github仓库地址：https://github.com/106877952/DM_2

In [ ]:
```
import numpy as np
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt
from collections import defaultdict
```

In [ ]:
```
# 训练集和测试集数据路径
data_path = 'anonymous-msweb.data'
test_path = 'anonymous-msweb.test'

A_count = 0  # 记录属性数
C_count = 0  # 记录case数
V_count = 0  # 记录投票数
Attributes = {}  # 属性与编号映射字典
Cases = []  # 案例号
Votes = []  # 各案例投票
```

In [ ]:
```
# 统计data数据
data = open(data_path).readlines()
for i in range(len(data)):
    item = data[i].split(',')
    if item[0] == 'A':
        A_count += 1
        Attributes[int(item[1])] = item[3]
    if item[0] == 'C':
        C_count += 1
        Cases.append(int(item[1].split('"')[1]))
        vote = []
        while (i < len(data) - 1):
            i += 1
            if data[i][0] != 'V':
                break
            V_count += 1
            vote.append(int(data[i].split(',')[1]))
        Votes.append(vote)

# 统计test数据
data = open(test_path).readlines()
for i in range(len(data)):
    item = data[i].split(',')
    if item[0] == 'A':
        A_count += 1
        Attributes[int(item[1])] = item[3]
    if item[0] == 'C':
        C_count += 1
        Cases.append(int(item[1].split('"')[1]))
        vote = []
        while (i < len(data) - 1):
            i += 1
            if data[i][0] != 'V':
                break
            V_count += 1
            vote.append(int(data[i].split(',')[1]))
        Votes.append(vote)
```

In [ ]:
```
# 打印基本信息
print('数据集属性A数量:', A_count)
print('数据集属性C数量:', C_count)
print('数据集属性V数量:', V_count)
```

```
数据集属性A数量: 588
数据集属性C数量: 37711
数据集属性V数量: 113845
```

In [ ]:
```python
print("\n去重之后: ")
# 统计属性种数
arr_unique = set()
for attributes in Attributes: # type: ignore
    arr_unique.add(attributes)
print('所有的属性有 %d 种' % len(arr_unique))

# 统计属性种数
case_unique = set()
for case in Cases:
    case_unique.add(case)
print('所有的投票用户有 %d 个' % len(case_unique))

# 统计投票属性种数
vote_unique = set()
for Vote in Votes:
    for item in Vote:
        vote_unique.add(item)
print('所有被投票的属性有 %d 种' % len(vote_unique))
```

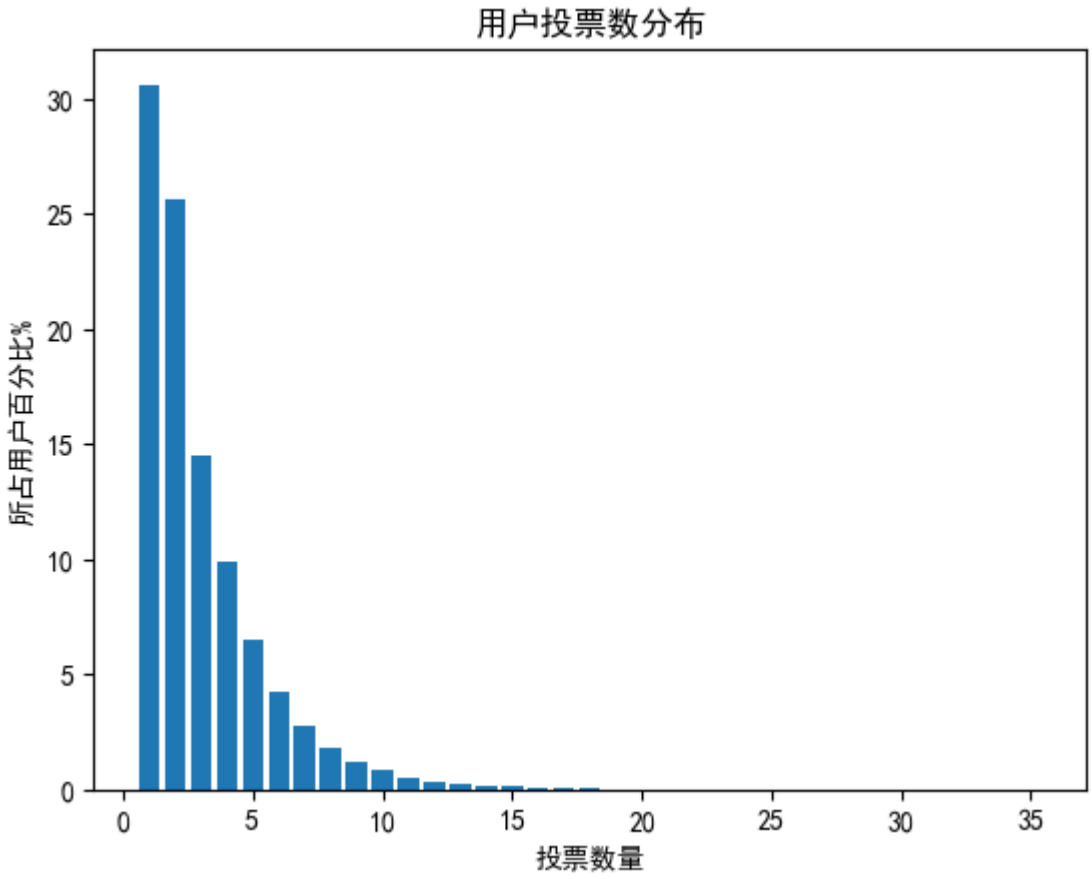
```
去重之后:
所有的属性有 294 种
所有的投票用户有 32711 个
所有被投票的属性有 286 种
```

