作业2: 频繁模式与关联规则挖掘

3120215523 金玉卿

仓库地址 https://github.com/Reyna-Jin/DataMining/tree/main/assignment2 (https://github.com/Reyna-Jin/DataMining/tree/main/assignment2)

1.问题描述

选择1个数据集进行频繁模式和关联规则挖掘。

数据分析要求

- 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

数据集

wine-reviews

一共2个csv文件

- winemag-data_first150k.csv

包含10列和15万条葡萄酒评论

- winemag-data_first150k.csv

包含10列和13万行葡萄酒评论

这里我们首先分析winemag-data_first150k.csv文件的情况，数据属性如下列出：

- country 国家
- desprition 描述
- designation 葡萄酒庄
- pints 得分
- price 价格
- province 省份
- region_1 区域1
- region_2 区域2
- variety 葡萄种类
- winery 酿酒厂

2.数据处理

首先导入数据集合

In [1]:

```
import matplotlib
import numpy as np
import pandas as pd
%matplotlib inline
path_15k = "../data/wine-reviews/winemag-data_first150k.csv"
data_15k = pd.read_csv(path_15k)
```

首先需要对数据集中的不同的属性进行处理

1. 数据集中第一个属性未命名，是评论的序号，是唯一的，description属性是对于葡萄酒的自然语言描述，也是唯一值，二者在分析过程中不做考虑。

1. country、province、region_1和region_2是对葡萄酒产地的位置信息，出于分析复杂性和这四个属性的数据缺失情况考虑，这四个属性中只选择country进行挖掘。country属性中存在3个缺失值，所以需要通过属性的相关关系来填补缺失值，使用designation的属性来判断所属国家。

In [2]:

```
#根据空值的分布，定义一个从designation到country的转换字典
designation2country = {
    "Askitikos":"Greece",
    "Shah":"US",
    "Piedra Feliz":"Chile",
}
#处理country的空值
def country_nan_hander(data):
    for i in range(0,len(data)):
        tmp = data.iloc[i,1]
        if pd.isnull(tmp):
            designation = data.iloc[i,3]
            data.iloc[i,1] = designation2country[designation]
    return data
```

1. price、points是数值属性，对price进行离散化处理，此外points和price属性需要加上前缀，方便区分频繁项生成结果。

In [3]:

```
def points_discretization(value):
    return "points-"+str(int(value/5))

def price_discretization(value):
    if value < 100:
        return "price-"+str(int(value/10))
    else:
        return "price-10"
```

1. variety、winery、designation三个标称属性聚类数目过多（分别达到了632、14810、30622项），出于计算复杂度的考虑，在初步分析之后，单独选取选取variety中出现频数大于4000和winery中出现频数大于200的非空聚类进行分析。

初步分析过程中选取的属性包括designation、country、price、points，在之后的找出频繁模式调用mlxtend库来实现，因此还需要将数据处理成相应的格式。

In [4]:

```
data_15k = pd.read_csv(path_15k)

#处理country的空值
country_nan_hander(data_15k)

#过滤属性
data_15k = data_15k.drop(['Unnamed: 0','description','province','region_1','region_2','variety',
'winery','designation'],axis = 1)
```

In [5]:

```
#离散化处理
data_15k.loc[:,'points'] = data_15k['points'].map(lambda x:points_discretization(x))
data_15k.loc[:,'price'] = data_15k['price'].map(lambda x:price_discretization(x))
```

In [6]:

```
#dataframe转换为列表
def deal(data):
    return data.to_list()
data_15k_arr = data_15k.apply(deal,axis=1).tolist()
```

In [7]:

```
#TransactionEncoder转换
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
tf = te.fit_transform(data_15k_arr)
new_df = pd.DataFrame(tf,columns=te.columns_)
```

3.频繁模式

然后调用mlxtend中的apriori函数寻找频繁模式，最小支持度阈值取0.03

In [8]:

```
from mlxtend.frequent_patterns import apriori
result = apriori(new_df, min_support=0.03, use_colnames=True, max_len=4).sort_values(by='suppor
t', ascending=False)
```

In [9]:

```
print(result.shape)
result[:20]
```

(52, 2)

Out[9]:

	support	itemsets
9	0.526887	(points-17)
7	0.413423	(US)
12	0.303419	(price-1)
10	0.299669	(points-18)
14	0.212986	(price-2)
37	0.201034	(points-17, price-1)
29	0.199788	(US, points-17)
4	0.155556	(Italy)
8	0.153694	(points-16)
3	0.139787	(France)
39	0.131604	(price-2, points-17)
30	0.128748	(US, points-18)
15	0.124554	(price-3)
13	0.118121	(price-10)
32	0.106460	(US, price-2)
31	0.101617	(US, price-1)
23	0.093964	(Italy, points-17)
16	0.082840	(price-4)
35	0.079454	(price-1, points-16)
33	0.076784	(US, price-3)

