:
                self.rules_conseq(freq_set, Hmp1, supports, rule_list)
```

[15]:
```
# ======= 结果可视化 =======
def vision_out():
    if not os.path.isfile("./freq_items.json"):
        print("Freq File not Exist!!!")
    with open("./freq_items.json") as f1:
        freqs = [json.loads(each) for each in f1.readlines()]

    if not os.path.isfile("./rule_items.json"):
        print("Rule File not Exist!!!")
    with open("./rule_items.json") as f2:
        rules = [json.loads(each) for each in f2.readlines()]

    # ====== 频繁项支持度直方图 =======
    x_supps = [str(each["freq_set"]) for each in freqs]
```

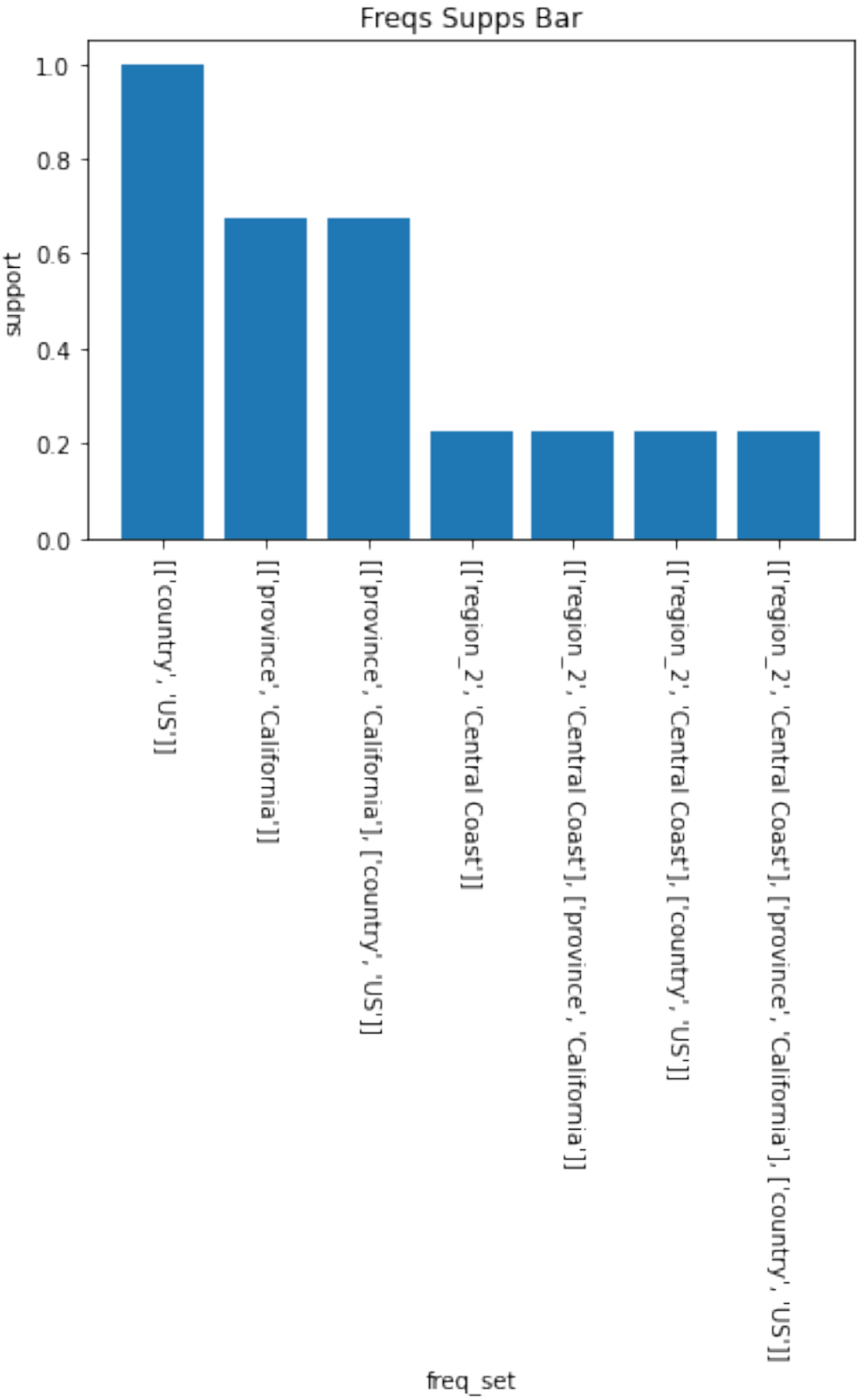
```
    y_supps = [each["supp"] for each in freqs]
    plt.bar(x_supps, y_supps)
    plt.xticks(rotation=-90)
    plt.title("Freqs Supps Bar")
    plt.xlabel("freq_set")
    plt.ylabel("support")
    plt.savefig("./freqs_supps_bar.jpg", bbox_inches='tight')
    plt.show()

    # ====== 频繁项支持度盒图 =======