In [ ]:
```python
# 数据可视化
votes_every_attr = defaultdict(int)  # 每个属性获得的投票数
votes_every_case = []  # 每个用户的投票数
# 遍历数据项
for item in Votes:
    votes_every_case.append(len(item))
    for attr in item:
        votes_every_attr[attr] += 1
```

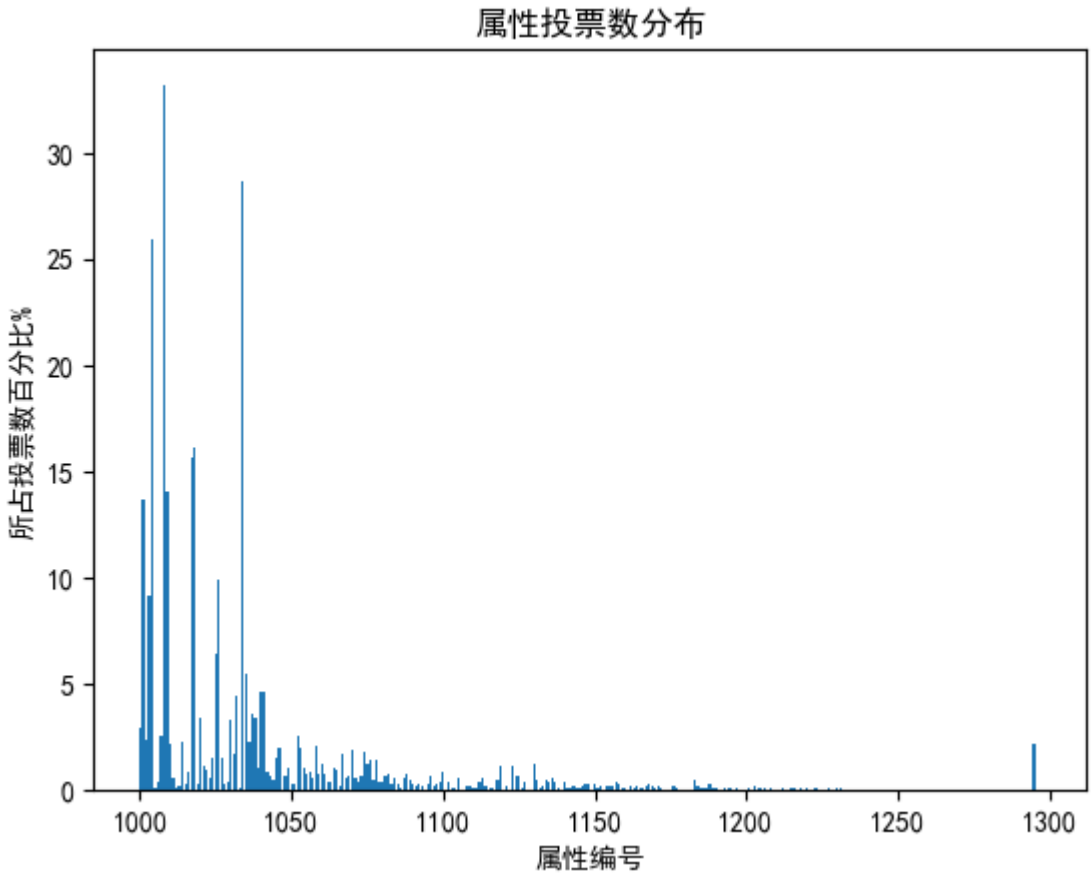
In [ ]:
```python
# 画出不同投票数的用户的分布
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']

lens, counts = np.unique(np.array(votes_every_case), return_counts=True)
plt.bar(lens, counts / len(Votes) * 100)
plt.title('用户投票数分布')
plt.xlabel('投票数量')
plt.ylabel('所占用户百分比%')
```