4.关联规则

然后从频繁项集中导出关联规则，并计算其支持度和置信度。这里使用mlxtend包中的association_rules方法，支持度阈值为0.03，置信度阈值设为0.4，方法默认状态下会计算关联规则的计算支持度、置信度和提升度。

In [10]:

```
from mlxtend.frequent_patterns import association_rules
rules =  association_rules(result,metric ='confidence',min_threshold = 0.4)
rules = rules.drop(['leverage','conviction'],axis = 1)
print(rules.shape)
rules
```

(28， 7)

Out[10]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(price-1)	(points-17)	0.303419	0.526887	0.201034	0.662561	1.2575
1	(US)	(points-17)	0.413423	0.526887	0.199788	0.483253	0.9171
2	(price-2)	(points-17)	0.212986	0.526887	0.131604	0.617900	1.1727
3	(points-18)	(US)	0.299669	0.413423	0.128748	0.429636	1.0392
4	(price-2)	(US)	0.212986	0.413423	0.106460	0.499844	1.2090
5	(Italy)	(points-17)	0.155556	0.526887	0.093964	0.604055	1.1464
6	(points-16)	(price-1)	0.153694	0.303419	0.079454	0.516963	1.7037
7	(price-3)	(US)	0.124554	0.413423	0.076784	0.616469	1.4911
8	(points-16)	(US)	0.153694	0.413423	0.076048	0.494805	1.1968
9	(France)	(points-17)	0.139787	0.526887	0.066998	0.479287	0.9096
10	(price-3)	(points-17)	0.124554	0.526887	0.062327	0.500399	0.9497
11	(US, price-2)	(points-17)	0.106460	0.526887	0.060757	0.570700	1.0831
12	(price-2, points-17)	(US)	0.131604	0.413423	0.060757	0.461662	1.1166
13	(US, price-1)	(points-17)	0.101617	0.526887	0.058424	0.574949	1.0912
14	(price-10)	(points-18)	0.118121	0.299669	0.051898	0.439365	1.4661
15	(price-3)	(points-18)	0.124554	0.299669	0.049990	0.401351	1.3393
16	(price-10)	(France)	0.118121	0.139787	0.049930	0.422706	3.0239
17	(price-4)	(US)	0.082840	0.413423	0.049692	0.599856	1.4509
18	(price-10)	(points-17)	0.118121	0.526887	0.049102	0.415694	0.7889
19	(price-4)	(points-18)	0.082840	0.299669	0.043855	0.529393	1.7665
20	(US, price-3)	(points-17)	0.076784	0.526887	0.038064	0.495729	0.9408
21	(points-17, price-3)	(US)	0.062327	0.413423	0.038064	0.610715	1.4772
22	(US, points-16)	(price-1)	0.076048	0.303419	0.033936	0.446245	1.4707
23	(price-1, points-16)	(US)	0.079454	0.413423	0.033936	0.427118	1.0331
24	(Italy, price-1)	(points-17)	0.039422	0.526887	0.033426	0.847899	1.6092
25	(price-4)	(points-17)	0.082840	0.526887	0.033380	0.402943	0.7647
26	(Spain)	(points-17)	0.054780	0.526887	0.030504	0.556846	1.0568
27	(price-5)	(points-18)	0.049990	0.299669	0.030405	0.608217	2.0296

如下列出导出的各项关联规则：

In [11]:

```
for index, row in rules.iterrows():
    #print(row)
    t1 = tuple(row['antecedents'])
    t2 = tuple(row['consequents'])
    print("%s ⇒ %s (suupport = %f, confidence = %f )"%(t1,t2,row['support'],row['confidence'
]))
```

```
('price-1',) ⇒ ('points-17',) (suupport = 0.201034, confidence = 0.662561 )
('US',) ⇒ ('points-17',) (suupport = 0.199788, confidence = 0.483253 )
('price-2',) ⇒ ('points-17',) (suupport = 0.131604, confidence = 0.617900 )
('points-18',) ⇒ ('US',) (suupport = 0.128748, confidence = 0.429636 )
('price-2',) ⇒ ('US',) (suupport = 0.106460, confidence = 0.499844 )
('Italy',) ⇒ ('points-17',) (suupport = 0.093964, confidence = 0.604055 )
('points-16',) ⇒ ('price-1',) (suupport = 0.079454, confidence = 0.516963 )
('price-3',) ⇒ ('US',) (suupport = 0.076784, confidence = 0.616469 )
('points-16',) ⇒ ('US',) (suupport = 0.076048, confidence = 0.494805 )
('France',) ⇒ ('points-17',) (suupport = 0.066998, confidence = 0.479287 )
('price-3',) ⇒ ('points-17',) (suupport = 0.062327, confidence = 0.500399 )
('US', 'price-2') ⇒ ('points-17',) (suupport = 0.060757, confidence = 0.570700 )
('price-2', 'points-17') ⇒ ('US',) (suupport = 0.060757, confidence = 0.461662 )
('US', 'price-1') ⇒ ('points-17',) (suupport = 0.058424, confidence = 0.574949 )
('price-10',) ⇒ ('points-18',) (suupport = 0.051898, confidence = 0.439365 )
('price-3',) ⇒ ('points-18',) (suupport = 0.049990, confidence = 0.401351 )
('price-10',) ⇒ ('France',) (suupport = 0.049930, confidence = 0.422706 )
('price-4',) ⇒ ('US',) (suupport = 0.049692, confidence = 0.599856 )
('price-10',) ⇒ ('points-17',) (suupport = 0.049102, confidence = 0.415694 )
('price-4',) ⇒ ('points-18',) (suupport = 0.043855, confidence = 0.529393 )
('US', 'price-3') ⇒ ('points-17',) (suupport = 0.038064, confidence = 0.495729 )
('points-17', 'price-3') ⇒ ('US',) (suupport = 0.038064, confidence = 0.610715 )
('US', 'points-16') ⇒ ('price-1',) (suupport = 0.033936, confidence = 0.446245 )
('price-1', 'points-16') ⇒ ('US',) (suupport = 0.033936, confidence = 0.427118 )
('Italy', 'price-1') ⇒ ('points-17',) (suupport = 0.033426, confidence = 0.847899
)
('price-4',) ⇒ ('points-17',) (suupport = 0.033380, confidence = 0.402943 )
('Spain',) ⇒ ('points-17',) (suupport = 0.030504, confidence = 0.556846 )
('price-5',) ⇒ ('points-18',) (suupport = 0.030405, confidence = 0.608217 )
```