    box_supps = [each["supp"] for each in freqs]
    plt.boxplot(box_supps)
    plt.title("Freqs Supps Box")
    plt.savefig("./freqs_supps_box.jpg")
    plt.show()

    # ====== 关联度 supp, conf, lift, jacc 盒图 ======
    rule_supp = [each["supp"] for each in rules]
    rule_conf = [each["conf"] for each in rules]
    rule_lift = [each["lift"] for each in rules]
    rule_jacc = [each["jacc"] for each in rules]

    plt.boxplot(rule_supp)
    plt.title("Rule Supps Box")
    plt.savefig("./rule_supps_box.jpg")
    plt.show()

    plt.boxplot(rule_conf)
    plt.title("Rule Conf Box")
    plt.savefig("./rule_conf_box.jpg")
    plt.show()

    plt.boxplot(rule_lift)
    plt.title("Rule Lift Box")
    plt.savefig("./rule_lift_box.jpg")
    plt.show()
```

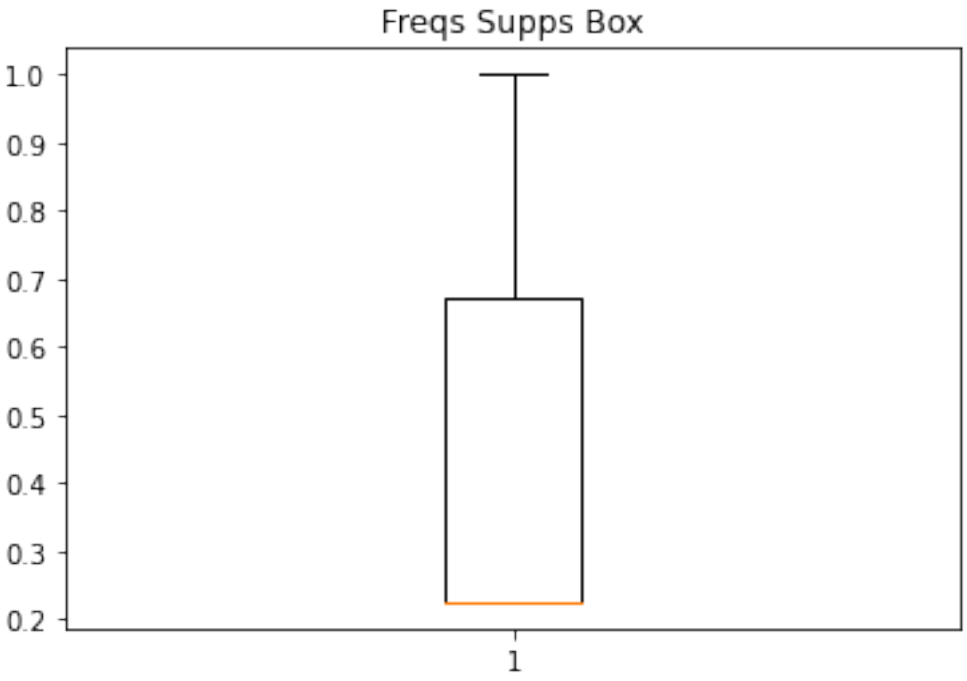
```
    plt.boxplot(rule_jacc)
    plt.title("Rule Jacc Box")
    plt.savefig("./rule_jacc_box.jpg")
    plt.show()
```

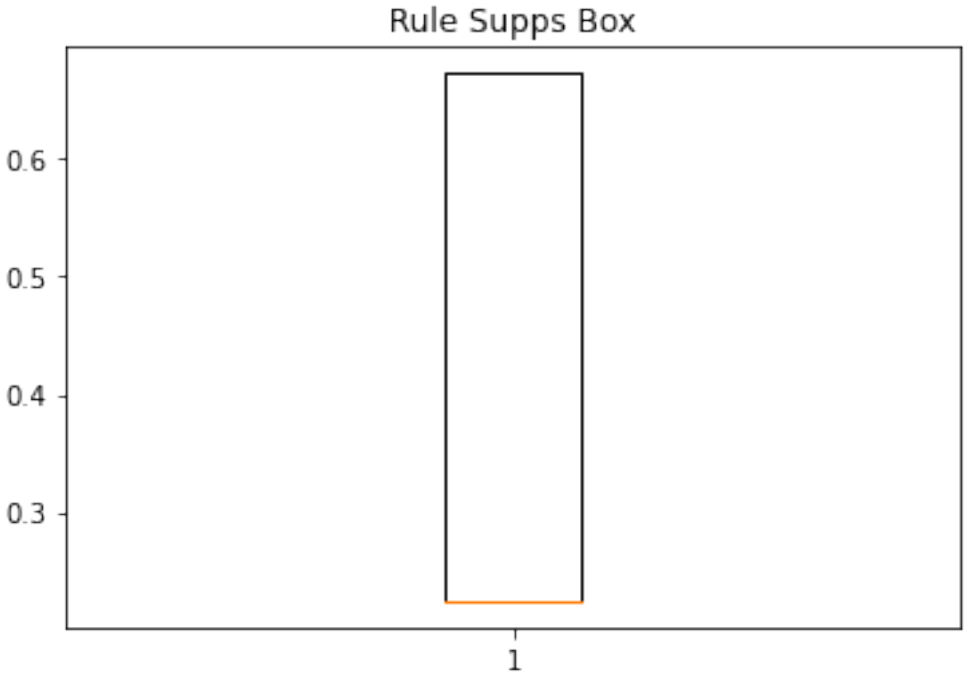
[16]:
```
if __name__ == '__main__':