Out[ ]:
```
Text(0, 0.5, '所占用户百分比%')
```



In [ ]:
```
# 画出不同属性的投票数分布
plt.figure(2)
attrs = list(votes_every_attr.keys())
vote_counts = list(votes_every_attr.values())
plt.bar(attrs, np.array(vote_counts) / len(Votes) * 100)
plt.title('属性投票数分布')
plt.xlabel('属性编号')
plt.ylabel('所占投票数百分比%')
plt.show(block=True)
```



In [ ]:
```
te = TransactionEncoder()
df_tf = te.fit_transform(Votes)
df = pd.DataFrame(df_tf, columns = te.columns_) # type: ignore


# 计算频繁项集
frequent_itemsets = apriori(df, min_support=0.02, use_colnames=True)
frequent_itemsets.sort_values(by='support', ascending=False, inplace=True)
print("\n频繁项集如下：")
print(frequent_itemsets)
```

频繁项集如下：

	support	itemsets
6	0.331760	(1008)
17	0.287025	(1034)
4	0.259526	(1004)
49	0.161544	(1008, 1034)
11	0.161279	(1018)
..	...	...
69	0.021214	(1018, 1035, 1003)
26	0.020525	(1058)
67	0.020418	(1008, 1001, 1018)
25	0.020312	(1053)
78	0.020100	(1008, 1009, 1018, 1035)

[79 rows x 2 columns]

In [ ]:
```
# 计算关联规则
association_rule = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.
print("\n关联规则如下：")
print(association_rule)
```

关联规则如下：

	antecedents	consequents	antecedent support	consequent support	\
0	(1035)	(1018)	0.054281	0.161279	
1	(1037)	(1009)	0.035719	0.140410	
2	(1009, 1035)	(1018)	0.032696	0.161279	
3	(1038)	(1026)	0.034446	0.099096	
4	(1008, 1035)	(1018)	0.027127	0.161279	
5	(1001, 1035)	(1018)	0.029275	0.161279	
6	(1008, 1035)	(1009)	0.027127	0.140410	
7	(1035, 1003)	(1018)	0.024104	0.161279	
8	(1008, 1009, 1035)	(1018)	0.022354	0.161279	
9	(1008, 1018, 1035)	(1009)	0.024396	0.140410	

	support	confidence	lift	leverage	conviction	zhangs_metric
0	0.045584	0.839766	5.206905	0.036829	5.234334	0.854321
1	0.032749	0.916852	6.529824	0.027734	10.338105	0.878226
2	0.028400	0.868613	5.385773	0.023127	6.383597	0.841851
3	0.027896	0.809854	8.172436	0.024483	4.737954	0.908947
4	0.024396	0.899316	5.576142	0.020021	8.330208	0.843548
5	0.023972	0.818841	5.077162	0.019250	4.629739	0.827258
6	0.022354	0.824047	5.868864	0.018545	4.885337	0.852742
7	0.021214	0.880088	5.456922	0.017326	6.994470	0.836920
8	0.020100	0.899170	5.575236	0.016495	8.318137	0.839400
9	0.020100	0.823913	5.867910	0.016675	4.881622	0.850326

结果分析与应用: 根据以上得到的关联规则，为提升用户体验，网站应对导航结构进行如下优化： (1)在网站1035中，提供面向网站1018、1009的导航 (2)在网站1037中，提供面向网站1009的导航 (3)在网站1009中，提供面向网站1018的导航 (4)在网站1038中，提供面向网站1026的导航 (5)在网站1008中，提供面向网站1018、1009的导航 (6)在网站1001中，提供面向网站1018的导航 (7)在网站1003中，提供面向网站1018的导航 (8)在网站1018中，提供面向网站1009的导航