MO433_Tarefa_1

October 19, 2021

1 Iremos usar a lib em Python https://pypi.org/project/mlxtend/

Parte 1 - Demonstração com exemplo da aula

Parte 2 - Usando o dataset provido para a tarefa

Instalando a Python lib

[ ]:
```
!pip install mlxtend
```

[ ]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

from sklearn.preprocessing import MultiLabelBinarizer


# https://pbpython.com/market-basket-analysis.html
```

Aqui temos a lista dada em aula como exemplo para entendimento do conceito e validação do código python

[3]:
```
lista_aula = [['A', 'B', 'C'],
 ['A', 'C'],
 ['C','D'],
 ['A','B'],
 ['B','D'],
 ['D']]
```

[4]:
```
series_list = pd.Series(lista_aula)

# Aqui iremos transformar a diferentes listas em OneHotEncoding
mlb = MultiLabelBinarizer()
basket_sample = pd.DataFrame(mlb.fit_transform(series_list),
                   columns=mlb.classes_,
                   index=series_list.index)
basket_sample
```

[4]:

	A	B	C	D
0	1	1	1	0
1	1	0	1	0
2	0	0	1	1
3	1	1	0	0
4	0	1	0	1
5	0	0	0	1

Em aula o suporte para a tabela usada foi de $1/3 = 0.333$ aqui usaremos um número muito baixo para que possamos ver todos. Vemos que até o sexto elemento(index 5) contém tal suporte.

[5]:
```
frequent_itemsets = apriori(basket_sample, min_support=0.000001,␣
 ↪use_colnames=True)
frequent_itemsets
```

[5]:

	support	itemsets
0	0.500000	(A)
1	0.500000	(B)
2	0.500000	(C)
3	0.500000	(D)
4	0.333333	(A, B)
5	0.333333	(A, C)
6	0.166667	(C, B)
7	0.166667	(B, D)
8	0.166667	(C, D)
9	0.166667	(A, B, C)

[6]:
```
# Create the rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules
```

[6]:

	antecedents	consequents	antecedent support	consequent support	support	\
0	(A)	(B)	0.500000	0.500000	0.333333	
1	(B)	(A)	0.500000	0.500000	0.333333	
2	(A)	(C)	0.500000	0.500000	0.333333	
3	(C)	(A)	0.500000	0.500000	0.333333	
4	(A, B)	(C)	0.333333	0.500000	0.166667	
5	(A, C)	(B)	0.333333	0.500000	0.166667	
6	(C, B)	(A)	0.166667	0.500000	0.166667	
7	(A)	(C, B)	0.500000	0.166667	0.166667	
8	(B)	(A, C)	0.500000	0.333333	0.166667	
9	(C)	(A, B)	0.500000	0.333333	0.166667	

	confidence	lift	leverage	conviction
0	0.666667	1.333333	0.083333	1.50
1	0.666667	1.333333	0.083333	1.50

```
2    0.666667  1.333333  0.083333        1.50
3    0.666667  1.333333  0.083333        1.50
4    0.500000  1.000000  0.000000        1.00
5    0.500000  1.000000  0.000000        1.00
6    1.000000  2.000000  0.083333         inf
7    0.333333  2.000000  0.083333        1.25
8    0.333333  1.000000  0.000000        1.00
9    0.333333  1.000000  0.000000        1.00
```

1.0.1 Parte 2

Aqui iremos fazer com o dataset dado para a tarefa http://fimi.uantwerpen.be/data/retail.dat

[27]:
```python
f = open("retail.dat.txt", "r")
data = f.read()
data_lists = []
for line in data.split(' \n'):
    data_lists.append(line.split(' '))
```

[28]:
```python
series_list = pd.Series(data_lists)

# Aqui iremos transformar a diferentes listas em OneHotEncoding
mlb = MultiLabelBinarizer()
basket = pd.DataFrame(mlb.fit_transform(series_list),
                      columns=mlb.classes_,
                      index=series_list.index)
basket.head()
```

[28]:
```
   0  1  10  100  1000  10000  10001  10002  10003  …  9990  9991  9992  \
0  0  1  1    1     0      0      0      0      0  …     0     0     0
1  0  0  0    0     0      0      0      0      0  …     0     0     0
2  0  0  0    0     0      0      0      0      0  …     0     0     0
3  0  0  0    0     0      0      0      0      0  …     0     0     0
4  0  0  0    0     0      0      0      0      0  …     0     0     0

   9993  9994  9995  9996  9997  9998  9999
0     0     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0
3     0     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 16471 columns]
```

[29]:

```
# Mostra as classes em ordem alfabética, por isso vemos o 9999 por último,␣
↪mesmo tendo números maiores que ele como 15000, por exemplo.
mlb.classes_
```

[29]:
```
array(['', '0', '1', …, '9997', '9998', '9999'], dtype=object)
```

Agora iremos calcular os mais frequentues colocando suporte como 0 para trazer todos e mais a frente filtraremos.

[32]:
```
frequent_itemsets = apriori(basket, min_support=0.005, use_colnames=True)
frequent_itemsets.head(10) # 10 mais frenquentes
```

[32]:
```
    support itemsets
0  0.008076     (10)
1  0.012500   (1004)
2  0.025373    (101)
3  0.005274   (1020)
4  0.009210    (103)
5  0.006250   (1043)
6  0.005365  (10444)
7  0.007010  (10446)
8  0.007452    (105)
9  0.010004  (10515)
```

[44]:
```
print("{} itens com suporte maior que 0.005".format(len(frequent_itemsets)))
```

```
580 itens com suporte maior que 0.005
```

[33]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

[33]:
```
  antecedents consequents  antecedent support  consequent support   support  \
0        (39)        (10)            0.574788            0.008076  0.005127
1        (10)        (39)            0.008076            0.574788  0.005127
2        (48)      (1004)            0.477922            0.012500  0.006964
3      (1004)        (48)            0.012500            0.477922  0.006964
4        (39)       (101)            0.574788            0.025373  0.015880

   confidence      lift  leverage  conviction
0    0.008920  1.104463  0.000485    1.000851
1    0.634831  1.104463  0.000485    1.164428
2    0.014572  1.165816  0.000991    1.002103
3    0.557169  1.165816  0.000991    1.178956
4    0.027627  1.088816  0.001295    1.002318
```

Obtendo itens com confiança maior que 90%

[35]:
```
confidence_more_than_ninety = rules[(rules['confidence'] >= 0.9)]
confidence_more_than_ninety
```

[35]:
```
          antecedents consequents  antecedent support  consequent support  \
12              (105)        (38)            0.007452            0.176900
27              (110)        (38)            0.031691            0.176900
97            (16011)     (16010)            0.007588            0.014927
114             (170)        (38)            0.035151            0.176900
220             (286)        (38)            0.013418            0.176900
270              (36)        (38)            0.033302            0.176900
278              (37)        (38)            0.012182            0.176900
284             (371)        (38)            0.008870            0.176900
294              (55)        (38)            0.007985            0.176900
297              (56)        (38)            0.006068            0.176900
298             (790)        (38)            0.005932            0.176900
456         (105, 39)        (38)            0.005161            0.176900
463         (32, 110)        (38)            0.005093            0.176900
468         (39, 110)        (38)            0.019952            0.176900
473         (41, 110)        (38)            0.007679            0.176900
480         (48, 110)        (38)            0.015653            0.176900
556         (32, 170)        (38)            0.006125            0.176900
562         (170, 39)        (38)            0.023354            0.176900
568         (41, 170)        (38)            0.009131            0.176900
574         (170, 48)        (38)            0.017660            0.176900
718         (286, 39)        (38)            0.008507            0.176900
724         (286, 48)        (38)            0.006703            0.176900
766          (36, 32)        (38)            0.005603            0.176900
836          (36, 39)        (38)            0.023105            0.176900
842          (36, 41)        (38)            0.007940            0.176900
848          (36, 48)        (38)            0.016061            0.176900
866          (37, 39)        (38)            0.008019            0.176900
872          (37, 48)        (38)            0.006409            0.176900
878         (371, 39)        (38)            0.006034            0.176900
1056    (41, 39, 110)        (38)            0.005841            0.176900
1071    (110, 39, 48)        (38)            0.011762            0.176900
1082    (41, 170, 39)        (38)            0.007078            0.176900
1096    (170, 39, 48)        (38)            0.013679            0.176900
1110    (41, 170, 48)        (38)            0.005581            0.176900
1138    (286, 39, 48)        (38)            0.005263            0.176900
1236     (36, 41, 39)        (38)            0.006488            0.176900
1250     (36, 39, 48)        (38)            0.012658            0.176900

       support  confidence       lift  leverage  conviction
12    0.007293    0.978691   5.532466  0.005975   38.626926
27    0.030909    0.975304   5.513320  0.025302   33.329601
97    0.007384    0.973094  65.190655  0.007271   36.611884
114   0.034380    0.978057   5.528884  0.028161   37.511590
```

220	0.012658	0.943364	5.332767	0.010285	14.533250
270	0.031646	0.950272	5.371818	0.025755	16.552211
278	0.011864	0.973929	5.505548	0.009709	31.571779
284	0.008700	0.980818	5.544492	0.007131	42.910967
294	0.007452	0.933239	5.275527	0.006040	12.328993
297	0.005830	0.960748	5.431033	0.004757	20.969462
298	0.005762	0.971319	5.490794	0.004713	28.698767
456	0.005093	0.986813	5.578380	0.004180	62.418447
463	0.005025	0.986637	5.577384	0.004124	61.595346
468	0.019736	0.989198	5.591863	0.016207	76.201768
473	0.007554	0.983752	5.561074	0.006196	50.658088
480	0.015437	0.986232	5.575094	0.012668	59.783081
556	0.006034	0.985185	5.569177	0.004951	55.559277
562	0.022901	0.980573	5.543105	0.018769	42.369093
568	0.009006	0.986335	5.575679	0.007391	60.235983
574	0.017445	0.987797	5.583941	0.014321	67.450911
718	0.008257	0.970667	5.487105	0.006753	28.060241
724	0.006590	0.983080	5.557274	0.005404	48.645233
766	0.005354	0.955466	5.401174	0.004363	18.482345
836	0.022061	0.954836	5.397613	0.017974	18.224516
842	0.007611	0.958571	5.418731	0.006206	19.867941
848	0.015426	0.960452	5.429362	0.012585	20.812681
866	0.007758	0.967468	5.469024	0.006340	25.301390
872	0.006318	0.985841	5.572882	0.005184	58.131465
878	0.005966	0.988722	5.589169	0.004899	72.981568
1056	0.005796	0.992233	5.609018	0.004763	105.974176
1071	0.011694	0.994214	5.620216	0.009614	142.259185
1082	0.006976	0.985577	5.571391	0.005724	57.068294
1096	0.013532	0.989221	5.591988	0.011112	76.358390
1110	0.005490	0.983740	5.561006	0.004503	50.620674
1138	0.005195	0.987069	5.579826	0.004264	63.653097
1236	0.006272	0.966783	5.465152	0.005125	24.779654
1250	0.012250	0.967742	5.470571	0.010011	25.516112

[41]:
```
print("Temos {} itens.".format(len(confidence_more_than_ninety)))
```

```
Temos 37 itens.
```