BA Project Association Rule

April 1, 2020

[2]:
```
from itertools import combinations, groupby
from collections import Counter
from IPython.display import display
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
!pip install apyori
from apyori import apriori
```

```
Requirement already satisfied: apyori in c:\users\himat\anaconda3\lib\site-
packages (1.1.2)
```

[3]:
```
orders = pd.read_csv('order_products_original.csv')
```

[4]:
```
display(orders.head())
```

	order_id	product_id	add_to_cart_order	reordered
0	2	33120	1	1
1	2	28985	2	1
2	2	9327	3	0
3	2	45918	4	1
4	2	30035	5	0

[5]:
```
# Convert from DataFrame to a Series, with order_id as index

orders = orders.set_index('order_id')['product_id'].rename('item_id')
display(orders.head(10))
type(orders)
```

```
order_id
2    33120
2    28985
2     9327
2    45918
2    30035
```

```
2    17794
2    40141
2     1819
2    43668
3    33754
Name: item_id, dtype: int64
```

[5]:
```
pandas.core.series.Series
```

[6]:
```
# Returns frequency counts for items and item pairs
def freq(iterable):
    if type(iterable) == pd.core.series.Series:
        return iterable.value_counts().rename("freq")
    else:
        return pd.Series(Counter(iterable)).rename("freq")


# Returns number of unique orders
def order_count(order_item):
    return len(set(order_item.index))


# Returns generator that yields item pairs, one at a time
def get_item_pairs(order_item):
    order_item = order_item.reset_index().as_matrix()
    for order_id, order_object in groupby(order_item, lambda x: x[0]):
        item_list = [item[1] for item in order_object]

        for item_pair in combinations(item_list, 2):
            yield item_pair


# Returns frequency and support associated with item
def merge_item_stats(item_pairs, item_stats):
    return (item_pairs
                .merge(item_stats.rename(columns={'freq': 'freqA', 'support':␣
 ↪'supportA'}), left_on='item_A', right_index=True)
                .merge(item_stats.rename(columns={'freq': 'freqB', 'support':␣
 ↪'supportB'}), left_on='item_B', right_index=True))


# Returns name associated with item
def merge_item_name(rules, item_name):
    columns =␣
 ↪['itemA','itemB','freqAB','supportAB','freqA','supportA','freqB','supportB',
                'confidenceAtoB','confidenceBtoA','lift']
    rules = (rules
```

```
                .merge(item_name.rename(columns={'item_name': 'itemA'}),␣
↪left_on='item_A', right_on='item_id')
                .merge(item_name.rename(columns={'item_name': 'itemB'}),␣
↪left_on='item_B', right_on='item_id'))
    return rules[columns]
```

[7]:
```
def association_rules(order_item, min_support):

    print("Starting order_item: {:22d}".format(len(order_item)))


    # Calculate item frequency and support
    item_stats             = freq(order_item).to_frame("freq")
    item_stats['support']  = item_stats['freq'] / order_count(order_item) * 100


    # Filter from order_item items below min support
    qualifying_items       = item_stats[item_stats['support'] >= min_support].
↪index
    order_item             = order_item[order_item.isin(qualifying_items)]

    print("Items with support >= {}: {:15d}".format(min_support,␣
↪len(qualifying_items)))
    print("Remaining order_item: {:21d}".format(len(order_item)))


    # Filter from order_item orders with less than 2 items
    order_size             = freq(order_item.index)
    qualifying_orders      = order_size[order_size >= 2].index
    order_item             = order_item[order_item.index.isin(qualifying_orders)]

    print("Remaining orders with 2+ items: {:11d}".
↪format(len(qualifying_orders)))
    print("Remaining order_item: {:21d}".format(len(order_item)))


    # Recalculate item frequency and support
    item_stats             = freq(order_item).to_frame("freq")
    item_stats['support']  = item_stats['freq'] / order_count(order_item) * 100


    # Get item pairs generator
    item_pair_gen          = get_item_pairs(order_item)


    # Calculate item pair frequency and support
    item_pairs              = freq(item_pair_gen).to_frame("freqAB")
```

```
    item_pairs['supportAB'] = item_pairs['freqAB'] / len(qualifying_orders) * 100

    print("Item pairs: {:31d}".format(len(item_pairs)))


    # Filter from item_pairs those below min support
    item_pairs              = item_pairs[item_pairs['supportAB'] >= min_support]

    print("Item pairs with support >= {}: {:10d}\n".format(min_support,␣
↪len(item_pairs)))


    # Create table of association rules and compute relevant metrics
    item_pairs = item_pairs.reset_index().rename(columns={'level_0': 'item_A',␣
↪'level_1': 'item_B'})
    item_pairs = merge_item_stats(item_pairs, item_stats)

    item_pairs['confidenceAtoB'] = item_pairs['supportAB'] /␣
↪item_pairs['supportA']
    item_pairs['confidenceBtoA'] = item_pairs['supportAB'] /␣
↪item_pairs['supportB']
    item_pairs['lift']           = item_pairs['supportAB'] /␣
↪(item_pairs['supportA'] * item_pairs['supportB'])


    # Return association rules sorted by lift in descending order
    return item_pairs.sort_values('lift', ascending=False)
```

[8]:
```
%%time
rules = association_rules(orders, 0.02)
```

```
Starting order_item:                32434489
Items with support >= 0.02:            6985
Remaining order_item:               28050418
Remaining orders with 2+ items:      2975061
Remaining order_item:               27848926

C:\Users\himat\Anaconda3\lib\site-packages\ipykernel_launcher.py:16:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  app.launch_new_instance()

Item pairs:                         20384055
Item pairs with support >= 0.02:       18153

Wall time: 7min 11s
```

[10]:
```
item_name   = pd.read_csv('products.csv')
item_name   = item_name.rename(columns={'product_id':'item_id', 'product_name':
 ↪'item_name'})
rules_final = merge_item_name(rules, item_name).sort_values('lift',␣
 ↪ascending=False)
rules_final
```

[10]:

	itemA
0	Oh My Yog! Pacific Coast Strawberry Trilayer Y...
2	Unsweetened Blackberry Water
3	Organic Fiber & Protein Pear Blueberry & Spina...
4	Oh My Yog! Organic Wild Quebec Blueberry Cream...
1	Oh My Yog! Pacific Coast Strawberry Trilayer Y...
5	Unsweetened Watermelon Water
7	Organic Greek Whole Milk Blended Strawberry Yo...
8	Organic Greek Whole Milk Blended Strawberry Yo...
9	Chocolate Sea Salt
11	Coconut Chia Bar
15	Greek Whole Milk Blended Blueberry Yogurt
10	Chocolate Sea Salt
17	Almond Milk Strawberry Yogurt
18	Almond Milk Peach Yogurt
35	Stage 1 Apples Sweet Potatoes Pumpkin & Bluebe...
6	Unsweetened Watermelon Water
12	Chocolate Peanut Butter
37	Stage 1 Apples Sweet Potatoes Pumpkin & Bluebe...
21	Almond Milk Strawberry Yogurt
38	Organic Pears, Peas and Broccoli Puree Stage 1
36	Organic Pears, Peas and Broccoli Puree Stage 1
53	Non-Fat Greek Yogurt With Strawberries on the ...
45	Chocolate Peanut Butter
22	Almond Milk Blueberry Yogurt
55	Organic Lemon Lowfat Yogurt
13	Coconut Chia Bar
30	Almond Milk Peach Yogurt
56	Yotoddler Organic Pear Spinach Mango Yogurt
31	Almond Milk Blueberry Yogurt
1017	YoKids Squeeze! Organic Strawberry Flavor Yogurt
...	...
741	Organic Broccoli Crowns
1202	Cucumber Kirby
610	Sparkling Mineral Water
811	Organic Unsalted Butter
595	Sinfully Sweet Campari Tomatoes
1170	Boneless Skinless Chicken Breasts
597	Sustainably Soft Bath Tissue
1215	Granny Smith Apples

714	Organic Lemon
826	Organic Green Seedless Grapes
1250	Green Beans
14600	Organic Strawberries
1188	Yellow Onions
624	Blackberries
1116	2% Reduced Fat Milk
14604	Organic Baby Spinach
667	Organic Carrot Bunch
1199	Russet Potato
1190	Broccoli Crown
731	Organic Navel Orange
14684	Bag of Organic Bananas
1144	Bunched Cilantro
1203	Roma Tomato
2773	Raspberries
590	Hass Avocados
1181	Red Onion
7291	Banana
146	Strawberries
3600	Organic Strawberries
1195	Banana

	itemB	freqAB	supportAB	\
0	Oh My Yog! Organic Wild Quebec Blueberry Cream...	860	0.028907	
2	Raspberry Essence Water	660	0.022184	
3	Fiber & Protein Organic Pears, Raspberries, Bu...	606	0.020369	
4	Oh My Yog! Pacific Coast Strawberry Trilayer Y...	642	0.021579	
1	Oh My Yog! Madagascar Vanilla Trilayer Yogyurt	640	0.021512	
5	Unsweet Peach Water	599	0.020134	
7	Organic Blended Raspberry Whole Milk Greek Yogurt	780	0.026218	
8	Greek Whole Milk Blended Blueberry Yogurt	769	0.025848	
9	Coconut Chocolate Bar	859	0.028873	
11	Acai Berry Chia Bar	870	0.029243	
15	Organic Greek Whole Milk Blended Strawberry Yo...	689	0.023159	
10	Peanut Butter Egg White Protein Bar	631	0.021210	
17	Almond Milk Blueberry Yogurt	1640	0.055125	
18	Almond Milk Blueberry Yogurt	1289	0.043327	
35	Organic Apples, Carrots and Parsnips Puree	813	0.027327	
6	Unsweetened Blackberry Water	710	0.023865	
12	Acai Berry Chia Bar	729	0.024504	
37	Organic 4 Months Butternut Squash Carrots Appl...	930	0.031260	
21	Almond Milk Peach Yogurt	1376	0.046251	
38	Organic 4 Months Butternut Squash Carrots Appl...	812	0.027294	
36	Organic Apples, Carrots and Parsnips Puree	628	0.021109	
53	Non-Fat Blueberry on the Bottom Greek Yogurt	670	0.022521	
45	Coconut Chia Bar	970	0.032604	

22	Almond Milk Peach Yogurt	971	0.032638
55	Lowfat Key Lime Yogurt	657	0.022084
13	Chocolate Peanut Butter	900	0.030251
30	Almond Milk Strawberry Yogurt	1101	0.037008
56	Organic Whole Milk Strawberry Beet Berry Yogur...	1533	0.051528
31	Almond Milk Strawberry Yogurt	1062	0.035697
1017	YoKids Squeeze Organic Blueberry Blue Yogurt	926	0.031125
...	...	...	...
741	Banana	743	0.024974
1202	Bag of Organic Bananas	3103	0.104300
610	Banana	682	0.022924
811	Banana	851	0.028604
595	Banana	617	0.020739
1170	Bag of Organic Bananas	1573	0.052873
597	Banana	710	0.023865
1215	Bag of Organic Bananas	1121	0.037680
714	Banana	3408	0.114552
826	Banana	838	0.028167
1250	Bag of Organic Bananas	893	0.030016
14600	Whole Milk	756	0.025411
1188	Bag of Organic Bananas	2182	0.073343
624	Banana	842	0.028302
1116	Bag of Organic Bananas	1087	0.036537
14604	Whole Milk	663	0.022285
667	Banana	1318	0.044302
1199	Bag of Organic Bananas	840	0.028235
1190	Bag of Organic Bananas	1197	0.040234
731	Banana	1477	0.049646
14684	2% Reduced Fat Milk	1013	0.034050
1144	Bag of Organic Bananas	1197	0.040234
1203	Bag of Organic Bananas	1033	0.034722
2773	Organic Hass Avocado	810	0.027226
590	Banana	1482	0.049814
1181	Bag of Organic Bananas	1008	0.033882
7291	Soda	864	0.029041
146	Organic Strawberries	706	0.023731
3600	Strawberries	640	0.021512
1195	Bag of Organic Bananas	654	0.021983

	freqA	supportA	freqB	supportB	confidenceAtoB	confidenceBtoA	\
0	2856	0.095998	2271	0.076335	0.301120	0.378688	
2	3108	0.104468	2025	0.068066	0.212355	0.325926	
3	2782	0.093511	2167	0.072839	0.217829	0.279649	
4	2271	0.076335	2856	0.095998	0.282695	0.224790	
1	2856	0.095998	2567	0.086284	0.224090	0.249318	
5	4107	0.138048	1777	0.059730	0.145849	0.337085	
7	4198	0.141106	2598	0.087326	0.185803	0.300231	

8	4198	0.141106	2647	0.088973	0.183182	0.290518
9	4326	0.145409	3041	0.102216	0.198567	0.282473
11	4646	0.156165	2879	0.096771	0.187258	0.302188
15	2647	0.088973	4198	0.141106	0.260295	0.164126
10	4326	0.145409	2373	0.079763	0.145862	0.265908
17	5712	0.191996	4714	0.158451	0.287115	0.347900
18	4703	0.158081	4714	0.158451	0.274080	0.273441
35	5577	0.187458	2568	0.086318	0.145777	0.316589
6	4107	0.138048	3108	0.104468	0.172876	0.228443
12	4649	0.156266	2879	0.096771	0.156808	0.253213
37	5577	0.187458	3234	0.108704	0.166756	0.287570
21	5712	0.191996	4703	0.158081	0.240896	0.292579
38	5028	0.169005	3234	0.108704	0.161496	0.251082
36	5028	0.169005	2568	0.086318	0.124901	0.244548
53	4634	0.155762	3015	0.101342	0.144584	0.222222
45	4649	0.156266	4646	0.156165	0.208647	0.208782
22	4714	0.158451	4703	0.158081	0.205982	0.206464
55	3954	0.132905	3912	0.131493	0.166161	0.167945
13	4646	0.156165	4649	0.156266	0.193715	0.193590
30	4703	0.158081	5712	0.191996	0.234106	0.192752
56	6120	0.205710	6271	0.210786	0.250490	0.244459
31	4714	0.158451	5712	0.191996	0.225286	0.185924
1017	6799	0.228533	3468	0.116569	0.136196	0.267013
...	...	...	...	...	...	...
741	18295	0.614945	469627	15.785458	0.040612	0.001582
1202	97039	3.261748	376021	12.639102	0.031977	0.008252
610	17119	0.575417	469627	15.785458	0.039839	0.001452
811	21378	0.718574	469627	15.785458	0.039807	0.001812
595	15551	0.522712	469627	15.785458	0.039676	0.001314
1170	49914	1.677747	376021	12.639102	0.031514	0.004183
597	18136	0.609601	469627	15.785458	0.039149	0.001512
1215	35895	1.206530	376021	12.639102	0.031230	0.002981
714	87441	2.939133	469627	15.785458	0.038975	0.007257
826	21641	0.727414	469627	15.785458	0.038723	0.001784
1250	28997	0.974669	376021	12.639102	0.030796	0.002375
14600	263285	8.849735	35072	1.178867	0.002871	0.021556
1188	73026	2.454605	376021	12.639102	0.029880	0.005803
624	22576	0.758842	469627	15.785458	0.037296	0.001793
1116	36639	1.231538	376021	12.639102	0.029668	0.002891
14604	240508	8.084137	35072	1.178867	0.002757	0.018904
667	35725	1.200816	469627	15.785458	0.036893	0.002806
1199	29196	0.981358	376021	12.639102	0.028771	0.002234
1190	41959	1.410358	376021	12.639102	0.028528	0.003183
731	42633	1.433013	469627	15.785458	0.034645	0.003145
14684	376021	12.639102	36639	1.231538	0.002694	0.027648
1144	45411	1.526389	376021	12.639102	0.026359	0.003183
1203	40053	1.346292	376021	12.639102	0.025791	0.002747

2773	56774	1.908331	212674	7.148559	0.014267	0.003809
590	49154	1.652201	469627	15.785458	0.030150	0.003156
1181	42881	1.441349	376021	12.639102	0.023507	0.002681
7291	469627	15.785458	32674	1.098263	0.001840	0.026443
146	141660	4.761583	263285	8.849735	0.004984	0.002682
3600	263285	8.849735	141660	4.761583	0.002431	0.004518
1195	469627	15.785458	376021	12.639102	0.001393	0.001739

	lift
0	3.944745
2	3.119850
3	2.990560
4	2.944798
1	2.597119
5	2.441803
7	2.127693
8	2.058855
9	1.942612
11	1.935059
15	1.844670
10	1.828694
17	1.812016
18	1.729754
35	1.688849
6	1.654811
12	1.620400
37	1.534045
21	1.523881
38	1.485650
36	1.446989
53	1.426682
45	1.336069
22	1.303018
55	1.263647
13	1.239652
30	1.219327
56	1.188365
31	1.173391
1017	1.168376
...	...
741	0.002573
1202	0.002530
610	0.002524
811	0.002522
595	0.002513
1170	0.002493
597	0.002480

```
1215   0.002471
714    0.002469
826    0.002453
1250   0.002437
14600  0.002436
1188   0.002364
624    0.002363
1116   0.002347
14604  0.002338
667    0.002337
1199   0.002276
1190   0.002257
731    0.002195
14684  0.002188
1144   0.002086
1203   0.002041
2773   0.001996
590    0.001910
1181   0.001860
7291   0.001675
146    0.000563
3600   0.000511
1195   0.000110

[18153 rows x 11 columns]
```

[2]:
```
type(rules_final)
rules_final.to_csv("ResultBA1.csv", index= False)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-2-6bb8d59a913c> in <module>
----> 1 type(rules_final)
      2 rules_final.to_csv("ResultBA1.csv", index= False)


NameError: name 'rules_final' is not defined
```

[ ]: