网页浏览行为关联规则挖掘

地址：https://github.com/YuanJH728/homework2

1、数据预处理

可以看到元数据集是data文件，需要改成txt文件并导入csv文件里。其中数据集里面属性为网站，例如，A，1277,1，"NetShow for PowerPoint"，"/stream"，其中：

"A"将其标记为属性行，

"1277"是网站某个区域的属性ID号，（称为Vroot），

可以忽略"1"，

"NetShow for PowerPoint"是Vroot的标题题，

""/stream"是相对URL。

共有294个属性。

对于每个用户，都有一条案例线，后面跟着零条或多条投票线。

例如：

C、 "10164"，10164

V、 1123,1年

V、 1009，1

V、 1052，1

其中：

"C"将其标记为一个用户案例，

"10164"是用户的案例ID号，

"V"标记了这种情况下的投票线，

"1123"、"1009"、"1052"是用户访问过。

可以忽略"1"。共有32711个用户。

这里我用两个csv文件分别存储属性和用户案例命名为attribute.csv和user.csv，代码如下：

```
data = pd.read_csv("网站浏览.csv")# 数据读取，改文件保存加载后的数据
data1 = data.loc[7:300,'Column1':'Column5']#属性在7-300行
data1.to_csv("attribute.csv")
data2 = data.loc[301:131666,'Column1':'Column3']#用户案例在301-131666行
data2.to_csv("user.csv")
```

数据观察和缺失处理：

```
data_arr = pd.read_csv("attribute.csv")# 数据读取
data_user = pd.read_csv("user.csv")
print(data_arr.shape)
print(data_user.shape)
print(data1.isnull().sum())
print(data_arr.isnull().sum())
```

结果如下：

```
(294, 6)
(131365, 4)
Unnamed: 0    0
Column1       0
Column2       0
Column3       0
dtype: int64
Unnamed: 0    0
Column1       0
Column2       0
Column3       0
Column4       0
Column5       0
dtype: int64
```

可见数据没有缺失值。

```
data_arr.head(5)
```

	Unnamed: 0	Column1	Column2	Column3	Column4	Column5
0	7	A	1287	1	International AutoRoute	/autoroute
1	8	A	1288	1	library	/library
2	9	A	1289	1	Master Chef Product Information	/masterchef
3	10	A	1297	1	Central America	/centroam
4	11	A	1215	1	For Developers Only Info	/developer

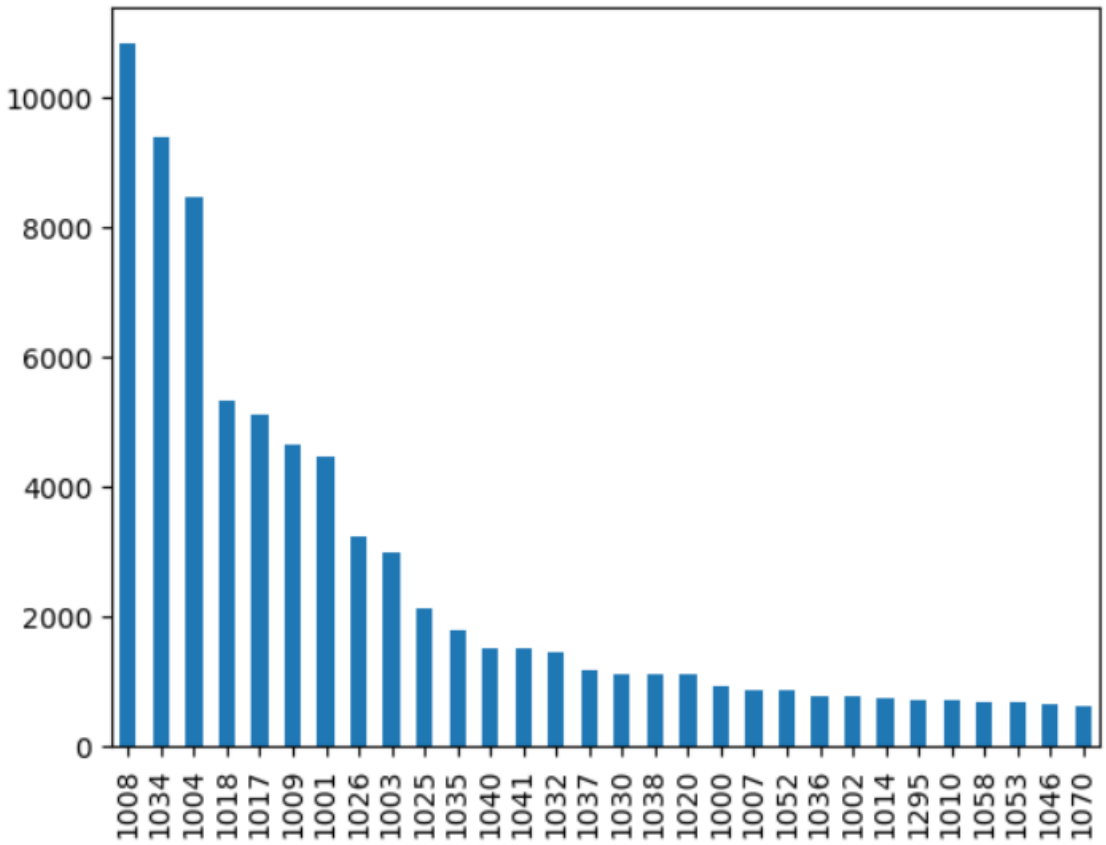
```
data_user.head(5)
```

	Unnamed: 0	Column1	Column2	Column3
0	301	C	10001	10001
1	302	V	1000	1
2	303	V	1001	1
3	304	V	1002	1
4	305	C	10002	10002

2、数据探索性分析

页面浏览数量直方图如下：

```
data_user['Column2'].value_counts().head(30).plot.bar()
```



由于人数太多无法全部显示，这里只显示访问量前30的数据，可以看出访问率第一的区域ID为1008，约有三分之一的用户都有浏览，其相对URL为"/msdownload"。下面进一步统计各区域访问率：

```
li = data_user['Column2'].value_counts()
print(sum((i >= 2000) & (i < 10000) for i in li))
print(sum((i >= 1000) & (i < 2000) for i in li))
print(sum((i >= 500) & (i < 1000) for i in li))
print(sum((i >= 200) & (i < 500) for i in li))
print(sum((i >= 100) & (i < 200) for i in li))
print(sum((i >= 50) & (i < 100) for i in li))
print(sum((i >= 10) & (i < 50) for i in li))
print(sum((i >= 5) & (i < 10) for i in li))
print(sum((i >= 2) & (i < 5) for i in li))
```

```
9
8
17
31
38
31
80
16
33
```

其次有9个区域的用户访问率大于0.06（即超过2000个用户访问过该区域），有8个区域的用户浏览率在0.03到0.06之间（即有大于1000小于2000个用户访问过该区域），有17个区域的用户浏览率在0.015到0.03之间，有31个区域的用户浏览率在0.006到0.015之间，有38个区域的用户浏览率在0.003到0.006之间，有31个区域的用户浏览率在0.0015到0.003之间，有80个区域的用户浏览率在0.0003到0.0015之间，其他访问率过小不再计算。

3、关联规则挖掘

本次挖掘采用Apriori算法，首先简单介绍Apriori算法：

关联规则：关联规则是形如 X→Y 的蕴涵表达式，其中X和Y是不相交的项集，即 X∩Y=∅。关联规则的强度可以用它的支持度（support）和置信度（confidence）来度量。

支持度：

$$Support(X,Y) = P(XY) = \frac{num(XY)}{num(allsample)}$$

置信度:

$$Confidence(X \to Y) = P(X|Y) = \frac{P(XY)}{P(X)}$$

提升度：

$$Lift(X \to Y) = \frac{Confidence(X \to Y)}{P(Y)}$$

最小支持度：最小支持度就是人为规定的阈值，表示项集在统计意义上的最低重要性。

最小置信度：最小置信度也是人为规定的阈值，表示关联规则最低可靠性。 只有支持度与置信度同时达到了最小支持度与最小置信度，此关联规则才会被称为强规则。

频繁项集：满足最小支持度的所有项集，称作频繁项集。

频繁项集性质：1、频繁项集的所有非空子集也为频繁项集；2、若A项集不是频繁项集，则其他项集或事务与A项集的并集也不是频繁项集）

根据先验原理，如果一个集合是频繁项集，则它的所有子集都是频繁项集，如果一个集合不是频繁项集，则它的所有超集都不是频繁项集。算法流程如下：

> 输入：数据集合D，支持度阈值α
>
> 输出：最大的频繁k项集
>
> 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。
>
> 2）挖掘频繁k项集
>
> a) 扫描数据计算候选频繁k项集的支持度
>
> b) 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。
>
> c) 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。

得到频繁项集以及对应支持度后就能求出关联规则的置信度和提升度，进而得出强关联规则。

本次实验根据前面用户访问率最小支持度设置为0.003，最小置信度为0.5。

4、结果评估

经过程序运行，共得到频繁 1 项数 80，频繁 2 项数206，频繁 3 项数为166，频繁 4 项数 79，频繁 5 项数为11，412条规则数，结果保存到对应txt文件中。

5、结果分析与应用

由于规则数太多无法全部使用，需要进一步过滤，这里选择置信度大于0.8且提升度大于10的规则作为强关联规则，如下所示：

```
frozenset({1008, 1060}) => frozenset({1052}) conf:  0.850 lift:  33.021
frozenset({1018, 1060}) => frozenset({1052}) conf:  0.909 lift:  35.317
frozenset({1035, 1037}) => frozenset({1009, 1018}) conf:  0.838 lift:  18.625
```

其对应相对URL为：

```
{"/msdownload", "/msword"} => {"/word"}  conf:  0.850 lift:  33.021
{"/isapi", "/msword"} => {"/word"} conf:  0.909 lift:  35.317
{"/windowssupport", "/windows95"} => {"/windows", "/isapi"} conf:  0.838 lift:
18.625
```

从上面得到浏览了"/msdownload",和"/msword"的用户再浏览"/word"的概率非常高，浏览了"/isapi",和"/msword" 的用户再浏览"/word"的可能性很高，浏览了"/windowssupport"和 "/windows95"的用户再浏览 "/windows",和"/isapi"的可能性很高，因此网站建设时可以考虑把"/msword"和"/word"放在相同页面，"/windowssupport", "/windows95","/windows", "/isapi"放在同一页面。