5.规则评价

然后对规则进行评价，这里使用提升度Lift和全置信度allconf。提升度Lift已经在 4.导出关联规则 的过程中被计算出来了，如下计算全置信度。

In [12]:

```
def allconf(x):
    return x.support/max(x['antecedent support'],x['consequent support'])
allconf_list = []
for index, row in rules.iterrows():
    allconf_list.append(allconf(row))
rules['allconf'] = allconf_list
rules.drop(['antecedent support','consequent support'],axis=1,inplace=False)#.sort_values(by=
['lift'], ascending=False)
```

Out[12]:

	antecedents	consequents	support	confidence	lift	allconf
0	(price-1)	(points-17)	0.201034	0.662561	1.257503	0.381550
1	(US)	(points-17)	0.199788	0.483253	0.917185	0.379186
2	(price-2)	(points-17)	0.131604	0.617900	1.172737	0.249777
3	(points-18)	(US)	0.128748	0.429636	1.039215	0.311420
4	(price-2)	(US)	0.106460	0.499844	1.209038	0.257508
5	(Italy)	(points-17)	0.093964	0.604055	1.146461	0.178338
6	(points-16)	(price-1)	0.079454	0.516963	1.703795	0.261863
7	(price-3)	(US)	0.076784	0.616469	1.491132	0.185727
8	(points-16)	(US)	0.076048	0.494805	1.196849	0.183948
9	(France)	(points-17)	0.066998	0.479287	0.909659	0.127158
10	(price-3)	(points-17)	0.062327	0.500399	0.949728	0.118293
11	(US, price-2)	(points-17)	0.060757	0.570700	1.083154	0.115313
12	(price-2, points-17)	(US)	0.060757	0.461662	1.116682	0.146960
13	(US, price-1)	(points-17)	0.058424	0.574949	1.091220	0.110886
14	(price-10)	(points-18)	0.051898	0.439365	1.466169	0.173185
15	(price-3)	(points-18)	0.049990	0.401351	1.339316	0.166818
16	(price-10)	(France)	0.049930	0.422706	3.023936	0.357190
17	(price-4)	(US)	0.049692	0.599856	1.450948	0.120196
18	(price-10)	(points-17)	0.049102	0.415694	0.788964	0.093193
19	(price-4)	(points-18)	0.043855	0.529393	1.766594	0.146344
20	(US, price-3)	(points-17)	0.038064	0.495729	0.940864	0.072243
21	(points-17, price-3)	(US)	0.038064	0.610715	1.477215	0.092070
22	(US, points-16)	(price-1)	0.033936	0.446245	1.470723	0.111846
23	(price-1, points-16)	(US)	0.033936	0.427118	1.033125	0.082086
24	(Italy, price-1)	(points-17)	0.033426	0.847899	1.609263	0.063441
25	(price-4)	(points-17)	0.033380	0.402943	0.764763	0.063353
26	(Spain)	(points-17)	0.030504	0.556846	1.056860	0.057895
27	(price-5)	(points-18)	0.030405	0.608217	2.029632	0.101461

过滤allconf小于0.1的规则，按照lift从大到小排序取前16项，得到用于分析的关联规则。

In [13]:

```
final_rules = rules.iloc[:]
from sklearn.preprocessing import LabelEncoder
for index, row in final_rules.iterrows():
    #print(row)
    if row['allconf'] < 0.1:
        final_rules.drop(index=index,inplace=True)
final_rules = final_rules.sort_values(by=['lift'], ascending=False)[:16]
final_rules
```

Out[13]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
16	(price-10)	(France)	0.118121	0.139787	0.049930	0.422706	3.0239
27	(price-5)	(points-18)	0.049990	0.299669	0.030405	0.608217	2.0296
19	(price-4)	(points-18)	0.082840	0.299669	0.043855	0.529393	1.7665
6	(points-16)	(price-1)	0.153694	0.303419	0.079454	0.516963	1.7037
7	(price-3)	(US)	0.124554	0.413423	0.076784	0.616469	1.4911
22	(US, points-16)	(price-1)	0.076048	0.303419	0.033936	0.446245	1.4707
14	(price-10)	(points-18)	0.118121	0.299669	0.051898	0.439365	1.4661
17	(price-4)	(US)	0.082840	0.413423	0.049692	0.599856	1.4509
15	(price-3)	(points-18)	0.124554	0.299669	0.049990	0.401351	1.3393
0	(price-1)	(points-17)	0.303419	0.526887	0.201034	0.662561	1.2575
4	(price-2)	(US)	0.212986	0.413423	0.106460	0.499844	1.2090
8	(points-16)	(US)	0.153694	0.413423	0.076048	0.494805	1.1968
2	(price-2)	(points-17)	0.212986	0.526887	0.131604	0.617900	1.1727
5	(Italy)	(points-17)	0.155556	0.526887	0.093964	0.604055	1.1464
12	(price-2, points-17)	(US)	0.131604	0.413423	0.060757	0.461662	1.1166
13	(US, price-1)	(points-17)	0.101617	0.526887	0.058424	0.574949	1.0912

6.结果分析/可视化展示

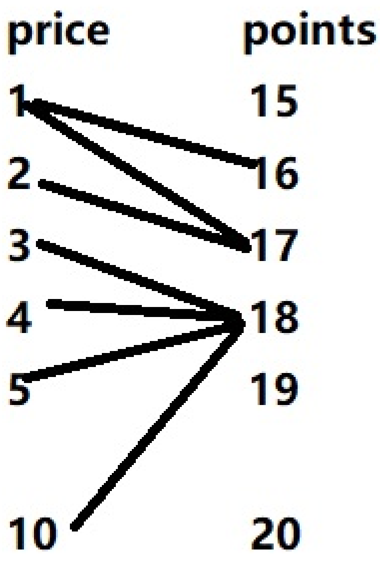
最后生成的规则如下列出:

In [14]:

```
i = 1
for index, row in final_rules.iterrows():
    t1 = tuple(row['antecedents'])
    t2 = tuple(row['consequents'])
    print("%d : %s ⇒ %s (suupport = %f, confidence = %f )"%(i,t1,t2,row['support'],row['confid
ence']))
    i = i + 1
```

```
1 : ('price-10',) ⇒ ('France',) (suupport = 0.049930, confidence = 0.422706 )
2 : ('price-5',) ⇒ ('points-18',) (suupport = 0.030405, confidence = 0.608217 )
3 : ('price-4',) ⇒ ('points-18',) (suupport = 0.043855, confidence = 0.529393 )
4 : ('points-16',) ⇒ ('price-1',) (suupport = 0.079454, confidence = 0.516963 )
5 : ('price-3',) ⇒ ('US',) (suupport = 0.076784, confidence = 0.616469 )
6 : ('US', 'points-16') ⇒ ('price-1',) (suupport = 0.033936, confidence = 0.44624
5 )
7 : ('price-10',) ⇒ ('points-18',) (suupport = 0.051898, confidence = 0.439365 )
8 : ('price-4',) ⇒ ('US',) (suupport = 0.049692, confidence = 0.599856 )
9 : ('price-3',) ⇒ ('points-18',) (suupport = 0.049990, confidence = 0.401351 )
10 : ('price-1',) ⇒ ('points-17',) (suupport = 0.201034, confidence = 0.662561 )
11 : ('price-2',) ⇒ ('US',) (suupport = 0.106460, confidence = 0.499844 )
12 : ('points-16',) ⇒ ('US',) (suupport = 0.076048, confidence = 0.494805 )
13 : ('price-2',) ⇒ ('points-17',) (suupport = 0.131604, confidence = 0.617900 )
14 : ('Italy',) ⇒ ('points-17',) (suupport = 0.093964, confidence = 0.604055 )
15 : ('price-2', 'points-17') ⇒ ('US',) (suupport = 0.060757, confidence = 0.4616
62 )
16 : ('US', 'price-1') ⇒ ('points-17',) (suupport = 0.058424, confidence = 0.5749
49 )
```