    folder_path = './WineReview/'
    file_130 = os.path.join(folder_path, 'winemag-data-130k-v2.csv')
    file_150 = os.path.join(folder_path, 'winemag-data_first150k.csv')
    min_supp = 0.2
    min_conf = 0.6
    print("running.......")
    obj = WineDataProc(file_130, file_150, priorities, min_supp, min_conf)
    obj.freq_assoc_mining()
    vision_out()
```

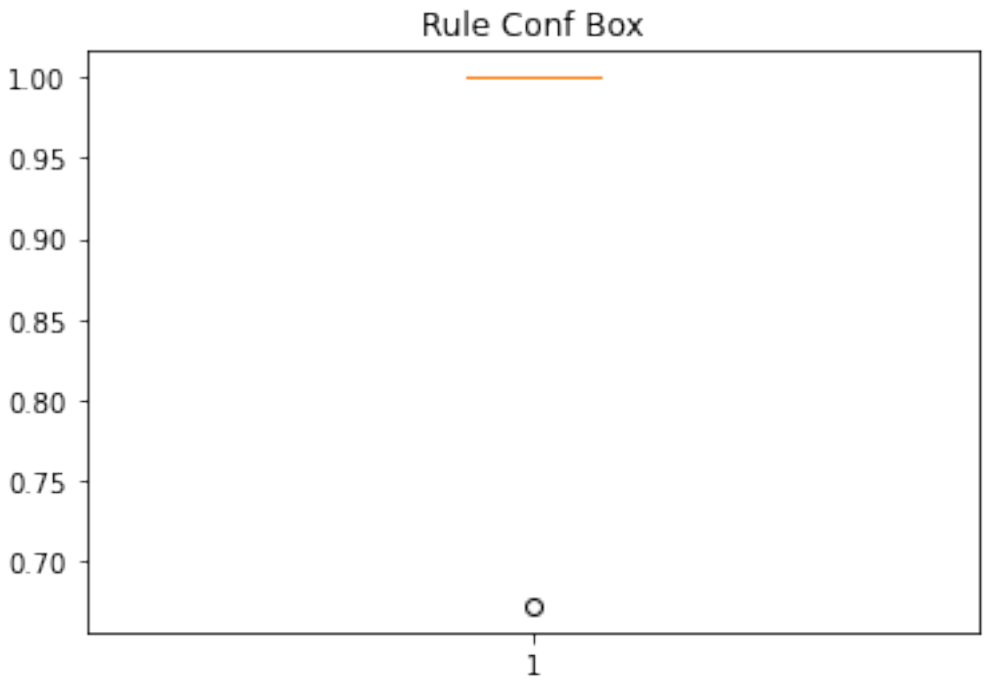
```
running…
supports_output  [(frozenset({('country', 'US')}), 1.0),
(frozenset({('province', 'California')}), 0.6726974751736223),
(frozenset({('province', 'California'), ('country', 'US')}),
0.6726974751736223), (frozenset({('region_2', 'Central Coast')}),
0.22455701953657428), (frozenset({('region_2', 'Central Coast'), ('province',
'California')}), 0.22455701953657428), (frozenset({('region_2', 'Central
Coast'), ('country', 'US')}), 0.22455701953657428), (frozenset({('region_2',
'Central Coast'), ('province', 'California'), ('country', 'US')}),
0.22455701953657428)]
