Market Basket Analysis

February 17, 2022

1 Research Question

We want to use Market Basket Analysis (MBA) to find connections in prescription drugs. The goal is to find patterns in patients with mutliple prescriptions. That is, if a person has a prescription for X will they likey have a prescription for Y as well?

2 Market Basket Justification

MBA analyzes data by finding pairs of items and counting how frequent the pairing occurs. It's underlying assumption is that joint occurence of two or more items in the most "baskets" imply that these products are complements in purchase. Or in our case, prescription. An example of a paring from our dataset is:

$\{paroxetine\}$ -> $\{allopurinol\}$

In this example allopurinol is the **consequent** to the **antecedent** of paroxetine. That is, being prescribed allopurinol could be described as the "result" of being prescribed paroxetine. Our expected outcome is to find the most common of these pairings.

3 Data Preparation and Analysis

We will begin by importing and inspecting our dataset

[ ]:
```
import pandas as pd

df = pd.read_csv('data/medical_market_basket.csv')

df.head()
```

[ ]:
```
      Presc01            Presc02                      Presc03       Presc04  \
0         NaN                NaN                          NaN           NaN
1  amlodipine  albuterol aerosol                  allopurinol  pantoprazole
2         NaN                NaN                          NaN           NaN
3  citalopram            benicar  amphetamine salt combo xr           NaN
4         NaN                NaN                          NaN           NaN

     Presc05     Presc06     Presc07      Presc08     Presc09      Presc10  \
0        NaN         NaN         NaN          NaN         NaN          NaN
```

```
1  lorazepam  omeprazole  mometasone  fluconozole  gabapentin  pravastatin
2        NaN         NaN         NaN          NaN         NaN          NaN
3        NaN         NaN         NaN          NaN         NaN          NaN
4        NaN         NaN         NaN          NaN         NaN          NaN

  Presc11   Presc12                 Presc13           Presc14  Presc15  \
0     NaN       NaN                     NaN               NaN      NaN
1  cialis  losartan  metoprolol succinate XL  sulfamethoxazole  abilify
2     NaN       NaN                     NaN               NaN      NaN
3     NaN       NaN                     NaN               NaN      NaN
4     NaN       NaN                     NaN               NaN      NaN

          Presc16        Presc17       Presc18       Presc19    Presc20
0             NaN            NaN           NaN           NaN        NaN
1  spironolactone  albuterol HFA  levofloxacin  promethazine  glipizide
2             NaN            NaN           NaN           NaN        NaN
3             NaN            NaN           NaN           NaN        NaN
4             NaN            NaN           NaN           NaN        NaN
```

We're given a dataset where a row represents a customer's perscriptions. We also see that this particular dataset is full of null values. We iterate the dataframe and extract the non-null perscription values.

[ ]:
```python
prescriptions = []
for i, row in df.iterrows():
    a = []
    if row[0] != 'nan':
        for item in row:
            if str(item) != 'nan':
                a.append(item)
        if len(a) > 0:
            prescriptions.append(a)
```

Now we will use a 'onehot' encoder to encode the perscrition data as boolean.

[ ]:
```python
from mlxtend.preprocessing import TransactionEncoder

encoder = TransactionEncoder().fit(prescriptions)
onehot = encoder.transform(prescriptions)
onehot = pd.DataFrame(onehot, columns=encoder.columns_)
onehot.to_csv('data/market_basket_clean.csv')

onehot.head()
```

[ ]:
```
   Duloxetine  Premarin    Yaz  abilify  acetaminophen  actonel  \
0       False     False  False     True          False    False
1       False     False  False    False          False    False
2       False     False  False    False          False    False
```

```
3       False     False  False    False          False    False
4       False     False  False     True          False    False

   albuterol HFA  albuterol aerosol  alendronate  allopurinol  …  \
0           True               True        False         True  …
1          False              False        False        False  …
2          False              False        False        False  …
3          False              False        False         True  …
4          False              False        False        False  …

   trazodone HCI  triamcinolone Ace topical  triamterene  trimethoprim DS  \
0          False                      False        False            False
1          False                      False        False            False
2          False                      False        False            False
3          False                      False        False            False
4          False                      False        False            False

   valaciclovir  valsartan  venlafaxine XR  verapamil SR  viagra  zolpidem
0         False      False           False         False   False     False
1         False      False           False         False   False     False
2         False      False           False         False   False     False
3         False      False           False         False   False     False
4         False      False           False         False   False     False

[5 rows x 119 columns]
```

We will run the apriori algorithm to filter for frequent itemsets. We then print our table showing support, lift, and confidence values.

