智能数据挖掘作业 8

19200300004 黄铭瑞

实验目的

使用 apriori 算法对数据进行关联分析。

实验原理

Apriori 算法

Apriori 算法是常用的用于挖掘出数据关联规则的算法，它用来找出数据值中频繁出现的数据集合，找出这些集合的模式有助于我们做一些决策。

关联分析

找出频繁一起出现的物品集的集合，我们称之为频繁项集，在频繁项集的基础上，使用关联规则算法找出其中物品的关联结果。

支持度

支持度就是几个关联的数据在数据集中出现的次数占总数据集的比重。或者说几个数据关联出现的概率。

$$support(A,B) = P(AB) = \frac{number(AB)}{number(Allsamples)}$$

置信度

置信度体现了一个数据出现后，另一个数据出现的概率，或者说数据的条件概率。

$$confidence(A \rightarrow B) = \frac{P(AB)}{P(A)}$$

提升度

提升度表示含有 A 的条件下，同时含有 B 的概率，与 X 总体发生的概率之比。

$$lift(A \rightarrow B) = \frac{P(AB)}{P(A)P(B)} = \frac{confidence(A \rightarrow B)}{P(B)}$$

Apriori 的两个定律

定律 1：如果某商品组合小于最小支持度，则就将它舍去，它的超集必然不是频繁项集。

定律 2：如果一个集合是频繁项集，即这个商品组合支持度大于最小支持度，则它的所有子集都是频繁项集。

算法步骤

(1) 扫描整个数据集，得到所有出现过的数据，作为候选频繁 1 项集。k=1，频繁 0 项集为空集。

(2) 挖掘频繁 k 项集

① 扫描数据计算候选频繁 k 项集的支持度。

② 去除候选频繁 k 项集中支持度低于阈值的数据集,得到频繁 k 项集。如果得到的频繁 k 项集为空，则直接返回频繁 k-1 项集的集合作为算法结果，算法结束。如果得到的频繁 k 项集只有一项，则直接返回频繁 k 项集的集合作为算法结果，算法结束。

③ 基于频繁 k 项集，连接生成候选频繁 k+1 项集。

(3) 令 k=k+1，转入步骤(2)。

实验过程

读取数据

Pd.read_csv()读取商品关联数据。

```
# 读取数据
store_data = pd.read_csv("./data/market_data.csv", header=None)
print(type(store_data))
store_data
```

类型转换

把数据转为列表

```
# 数据类型转换
dataset = []
for i in range(1,len(store_data)):
        dataset.append([str(store_data.values[i, j]) for j in range(0,20)])
print(len(dataset))
print(type(dataset))
print(dataset[:10])
```

频繁项集

from apyori import apriori，使用 apriori（）给定好最小支持度为 0.005，最小置信度为 0.2，最小提升度为 3。

```
# 根据关联规则划分频繁项集
association_rules = apriori(dataset, min_support=0.005, min_confidence=0.2, min_lift=3)
frequent_itemset = list(association_rules)
print(len(frequent_itemset))
print(frequent_itemset)
```

查看各频繁项集的支持度，置信度，提升度。

```
# 查看支持度，置信度，提升度
itemnames=[]
support=[]
confidence=[]
lift=[]
for item in frequent_itemset:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3])+"\n")

    itemnames.append(items[0]+'->'+items[1])
    support.append(item[1])
    confidence.append(item[2][0][2])
    lift.append(item[2][0][3])
```

作图

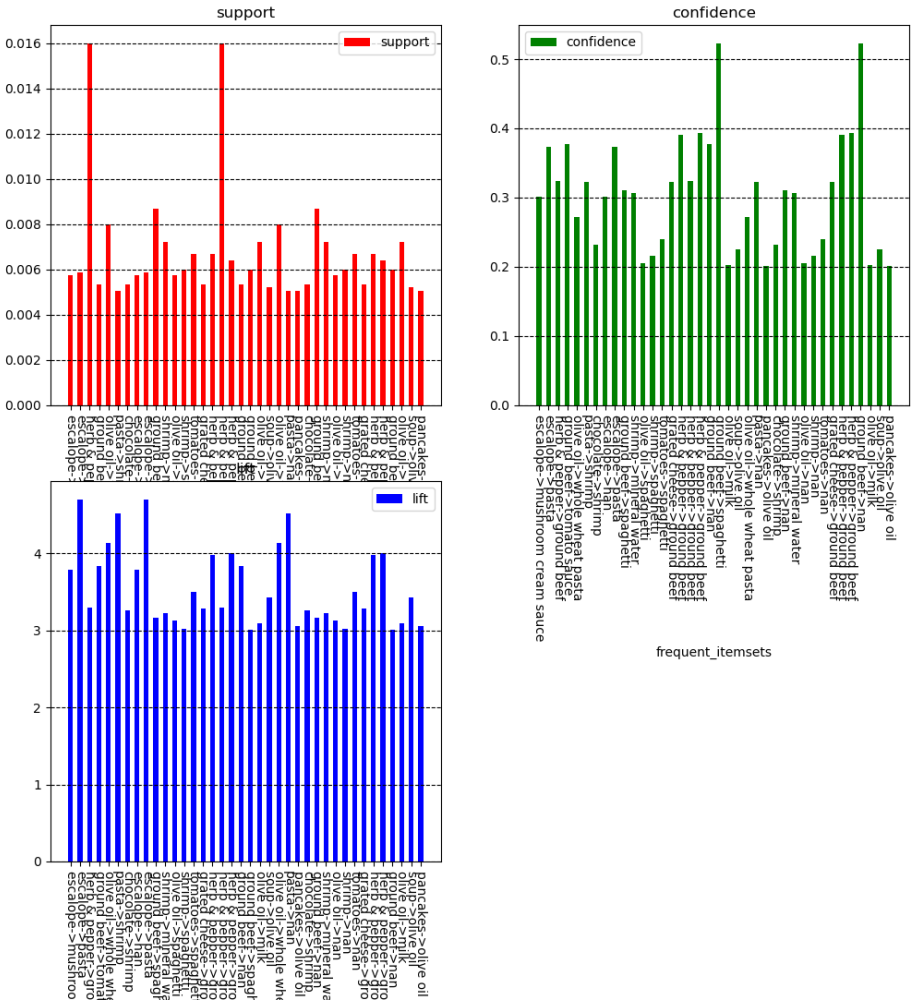
```
# 对结果作图
index1 = np.array(list(range(len(frequent_itemset))))
plt.figure(figsize=(12,12),dpi=100)

plt.subplot(2,2,1)
plt.bar(index1,support,width=0.5, color='r', label='support')
plt.xticks(index1,itemnames, rotation=-90)
plt.legend()
plt.xlabel('frequent_itemsets')
plt.title('support')
plt.grid(axis='y', ls='--',color='black')

plt.subplot(2,2,2)
plt.bar(index1,confidence,width=0.5, color='g', label='confidence')
plt.xticks(index1,itemnames, rotation=-90)
plt.legend()
plt.xlabel('frequent_itemsets')
plt.title('confidence')
plt.grid(axis='y', ls='--',color='black')

plt.subplot(2,2,3)
plt.bar(index1,lift,width=0.5, color='b', label='lift')
plt.xticks(index1,itemnames, rotation=-90)
plt.legend()
plt.xlabel('frequent_itemsets')
plt.title('lift')
plt.grid(axis='y', ls='--',color='black')
# fig = plt.figure()
# fig.tight_layout(h_pad=10.0)
plt.savefig('./pictures/result.png')
```

实验结果



附录

Apriori.html