- 在price和points的数值越大代表价格越高、分数越高。根据规则2，3，4，7，9，10，13可以看出，价格对葡萄酒的评分存在一定的影响，价格比较低（price-1和price-2，对应价格区间为10-29）的葡萄酒的评分更多地集中在16和17的评分档位（对应百分制评分的80-89）。而价格相对较高的葡萄酒（price-3到price-10，价格为30以上的）评分集中在18的评分档位（对应百分制评分的90-95），而且当价格高于price-40（price>40）档位后，评分并不会升高。



- 从('price-4',) ⇒ ('US',) ('price-2',) ⇒ ('US',) ('price-16',) ⇒ ('US',) ('price-1', 'US')的规则可以看出，来自美国的葡萄酒的价格分布比较广泛。

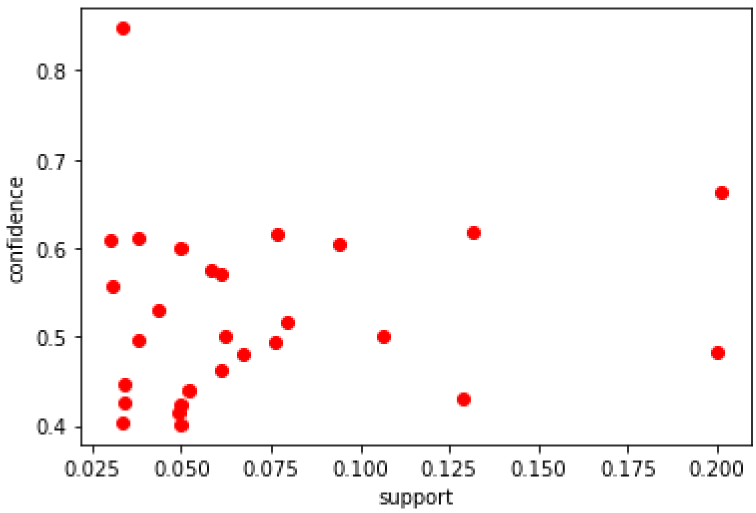
- 从('price-10',) ⇒ ('France',),('Italy',) ⇒ ('points-17',)的规则可以看出，法国的葡萄酒的价格较高（price超过100），来自意大利的葡萄酒评分居中（points位于85-90之间）。

可视化展示

使用散点图可视化生成的rules规则：

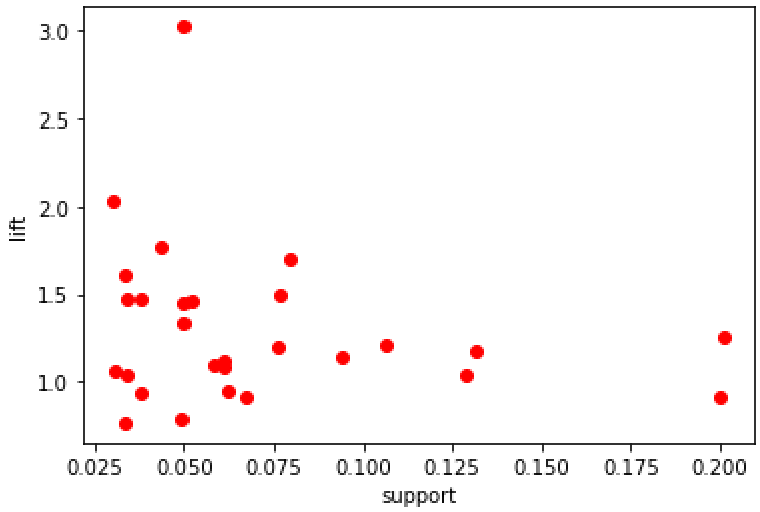
In [15]:

```
import  matplotlib.pyplot as plt
plt.xlabel('support')
plt.ylabel('confidence')
for i in range(rules.shape[0]):
    plt.scatter(rules.support[i],rules.confidence[i],c='r')
```



In [16]:

```
plt.xlabel('support')
plt.ylabel('lift')
for i in range(rules.shape[0]):
    plt.scatter(rules.support[i],rules.lift[i],c='r')
```



考虑variety和winery属性的频繁模式与关联规则挖掘

数据处理

In [17]:

```
df2 = pd.read_csv(path_15k)

#处理country的空值
country_nan_hander(df2)

#过滤属性
df2 = df2.drop(['Unnamed: 0','description','province','region_1','region_2','designation'],axis
= 1)

#离散化处理
df2.loc[:,'points'] = df2['points'].map(lambda x:points_discretization(x))
df2.loc[:,'price'] = df2['price'].map(lambda x:price_discretization(x))
```

In [32]:

```
#选取variety中出现频数大于4000的非空聚类所包括的行
variety_group = df2['variety'].value_counts()
variety_keys = []
for k in variety_group.keys():
    if variety_group[k]>4000: variety_keys.append(k)
df2_v = df2.loc[df2['variety'].isin(variety_keys)]
df2_v.drop(['winery'],axis = 1,inplace = True)
```

In [33]:

```
#选取winery中出现频数大于200的非空聚类所包括的行
winery_group = df2['winery'].value_counts()
winery_keys = []
for k in winery_group.keys():
    if winery_group[k]>200: winery_keys.append(k)
df2_w = df2.loc[df2['winery'].isin(winery_keys)]
df2_w.drop(['variety'],axis = 1,inplace = True)
```

In [19]:

```
#variety dataframe转换为列表
def deal(data):
    return data.to_list()
df2_v_arr = df2_v.apply(deal,axis=1).tolist()

#variety TransactionEncoder转换
te = TransactionEncoder()
tf = te.fit_transform(df2_v_arr)
new_df2_v = pd.DataFrame(tf,columns=te.columns_)
```

In [20]:

```
#winery dataframe转换为列表
def deal(data):
    return data.to_list()
df2_w_arr = df2_w.apply(deal,axis=1).tolist()

#winery TransactionEncoder转换
te = TransactionEncoder()
tf = te.fit_transform(df2_w_arr)
new_df2_w = pd.DataFrame(tf,columns=te.columns_)
```

频繁模式

variety和其它属性的频繁模式，最小支持度阈值取0.05

In [21]:

```
variety_result = apriori(new_df2_v, min_support=0.05, use_colnames=True, max_len=4).sort_values
(by='support', ascending=False)
```

In [22]:

```
variety_result
```

Out[22]:

	support	itemsets
12	0.550311	(US)
14	0.493068	(points-17)
15	0.329475	(points-18)
16	0.261695	(price-1)
39	0.257547	(US, points-17)
18	0.208062	(price-2)
40	0.186770	(US, points-18)
2	0.177213	(Chardonnay)
7	0.174875	(Pinot Noir)
46	0.166041	(points-17, price-1)
1	0.156630	(Cabernet Sauvignon)
13	0.152103	(points-16)
4	0.150206	(France)
19	0.143562	(price-3)
47	0.127275	(price-2, points-17)
33	0.126528	(US, Pinot Noir)
8	0.123126	(Red Blend)
42	0.123077	(US, price-2)
41	0.120654	(US, price-1)
23	0.112309	(US, Cabernet Sauvignon)
20	0.103841	(price-4)
17	0.103266	(price-10)
43	0.102825	(US, price-3)
26	0.099448	(US, Chardonnay)
38	0.091409	(US, points-16)
0	0.089903	(Bordeaux-style Red Blend)
27	0.088876	(points-17, Chardonnay)
34	0.082672	(points-17, Pinot Noir)
45	0.077557	(price-1, points-16)
10	0.077336	(Sauvignon Blanc)
44	0.076663	(US, price-4)
24	0.072588	(Cabernet Sauvignon, points-17)
48	0.071818	(points-17, price-3)

	support	itemsets
55	0.071683	(US, price-2, points-17)
11	0.071279	(Syrah)
54	0.069346	(US, points-17, price-1)
35	0.067987	(points-18, Pinot Noir)
9	0.067596	(Riesling)
30	0.066115	(France, points-17)
36	0.064622	(points-17, Red Blend)
31	0.064451	(France, points-18)
21	0.063460	(price-5)
5	0.062921	(Italy)
32	0.062261	(France, price-10)
6	0.062040	(Merlot)
22	0.060694	(France, Bordeaux-style Red Blend)
50	0.057782	(points-18, price-3)
52	0.056876	(US, points-17, Pinot Noir)
29	0.056093	(Chardonnay, price-1)
51	0.054943	(points-18, price-4)
49	0.053230	(price-10, points-18)
28	0.052887	(Chardonnay, points-18)
3	0.052851	(Chile)
25	0.052337	(Cabernet Sauvignon, points-18)
37	0.052300	(Syrah, US)
53	0.052080	(US, points-18, Pinot Noir)
56	0.050966	(US, points-17, price-3)

winery和其它属性的频繁模式，最小支持度阈值取0.05

In [23]:

```
winery_result = apriori(new_df2_w, min_support=0.05, use_colnames=True, max_len=4).sort_values(
by='support', ascending=False)
```

In [24]:

```
winery_result
```

Out[24]:

	support	itemsets
12	0.596892	(US)
15	0.462523	(points-17)
16	0.387569	(points-18)
19	0.272852	(price-1)
49	0.271024	(US, points-18)
...	...	...
26	0.051645	(Argentina, points-17)
71	0.051645	(US, Testarossa, price-5)
45	0.051645	(points-17, Trapiche)
44	0.051645	(Testarossa, price-5)
63	0.051645	(Argentina, points-17, Trapiche)

77 rows × 2 columns

导出关联规则/规则评价

然后从频繁项集中导出关联规则，并计算其支持度和置信度，支持度阈值为0.05，置信度阈值设为0.1，方法默认状态下会计算关联规则的计算支持度、置信度和提升度,此外额外计算规则的全置信度。

In [25]:

```
#variety 关联规则导出
rules_v =  association_rules(variety_result,metric ='confidence',min_threshold = 0.5)
rules_v = rules_v.drop(['leverage','conviction'],axis = 1)

allconf_list = []
for index, row in rules_v.iterrows():
    allconf_list.append(allconf(row))
rules_v['allconf'] = allconf_list

print(rules_v.shape)
rules_v[:]
```

(26, 8)

Out[25]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(points-17)	(US)	0.493068	0.550311	0.257547	0.522336	0.9491
1	(points-18)	(US)	0.329475	0.550311	0.186770	0.566871	1.0300
2	(price-1)	(points-17)	0.261695	0.493068	0.166041	0.634481	1.2868
3	(price-2)	(points-17)	0.208062	0.493068	0.127275	0.611716	1.2406
4	(Pinot Noir)	(US)	0.174875	0.550311	0.126528	0.723532	1.3147
5	(price-2)	(US)	0.208062	0.550311	0.123077	0.591543	1.0749
6	(Cabernet Sauvignon)	(US)	0.156630	0.550311	0.112309	0.717031	1.3029
7	(price-3)	(US)	0.143562	0.550311	0.102825	0.716246	1.3015
8	(Chardonnay)	(US)	0.177213	0.550311	0.099448	0.561179	1.0197
9	(points-16)	(US)	0.152103	0.550311	0.091409	0.600965	1.0920
10	(Chardonnay)	(points-17)	0.177213	0.493068	0.088876	0.501519	1.0171
11	(points-16)	(price-1)	0.152103	0.261695	0.077557	0.509895	1.9484
12	(price-4)	(US)	0.103841	0.550311	0.076663	0.738275	1.3415
13	(price-3)	(points-17)	0.143562	0.493068	0.071818	0.500256	1.0145
14	(US, price-2)	(points-17)	0.123077	0.493068	0.071683	0.582422	1.1812
15	(price-2, points-17)	(US)	0.127275	0.550311	0.071683	0.563215	1.0234
16	(US, price-1)	(points-17)	0.120654	0.493068	0.069346	0.574746	1.1656
17	(Red Blend)	(points-17)	0.123126	0.493068	0.064622	0.524846	1.0644
18	(price-10)	(France)	0.103266	0.150206	0.062261	0.602915	4.0139
19	(Bordeaux-style Red Blend)	(France)	0.089903	0.150206	0.060694	0.675105	4.4945
20	(points-17, Pinot Noir)	(US)	0.082672	0.550311	0.056876	0.687981	1.2501
21	(price-4)	(points-18)	0.103841	0.329475	0.054943	0.529107	1.6059
22	(price-10)	(points-18)	0.103266	0.329475	0.053230	0.515464	1.5645
23	(Syrah)	(US)	0.071279	0.550311	0.052300	0.733734	1.3333
24	(points-18, Pinot Noir)	(US)	0.067987	0.550311	0.052080	0.766019	1.3919
25	(points-17, price-3)	(US)	0.071818	0.550311	0.050966	0.709661	1.2895

In [26]:

```
#winery 关联规则导出
rules_w =  association_rules(winery_result,metric ='confidence',min_threshold = 0.5)
rules_w = rules_w.drop(['leverage','conviction'],axis = 1)

allconf_list = []
for index, row in rules_w.iterrows():
    allconf_list.append(allconf(row))
rules_w['allconf'] = allconf_list

print(rules_w.shape)
```

```
(88, 8)
```

结果分析/可视化

在variety和其它属性（price、points和country）导出的关联规则中，列出提升度前20条规则。

In [27]:

```
rules_v.sort_values(by='lift', ascending=False)[:20]
```

Out[27]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
19	(Bordeaux-style Red Blend)	(France)	0.089903	0.150206	0.060694	0.675105	4.4945
18	(price-10)	(France)	0.103266	0.150206	0.062261	0.602915	4.0139
11	(points-16)	(price-1)	0.152103	0.261695	0.077557	0.509895	1.9484
21	(price-4)	(points-18)	0.103841	0.329475	0.054943	0.529107	1.6059
22	(price-10)	(points-18)	0.103266	0.329475	0.053230	0.515464	1.5645
24	(points-18, Pinot Noir)	(US)	0.067987	0.550311	0.052080	0.766019	1.3919
12	(price-4)	(US)	0.103841	0.550311	0.076663	0.738275	1.3415
23	(Syrah)	(US)	0.071279	0.550311	0.052300	0.733734	1.3333
4	(Pinot Noir)	(US)	0.174875	0.550311	0.126528	0.723532	1.3147
6	(Cabernet Sauvignon)	(US)	0.156630	0.550311	0.112309	0.717031	1.3029
7	(price-3)	(US)	0.143562	0.550311	0.102825	0.716246	1.3015
25	(points-17, price-3)	(US)	0.071818	0.550311	0.050966	0.709661	1.2895
2	(price-1)	(points-17)	0.261695	0.493068	0.166041	0.634481	1.2868
20	(points-17, Pinot Noir)	(US)	0.082672	0.550311	0.056876	0.687981	1.2501
3	(price-2)	(points-17)	0.208062	0.493068	0.127275	0.611716	1.2406
14	(US, price-2)	(points-17)	0.123077	0.493068	0.071683	0.582422	1.1812
16	(US, price-1)	(points-17)	0.120654	0.493068	0.069346	0.574746	1.1656
9	(points-16)	(US)	0.152103	0.550311	0.091409	0.600965	1.0920
5	(price-2)	(US)	0.208062	0.550311	0.123077	0.591543	1.0749
17	(Red Blend)	(points-17)	0.123126	0.493068	0.064622	0.524846	1.0644

- (Bordeaux-style Red Blend)→(France) 可以看出Bordeaux-style Red Blend品种的葡萄大都种植在法国
- (Syrah)→(US)、 (Pinot Noir)→(US)、(Cabernet Sauvignon)→(US) Syrah、Pinot Noir和Cabernet Sauvignon品种的葡萄大都种植在美国

在winery和其它属性（price、points和country）导出的关联规则中，列出提升度前30条规则。

In [28]:

```
rules_w.sort_values(by='lift', ascending=False)[:12]
```

Out[28]:

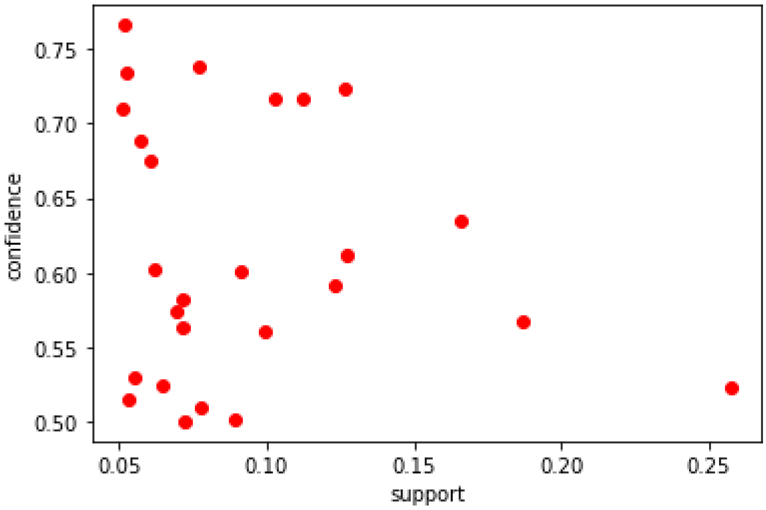
	antecedents	consequents	antecedent support	consequent support	support	confidence	
24	(Bouchard Père & Fils)	(France)	0.092779	0.092779	0.092779	1.000000	10.778
23	(France)	(Bouchard Père & Fils)	0.092779	0.092779	0.092779	1.000000	10.778
87	(Trapiche)	(Argentina, points-17)	0.093693	0.051645	0.051645	0.551220	10.673
85	(points-17, Trapiche)	(Argentina)	0.051645	0.093693	0.051645	1.000000	10.673
21	(Argentina)	(Trapiche)	0.093693	0.093693	0.093693	1.000000	10.673
83	(Argentina, points-17)	(Trapiche)	0.051645	0.093693	0.051645	1.000000	10.673
78	(Trapiche)	(Argentina, price-1)	0.093693	0.054388	0.054388	0.580488	10.673
77	(Argentina)	(price-1, Trapiche)	0.093693	0.054388	0.054388	0.580488	10.673
76	(price-1, Trapiche)	(Argentina)	0.054388	0.093693	0.054388	1.000000	10.673
22	(Trapiche)	(Argentina)	0.093693	0.093693	0.093693	1.000000	10.673
86	(Argentina)	(points-17, Trapiche)	0.093693	0.051645	0.051645	0.551220	10.673
74	(Argentina, price-1)	(Trapiche)	0.054388	0.093693	0.054388	1.000000	10.673

- (France)→(Bouchard Père & Fils)，Bouchard Père & Fils是法国比较普遍的葡萄酒庄园
- (Trapiche)→(Argentina)和(Argentina)→(Trapiche) Trapiche是阿根廷比较普遍的葡萄酒庄园
- (Trapiche)→(Argentina, price-1) Trapiche葡萄酒庄园的葡萄酒价格较为便宜(价格区间在11-19之间)

variety和其它属性关联规则可视化

In [29]:

```
import  matplotlib.pyplot as plt
plt.xlabel('support')
plt.ylabel('confidence')
for i in range(rules_v.shape[0]):
    plt.scatter(rules_v.support[i],rules_v.confidence[i],c='r')
```



winery和其它属性关联规则可视化

In [30]:

```
import  matplotlib.pyplot as plt
plt.xlabel('support')
plt.ylabel('confidence')
for i in range(rules_w.shape[0]):
    plt.scatter(rules_w.support[i],rules_w.confidence[i],c='r')
```