frozenset({('region_2', 'Central Coast')}) --> frozenset({('province',
'California')}) conf: 1.0
frozenset({('region_2', 'Central Coast')}) --> frozenset({('country', 'US')})
conf: 1.0
frozenset({('country', 'US')}) --> frozenset({('province', 'California')}) conf:
0.6726974751736223
frozenset({('province', 'California')}) --> frozenset({('country', 'US')}) conf:
1.0
frozenset({('region_2', 'Central Coast')}) --> frozenset({('province',
'California'), ('country', 'US')}) conf: 1.0
```

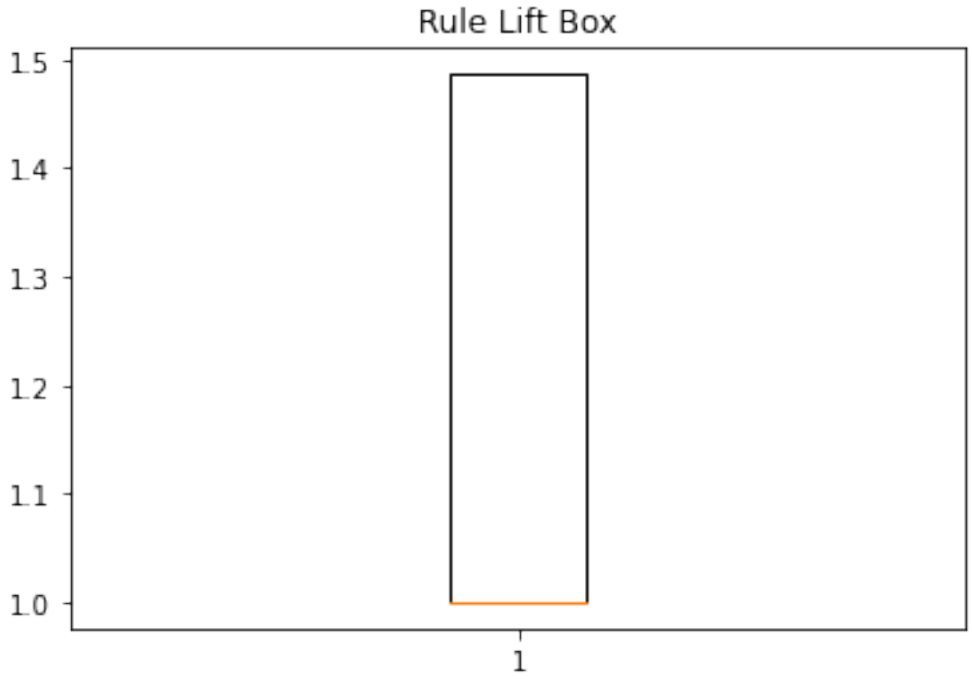
```
strong_rules_list  [(frozenset({('region_2', 'Central Coast')}),
frozenset({('province', 'California')}), 0.22455701953657428, 1.0,
1.486552331331259, 0.3338157609088936), (frozenset({('region_2', 'Central
Coast')}), frozenset({('country', 'US')}), 0.22455701953657428, 1.0, 1.0,
0.22455701953657428), (frozenset({('province', 'California')}),
frozenset({('country', 'US')}), 0.6726974751736223, 1.0, 1.0,
0.6726974751736223), (frozenset({('region_2', 'Central Coast')}),
frozenset({('province', 'California'), ('country', 'US')}), 0.22455701953657428,
1.0, 1.486552331331259, 0.3338157609088936), (frozenset({('country', 'US')}),
frozenset({('province', 'California')}), 0.6726974751736223, 0.6726974751736223,
1.0, 0.6726974751736223)]
```

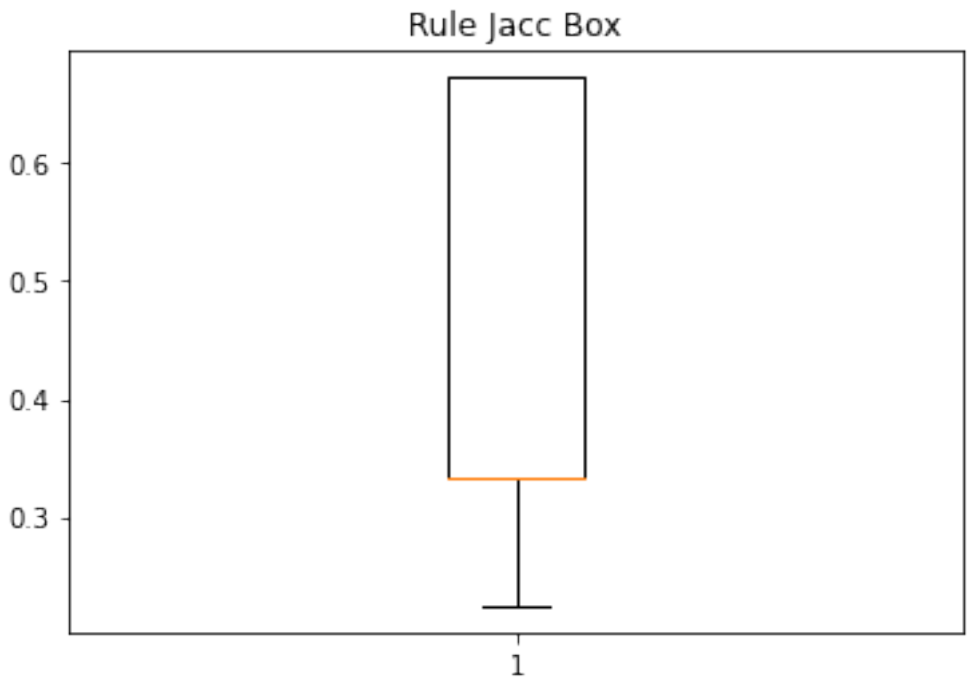

Freqs Supps Bar
support
1.0
0.8
0.6
0.4
0.2
0.0
[['country', 'US']]
[['province', 'California']]
[['province', 'California'], ['country', 'US']]
[['region_2', 'Central Coast']]
[['region_2', 'Central Coast'], ['province', 'California']]
[['region_2', 'Central Coast'], ['country', 'US']]
[['region_2', 'Central Coast'], ['province', 'California'], ['country', 'US']]
freq_set


Freqs Supps Box
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1


Rule Supps Box
0.6
0.5
0.4
0.3
1


Rule Conf Box
1.00
0.95
0.90
0.85
0.80
0.75
0.70
1


Rule Lift Box
1.5
1.4
1.3
1.2
1.1
1.0
1


Rule Jacc Box
0.6
0.5
0.4
0.3
1

[ ]: