AssociationRuleMiningGroceries

November 25, 2025

[54]:
```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

[55]:
```
groceries = pd.read_csv("Groceries_dataset.csv")
display(groceries.head())
```

```
   Member_number        Date   itemDescription
0           1808  21-07-2015    tropical fruit
1           2552  05-01-2015        whole milk
2           2300  19-09-2015         pip fruit
3           1187  12-12-2015  other vegetables
4           3037  01-02-2015        whole milk
```

Group Items by Purchase (Customer on the Same Day)

[56]:
```
basket = groceries.groupby(['Member_number', 'Date'])['itemDescription'].
↪apply(list).reset_index()
purchase = basket['itemDescription'].tolist()
```

Convert to One-Hot Format to have True/False

[57]:
```
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_array = te.fit(purchase).transform(purchase)
groceries_encoded = pd.DataFrame(te_array, columns = te.columns_)
```

Run the algorithm to find combinations that occur in at least .5% (.005) of transactions

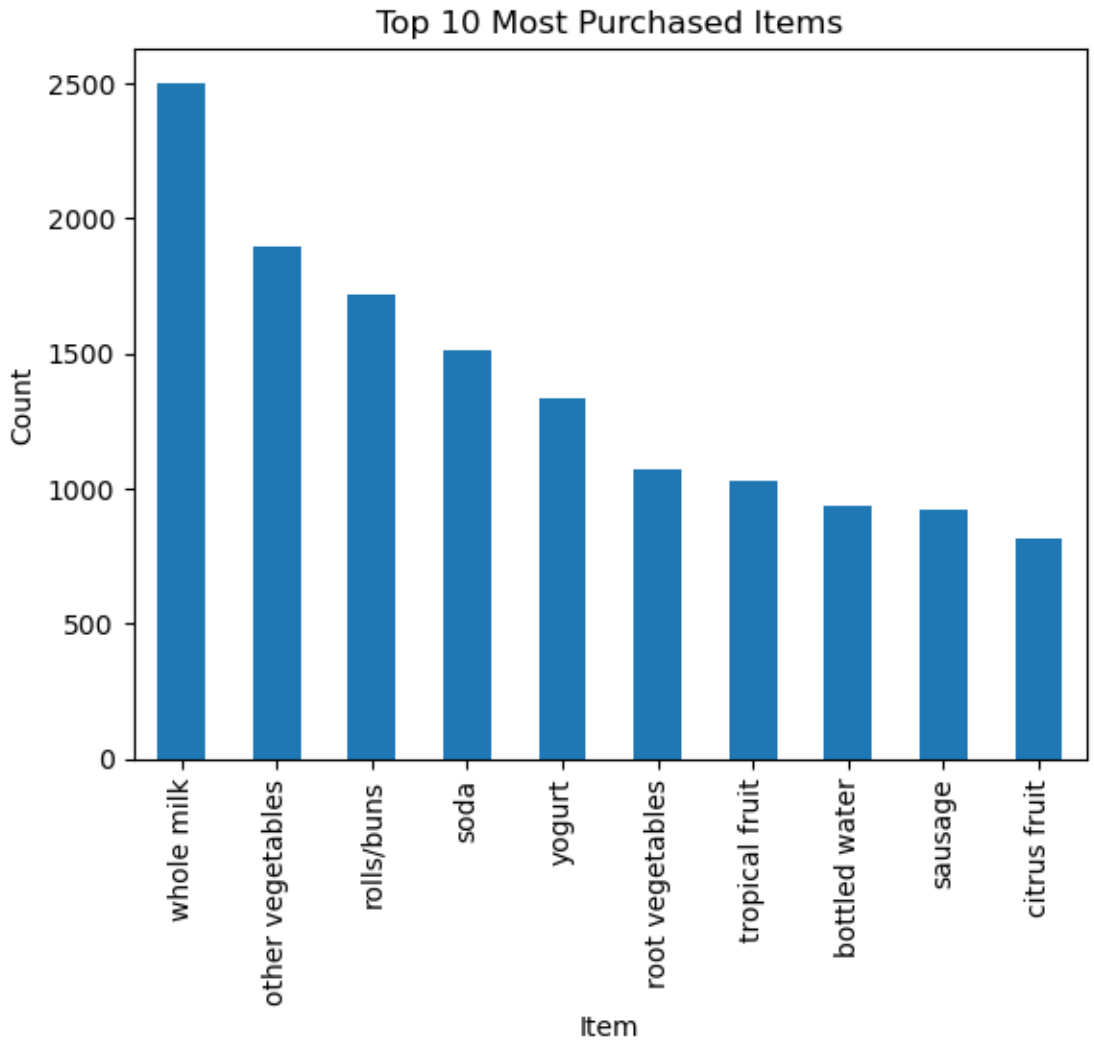
[58]:
```
from mlxtend.frequent_patterns import apriori
frequent_items = apriori(groceries_encoded, min_support = 0.005, use_colnames =␣
↪True)
print("Total Frequent Items Purchased:", frequent_items.shape[0])
```

```
Total Frequent Items Purchased: 126
```

Visualize some popular purchases

[64]:
```
import matplotlib.pyplot as plt
top_items = groceries['itemDescription'].value_counts().head(10)
top_items.plot(kind='bar', title='Top 10 Most Purchased Items')
```

```
plt.xlabel("Item")
plt.ylabel("Count")
plt.show()
```



Now, we create the association rules

[65]:
```
from mlxtend.frequent_patterns import association_rules
frequent_items = apriori(groceries_encoded,
                         min_support=0.003,
                         use_colnames=True)
rules = association_rules(frequent_items,
                          metric="confidence",
                          min_threshold=0.05)
rules = rules[(rules['lift'] > 1.2) &
```

```
              (rules['confidence'] > 0.1)]
rules = association_rules(frequent_items, metric="confidence", min_threshold=0.
 ↪1)
rules = rules[rules['antecedents'].apply(lambda x: len(x) >= 1) &␣
 ↪rules['consequents'].apply(lambda x: len(x) >= 1)]
print("Association Rules:", rules.shape[0])
rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head(5)
```

```
Association Rules: 39
```

[65]:

	antecedents	consequents	support	confidence	lift
0	(beef)	(whole milk)	0.004678	0.137795	0.872548
1	(bottled beer)	(other vegetables)	0.004678	0.103245	0.845568
2	(bottled beer)	(whole milk)	0.007151	0.157817	0.999330
3	(bottled water)	(whole milk)	0.007151	0.117841	0.746196
4	(brown bread)	(whole milk)	0.004478	0.119005	0.753566

[66]:
```
results = []

for idx, row in rules.iterrows():
    antecedent = list(row['antecedents'])
    consequent = list(row['consequents'])

    # Skip if either side is empty
    if len(antecedent) == 0 or len(consequent) == 0:
        continue

    print("Rule:", antecedent, "->", consequent)
    print("Support:", row['support'])
    print("Confidence:", row['confidence'])
    print("Lift:", row['lift'])
    print()

    results.append({
        "antecedent": antecedent,
        "consequent": consequent,
        "support": row['support'],
        "confidence": row['confidence'],
        "lift": row['lift']
    })
```

```
Rule: ['beef'] -> ['whole milk']
Support: 0.004678206242063757
Confidence: 0.1377952755905512
Lift: 0.8725479088706803

Rule: ['bottled beer'] -> ['other vegetables']
Support: 0.004678206242063757
```

```
Confidence: 0.10324483775811208
Lift: 0.8455678748629617

Rule: ['bottled beer'] -> ['whole milk']
Support: 0.007150972398583172
Confidence: 0.15781710914454278
Lift: 0.9993302598941151

Rule: ['bottled water'] -> ['whole milk']
Support: 0.007150972398583172
Confidence: 0.11784140969162996
Lift: 0.7461959429605837

Rule: ['brown bread'] -> ['whole milk']
Support: 0.004477711688832453
Confidence: 0.11900532859680285
Lift: 0.7535661158671016

Rule: ['butter'] -> ['whole milk']
Support: 0.004678206242063757
Confidence: 0.13282732447817835
Lift: 0.84108982486965

Rule: ['canned beer'] -> ['whole milk']
Support: 0.006014836596939116
Confidence: 0.1282051282051282
Lift: 0.8118211313302299

Rule: ['chicken'] -> ['whole milk']
Support: 0.003408407404932166
Confidence: 0.1223021582733813
Lift: 0.774442316650277

Rule: ['citrus fruit'] -> ['whole milk']
Support: 0.007150972398583172
Confidence: 0.13459119496855346
Lift: 0.8522590140983773

Rule: ['coffee'] -> ['whole milk']
Support: 0.003809396511394774
Confidence: 0.12050739957716701
Lift: 0.7630775369755185

Rule: ['curd'] -> ['other vegetables']
Support: 0.003542070440419702
Confidence: 0.10515873015873017
Lift: 0.861242517441204
```

```
Rule: ['curd'] -> ['whole milk']
Support: 0.004143554100113613
Confidence: 0.12301587301587302
Lift: 0.7789617045859112

Rule: ['domestic eggs'] -> ['whole milk']
Support: 0.005279689901757669
Confidence: 0.14234234234234236
Lift: 0.9013408668931312

Rule: ['frankfurter'] -> ['other vegetables']
Support: 0.005146026866270133
Confidence: 0.13628318584070798
Lift: 1.1161495948191098

Rule: ['frankfurter'] -> ['whole milk']
Support: 0.005279689901757669
Confidence: 0.13982300884955753
Lift: 0.8853879311959075

Rule: ['frozen vegetables'] -> ['other vegetables']
Support: 0.003141081333957094
Confidence: 0.11217183770883055
Lift: 0.9186793692595684

Rule: ['frozen vegetables'] -> ['whole milk']
Support: 0.003809396511394774
Confidence: 0.1360381861575179
Lift: 0.8614216586859672

Rule: ['fruit/vegetable juice'] -> ['rolls/buns']
Support: 0.003742564993651006
Confidence: 0.11001964636542241
Lift: 1.0001360683874942

Rule: ['fruit/vegetable juice'] -> ['whole milk']
Support: 0.004410880171088686
Confidence: 0.12966601178781928
Lift: 0.8210717454003978

Rule: ['hamburger meat'] -> ['whole milk']
Support: 0.0030742498162133263
Confidence: 0.14067278287461774
Lift: 0.890768874377023

Rule: ['margarine'] -> ['whole milk']
Support: 0.0040767225823698456
Confidence: 0.12655601659751037
```

```
Lift: 0.8013786188525382

Rule: ['newspapers'] -> ['whole milk']
Support: 0.005613847490476508
Confidence: 0.14432989690721648
Lift: 0.9139264694975372

Rule: ['pip fruit'] -> ['other vegetables']
Support: 0.004945532313038829
Confidence: 0.10081743869209808
Lift: 0.8256876492336418

Rule: ['pork'] -> ['other vegetables']
Support: 0.00394305954688231
Confidence: 0.10630630630630632
Lift: 0.870641084434188

Rule: ['shopping bags'] -> ['other vegetables']
Support: 0.004945532313038829
Confidence: 0.10393258426966293
Lift: 0.8512004698560297

Rule: ['other vegetables'] -> ['whole milk']
Support: 0.014836596939116486
Confidence: 0.12151067323481116
Lift: 0.7694304712706219

Rule: ['pastry'] -> ['whole milk']
Support: 0.006482657221145492
Confidence: 0.12532299741602068
Lift: 0.7935708888429613

Rule: ['pip fruit'] -> ['rolls/buns']
Support: 0.004945532313038829
Confidence: 0.10081743869209808
Lift: 0.9164831926791396

Rule: ['pip fruit'] -> ['whole milk']
Support: 0.006616320256633028
Confidence: 0.13487738419618528
Lift: 0.8540712229062719

Rule: ['pork'] -> ['whole milk']
Support: 0.005012363830782597
Confidence: 0.13513513513513514
Lift: 0.8557033546453776

Rule: ['rolls/buns'] -> ['whole milk']
```

```
Support: 0.013967787208447505
Confidence: 0.12697448359659783
Lift: 0.8040284376030019

Rule: ['root vegetables'] -> ['whole milk']
Support: 0.00755196150504578
Confidence: 0.10854947166186359
Lift: 0.6873574881406962

Rule: ['sausage'] -> ['whole milk']
Support: 0.008955423377664907
Confidence: 0.14839424141749724
Lift: 0.9396627314134623

Rule: ['shopping bags'] -> ['whole milk']
Support: 0.006348994185657956
Confidence: 0.13342696629213482
Lift: 0.8448868796568826

Rule: ['soda'] -> ['whole milk']
Support: 0.011628684087415625
Confidence: 0.11975223675154853
Lift: 0.7582956912879478

Rule: ['tropical fruit'] -> ['whole milk']
Support: 0.008220276682483459
Confidence: 0.121301775147929
Lift: 0.7681076857970638

Rule: ['whipped/sour cream'] -> ['whole milk']
Support: 0.004611374724319989
Confidence: 0.1055045871559633
Lift: 0.6680766557827672

Rule: ['white bread'] -> ['whole milk']
Support: 0.003141081333957094
Confidence: 0.1309192200557103
Lift: 0.829007316840285

Rule: ['yogurt'] -> ['whole milk']
Support: 0.011160863463209249
Confidence: 0.1299610894941634
Lift: 0.8229402378760758
```

Interpretation Support: How often the rule appears in the dataset (for example, the rule of if white bread are purchased, then whole milk is purchased occurs .003, or just 0.3%)

Confidence: Probability of buying the "consequent" item if the "antecedent" item is bought; it is the predictive strength of the rule (for example, the probability of buying whole milk if white bread is bought is .13, or 13%)

Lift: the strength of the rule (>1 means it is a positive association and is a good rule; for example, purchasing white bread slightly decreases the likelihood of pruchasing whole milk)

[89]:
```
readable_rules = []

for idx, row in rules.iterrows():
    antecedent = ', '.join(list(row['antecedents']))
    consequent = ', '.join(list(row['consequents']))

    support_percent = row['support'] * 100
    confidence_percent = row['confidence'] * 100
    lift = row['lift']

    if lift > 1:
        lift_desc = f"co-occurs MORE often than expected by chance (good rule)"
    elif lift < 1:
        lift_desc = f"co-occurs LESS often than expected by chance (negative␣
 ↪association)"
    else:
        lift_desc = f"occurs as expected by chance (no association)"

    readable_rules.append({
        'Rule': f"If someone buys [{antecedent}], they also buy [{consequent}]",
        'Support (%)': f"{support_percent:.2f}%",
        'Confidence (%)': f"{confidence_percent:.2f}%",
        'Lift': f"{lift:.2f} ({lift_desc})"
    })

readable_rules_df = pd.DataFrame(readable_rules)

pd.set_option('display.max_colwidth', None)

# Show more rows if needed
pd.set_option('display.max_rows', 100)

#Show the top 20 rules
readable_rules_df.head(20)
```

[89]:
```
                                                                  Rule  \
0                        If someone buys [beef], they also buy [whole milk]
1           If someone buys [bottled beer], they also buy [other vegetables]
2                 If someone buys [bottled beer], they also buy [whole milk]
3               If someone buys [bottled water], they also buy [whole milk]
4                 If someone buys [brown bread], they also buy [whole milk]
```

```
5                    If someone buys [butter], they also buy [whole milk]
6                If someone buys [canned beer], they also buy [whole milk]
7                    If someone buys [chicken], they also buy [whole milk]
8               If someone buys [citrus fruit], they also buy [whole milk]
9                    If someone buys [coffee], they also buy [whole milk]
10                If someone buys [curd], they also buy [other vegetables]
11                     If someone buys [curd], they also buy [whole milk]
12             If someone buys [domestic eggs], they also buy [whole milk]
13         If someone buys [frankfurter], they also buy [other vegetables]
14               If someone buys [frankfurter], they also buy [whole milk]
15  If someone buys [frozen vegetables], they also buy [other vegetables]
16         If someone buys [frozen vegetables], they also buy [whole milk]
17    If someone buys [fruit/vegetable juice], they also buy [rolls/buns]
18    If someone buys [fruit/vegetable juice], they also buy [whole milk]
19            If someone buys [hamburger meat], they also buy [whole milk]

   Support (%) Confidence (%)  \
0        0.47%          13.78%
1        0.47%          10.32%
2        0.72%          15.78%
3        0.72%          11.78%
4        0.45%          11.90%
5        0.47%          13.28%
6        0.60%          12.82%
7        0.34%          12.23%
8        0.72%          13.46%
9        0.38%          12.05%
10       0.35%          10.52%
11       0.41%          12.30%
12       0.53%          14.23%
13       0.51%          13.63%
14       0.53%          13.98%
15       0.31%          11.22%
16       0.38%          13.60%
17       0.37%          11.00%
18       0.44%          12.97%
19       0.31%          14.07%

                                                                                Lift
0   0.87 (co-occurs LESS often than expected by chance (negative association))
1   0.85 (co-occurs LESS often than expected by chance (negative association))
2   1.00 (co-occurs LESS often than expected by chance (negative association))
3   0.75 (co-occurs LESS often than expected by chance (negative association))
4   0.75 (co-occurs LESS often than expected by chance (negative association))
5   0.84 (co-occurs LESS often than expected by chance (negative association))
6   0.81 (co-occurs LESS often than expected by chance (negative association))
7   0.77 (co-occurs LESS often than expected by chance (negative association))
```

```
8   0.85 (co-occurs LESS often than expected by chance (negative association))
9   0.76 (co-occurs LESS often than expected by chance (negative association))
10  0.86 (co-occurs LESS often than expected by chance (negative association))
11  0.78 (co-occurs LESS often than expected by chance (negative association))
12  0.90 (co-occurs LESS often than expected by chance (negative association))
13            1.12 (co-occurs MORE often than expected by chance (good rule))
14  0.89 (co-occurs LESS often than expected by chance (negative association))
15  0.92 (co-occurs LESS often than expected by chance (negative association))
16  0.86 (co-occurs LESS often than expected by chance (negative association))
17            1.00 (co-occurs MORE often than expected by chance (good rule))
18  0.82 (co-occurs LESS often than expected by chance (negative association))
19  0.89 (co-occurs LESS often than expected by chance (negative association))
```

[87]:
```
top_rules = rules.sort_values(['lift', 'confidence'], ascending = False).
 ↪head(10)
top_rules
```

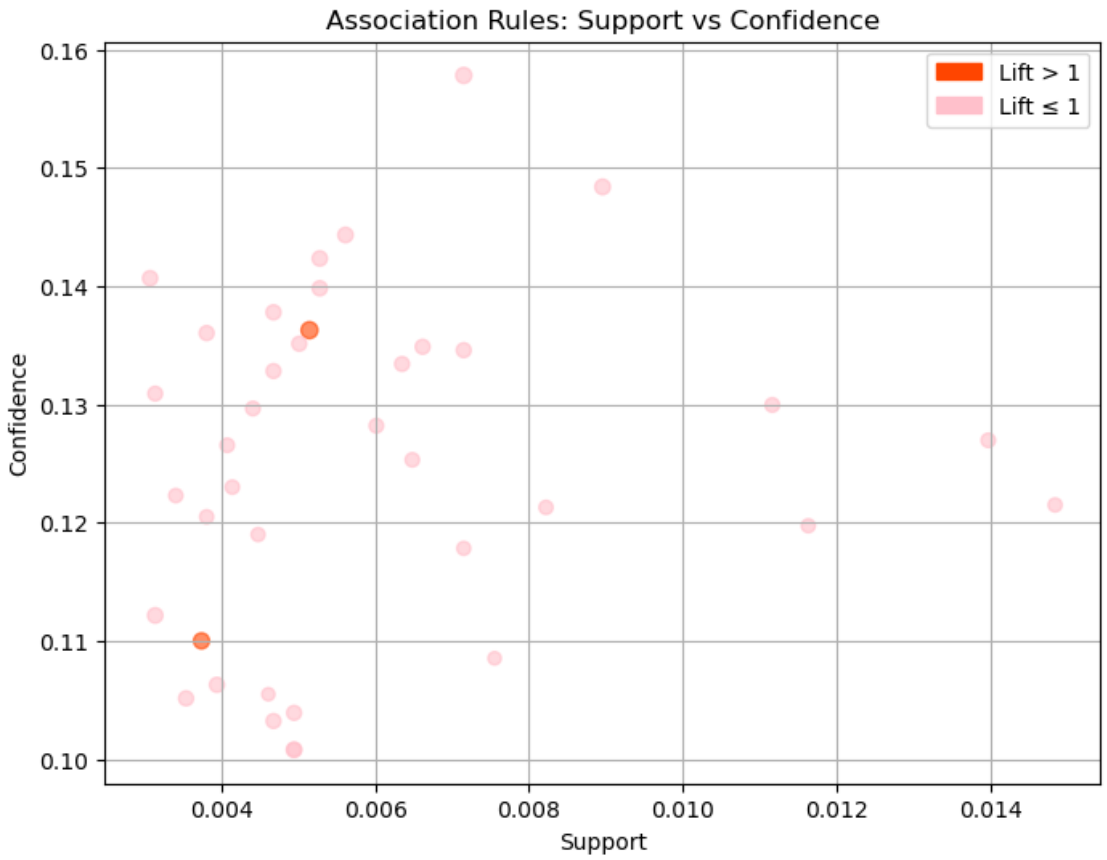
[87]:
```
                antecedents         consequents  antecedent support  \
13            (frankfurter)  (other vegetables)            0.037760
17  (fruit/vegetable juice)        (rolls/buns)            0.034017
2             (bottled beer)        (whole milk)            0.045312
32                (sausage)        (whole milk)            0.060349
15      (frozen vegetables)  (other vegetables)            0.028002
27              (pip fruit)        (rolls/buns)            0.049054
21             (newspapers)        (whole milk)            0.038896
12          (domestic eggs)        (whole milk)            0.037091
19         (hamburger meat)        (whole milk)            0.021854
14            (frankfurter)        (whole milk)            0.037760

    consequent support   support  confidence      lift  representativity  \
13            0.122101  0.005146    0.136283  1.116150               1.0
17            0.110005  0.003743    0.110020  1.000136               1.0
2             0.157923  0.007151    0.157817  0.999330               1.0
32            0.157923  0.008955    0.148394  0.939663               1.0
15            0.122101  0.003141    0.112172  0.918679               1.0
27            0.110005  0.004946    0.100817  0.916483               1.0
21            0.157923  0.005614    0.144330  0.913926               1.0
12            0.157923  0.005280    0.142342  0.901341               1.0
19            0.157923  0.003074    0.140673  0.890769               1.0
14            0.157923  0.005280    0.139823  0.885388               1.0

        leverage  conviction  zhangs_metric   jaccard  certainty  kulczynski
13  5.355097e-04    1.016420       0.108146  0.033261   0.016154    0.089214
17  5.091755e-07    1.000017       0.000141  0.026679   0.000017    0.072021
2  -4.792503e-06    0.999874      -0.000702  0.036469  -0.000126    0.101549
32 -5.750423e-04    0.988811      -0.063965  0.042784  -0.011316    0.102551
15 -2.780456e-04    0.988816      -0.083468  0.021373  -0.011310    0.068949
```

```
27 -4.506739e-04    0.989783      -0.087448  0.032090  -0.010323    0.072887
21 -5.287118e-04    0.984114      -0.089246  0.029360  -0.016142    0.089939
12 -5.779053e-04    0.981834      -0.102072  0.027827  -0.018503    0.087887
19 -3.769819e-04    0.979926      -0.111400  0.017398  -0.020485    0.080070
14 -6.834475e-04    0.978958      -0.118576  0.027729  -0.021494    0.086628
```

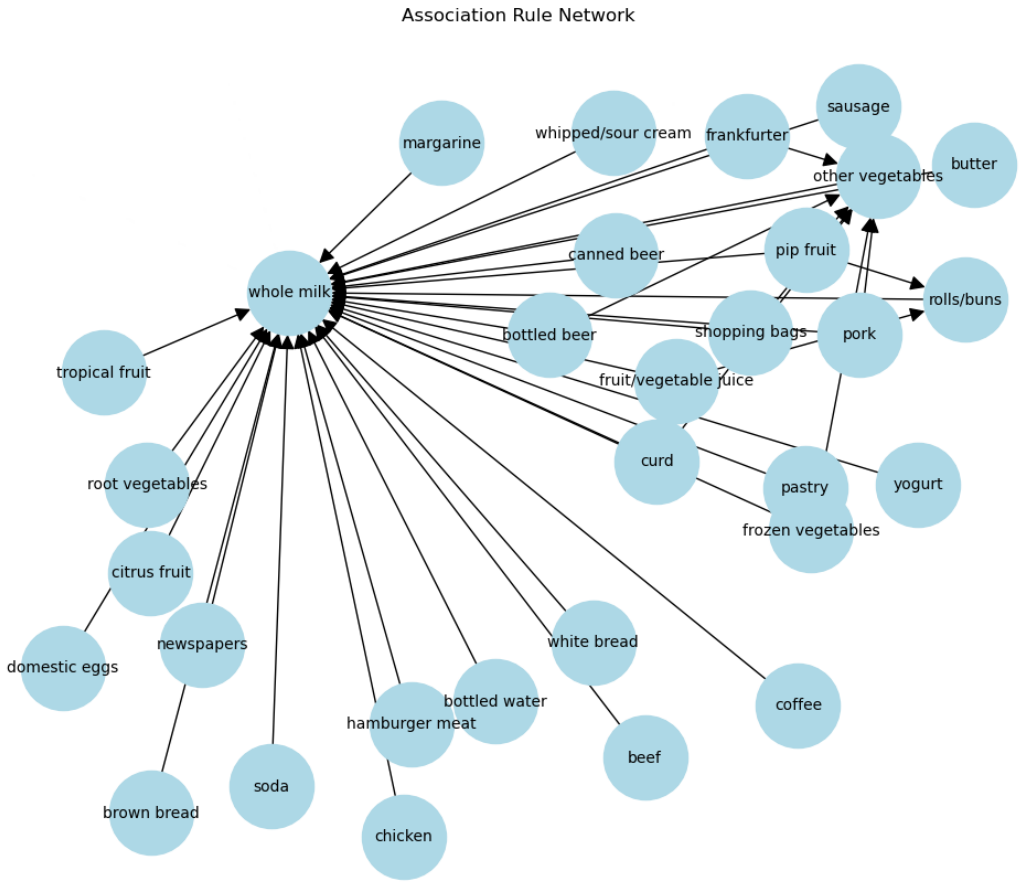
We can visualize support, confidence, and lift togethere.

Below shows us a scatter plot with support and confidence, colored by lift. The darker colors indicate non-random association rules.

[78]:
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches

lift_mask = rules['lift'] > 1
colors = np.where(lift_mask, 'orangered', 'pink') #orange indicates lift > 1,␣
 ↪pink is lift <= 1

plt.figure(figsize=(8,6))
scatter = plt.scatter(rules['support'], rules['confidence'],
                      s=rules['lift']*50, c = colors, alpha=0.6)

plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Association Rules: Support vs Confidence')
plt.grid(True)

orange_patch = mpatches.Patch(color='orangered', label='Lift > 1')
pink_patch = mpatches.Patch(color='pink', label='Lift   1')
plt.legend(handles=[orange_patch, pink_patch])

plt.show()
```

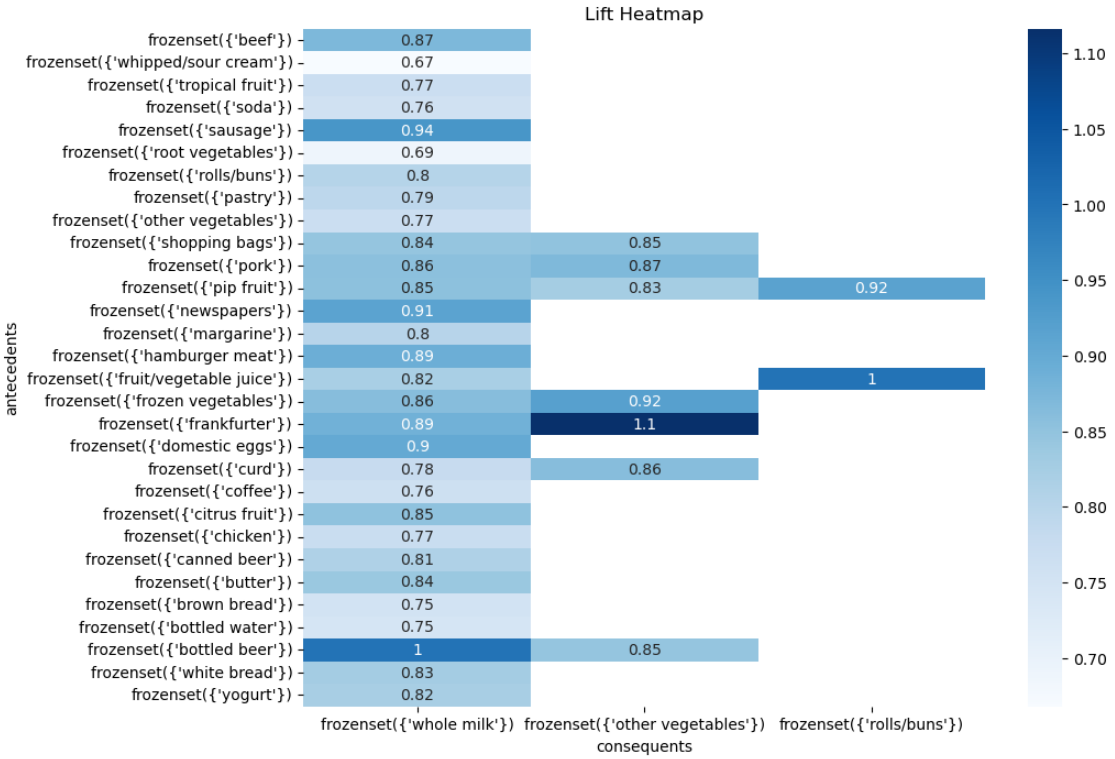


[79]:
```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()

for idx, row in rules.iterrows():
    for a in row['antecedents']:
        for c in row['consequents']:
            G.add_edge(a, c, weight=row['lift'])

plt.figure(figsize=(10,8))
pos = nx.spring_layout(G, k=0.5)

nx.draw(G, pos, with_labels=True,
        node_size=3000, node_color="lightblue",
        arrowsize=20, font_size=10)

plt.title('Association Rule Network')
plt.show()
```



We can also create a heatmap to show us the antecedents and consequents and their lift.

[80]:
```
import seaborn as sns
import pandas as pd

heatmap_data = rules.pivot_table(
    index='antecedents',
    columns='consequents',
    values='lift'
)

plt.figure(figsize=(10,8))
sns.heatmap(heatmap_data, annot=True, cmap='Blues')
plt.title('Lift Heatmap')
plt.show()
```


Lift Heatmap
frozenset({'beef'}) 0.87
frozenset({'whipped/sour cream'}) 0.67
frozenset({'tropical fruit'}) 0.77
frozenset({'soda'}) 0.76
frozenset({'sausage'}) 0.94
frozenset({'root vegetables'}) 0.69
frozenset({'rolls/buns'}) 0.8
frozenset({'pastry'}) 0.79
frozenset({'other vegetables'}) 0.77
frozenset({'shopping bags'}) 0.84 0.85
frozenset({'pork'}) 0.86 0.87
frozenset({'pip fruit'}) 0.85 0.83 0.92
frozenset({'newspapers'}) 0.91
frozenset({'margarine'}) 0.8
frozenset({'hamburger meat'}) 0.89
frozenset({'fruit/vegetable juice'}) 0.82 1
frozenset({'frozen vegetables'}) 0.86 0.92
frozenset({'frankfurter'}) 0.89 1.1
frozenset({'domestic eggs'}) 0.9
frozenset({'curd'}) 0.78 0.86
frozenset({'coffee'}) 0.76
frozenset({'citrus fruit'}) 0.85
frozenset({'chicken'}) 0.77
frozenset({'canned beer'}) 0.81
frozenset({'butter'}) 0.84
frozenset({'brown bread'}) 0.75
frozenset({'bottled water'}) 0.75
frozenset({'bottled beer'}) 1 0.85
frozenset({'white bread'}) 0.83
frozenset({'yogurt'}) 0.82
antecedents
frozenset({'whole milk'})
frozenset({'other vegetables'})
frozenset({'rolls/buns'})
consequents
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70