[ ]:
```python
from mlxtend.frequent_patterns import apriori, association_rules

# apriori
frequent_itemsets = apriori(onehot,
                            min_support=0.05,
                            max_len=3,
                            use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x:␣
 ↪len(x))
frequent_itemsets.sort_values('support', ascending=False, inplace=True)

rules = association_rules(frequent_itemsets,
                            metric="support",
                            min_threshold=.003)
rules.sort_values('support', ascending=False, inplace=True)

#show the top 3 rules
top3 = rules.head(3)
```

```
top3
```

[ ]:

	antecedents	consequents	antecedent support	consequent support
0	(abilify)	(carvedilol)	0.238368	0.174110
1	(carvedilol)	(abilify)	0.174110	0.238368
2	(abilify)	(diazepam)	0.238368	0.163845

	support	confidence	lift	leverage	conviction
0	0.059725	0.250559	1.439085	0.018223	1.102008
1	0.059725	0.343032	1.439085	0.018223	1.159314
2	0.052660	0.220917	1.348332	0.013604	1.073256

As we can see in the above table the top three rules are:

1. $\{abilify\}$ -> $\{carvedilol\}$
2. $\{carvedilol\}$ -> $\{abilify\}$
3. $\{diazepam\}$ -> $\{abilify\}$

4 Summary and Implications

4.1 Support

Support is a percentage showing the frequency that an item set (X, Y) is present in a full dataset N. The equation for support is:

support = frequency(X, Y) / N

If we look at the support of our top 3 pairings we see they are all over 0.05. this means that each is present in over 5% of all transactions.

[ ]:
```
top3[['antecedents', 'consequents', 'support']]
```

[ ]:

	antecedents	consequents	support
0	(abilify)	(carvedilol)	0.059725
1	(carvedilol)	(abilify)	0.059725
2	(abilify)	(diazepam)	0.052660

4.2 Confidence

Confidence measures how often one item predicts another. It measures the frequency of an item set (X, Y) over the number of times one of the items X or Y is present.

Confidence = frequency(X, Y) / Frequency(X)

Our confidence values are:

[ ]:
```
top3[['antecedents', 'consequents', 'confidence']]
```

[ ]:

	antecedents	consequents	confidence
0	(abilify)	(carvedilol)	0.250559
1	(carvedilol)	(abilify)	0.343032

```
2     (abilify)    (diazepam)    0.220917
```

This tells us that 25% of patients prescribed abilify are eventually prescribed carvedilol, 34% prescribed carvedilol are prescribed abilify, and 32% prescribed diazepam will be prescribed abilify.

4.3 Lift

Lift measures the "power" of a rule by comparing the combined support of an itemset with their individual supports.

Lift = support(X & Y) / (support(X) * support(Y))

If the lift is over 1.0 the rule is considered good, under 1.0 it is considered not as good. As we see below, all of our lifts are over 1.0.

[ ]:
```
top3[['antecedents', 'consequents', 'lift']]
```

[ ]:
```
    antecedents   consequents      lift
0     (abilify)  (carvedilol)  1.439085
1  (carvedilol)     (abilify)  1.439085
2     (abilify)    (diazepam)  1.348332
```

4.4 Significance of the findings and Recommendation

The drug Abilify is present in all of our top 3 rules, suggesting that abilify is commonly perscribed with other drugs. We also see that ability and carvedilol predict each other which suggests that the two are commonly prescribed in tandem.

It would be wise for healthcare providers to consider that abilify, an antidepressent, and carvedilol, a bloodpressure medication, are often prescribed in tandem. This could imply that mental health diseases such as depression or anxiety can also lead to blood pressure problems (or vice versa).