In [ ]:

```
Chapter 6 - Other Popular Machine Learning Methods

Part 1 - Association Rule Mining Using Apriori Algorithm

Import the required libraries
```

In [1]:

```
! pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.17.2-py2.py3-none-any.whl (1.3 MB)
Requirement already satisfied: scipy>=1.2.1 in c:\users\danal\anaconda3\lib\sit
e-packages (from mlxtend) (1.4.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\danal\anaconda3\lib\si
te-packages (from mlxtend) (1.18.1)
Requirement already satisfied: joblib>=0.13.2 in c:\users\danal\anaconda3\lib\s
ite-packages (from mlxtend) (0.14.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\danal\anaconda3\li
b\site-packages (from mlxtend) (3.1.3)
Requirement already satisfied: pandas>=0.24.2 in c:\users\danal\anaconda3\lib\s
ite-packages (from mlxtend) (1.0.1)
Requirement already satisfied: setuptools in c:\users\danal\anaconda3\lib\site-
packages (from mlxtend) (45.2.0.post20200210)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\danal\anaconda3
\lib\site-packages (from mlxtend) (0.22.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\danal\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\danal\anaconda3\lib\sit
e-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\danal\anaconda3
\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\u
sers\danal\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.4.
6)
Requirement already satisfied: pytz>=2017.2 in c:\users\danal\anaconda3\lib\sit
e-packages (from pandas>=0.24.2->mlxtend) (2019.3)
Requirement already satisfied: six in c:\users\danal\anaconda3\lib\site-package
s (from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.14.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.17.2
```

In [2]:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
Data Format
```

In [3]:

```
address = 'C:/Users/danal/Desktop/Ex_Files_Python_Data_Science_EssT_Pt2/Exercise
data = pd.read_csv(address)
```

In [4]:
```
data.head()
```

Out[4]:

	1	2	3	4	5	6	7	8	9
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN

```
Data Conversion
```

In [6]:
```
basket_sets = pd.get_dummies(data)
```

In [7]:
```
basket_sets.head()
```

Out[7]:

	1_Instant food products	1_UHT-milk	1_artif. sweetener	1_baby cosmetics	1_bags	1_baking powder	1_bathroom cleaner	1_beef	1_berries	1_I
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

5 rows × 1113 columns

```
Support Calculation
```

In [8]:

```
apriori(basket_sets, min_support=0.02)
```

Out[8]:

	support	itemsets
0	0.030421	(7)
1	0.034951	(17)
2	0.029126	(23)
3	0.049191	(26)
4	0.064401	(47)
5	0.044660	(83)
6	0.024272	(90)
7	0.040453	(92)
8	0.038835	(99)
9	0.033981	(100)
10	0.076052	(105)
11	0.028803	(111)
12	0.044984	(123)
13	0.073463	(130)
14	0.022977	(131)
15	0.028803	(159)
16	0.058900	(217)
17	0.022977	(224)
18	0.040129	(232)
19	0.036893	(233)
20	0.031068	(243)
21	0.034628	(256)
22	0.062136	(263)
23	0.028479	(264)
24	0.045955	(351)
25	0.033010	(366)
26	0.024272	(378)
27	0.057929	(397)
28	0.023301	(398)
29	0.020712	(479)
30	0.024595	(497)
31	0.024272	(510)
32	0.033333	(531)
33	0.023301	(532)

	support	itemsets
34	0.020065	(631)
35	0.021036	(217, 397)

In [9]:

```
apriori(basket_sets, min_support=0.02, use_colnames=True)
```

Out[9]:

	support	itemsets
0	0.030421	(1_beef)
1	0.034951	(1_canned beer)
2	0.029126	(1_chicken)
3	0.049191	(1_citrus fruit)
4	0.064401	(1_frankfurter)
5	0.044660	(1_other vegetables)
6	0.024272	(1_pip fruit)
7	0.040453	(1_pork)
8	0.038835	(1_rolls/buns)
9	0.033981	(1_root vegetables)
10	0.076052	(1_sausage)
11	0.028803	(1_soda)
12	0.044984	(1_tropical fruit)
13	0.073463	(1_whole milk)
14	0.022977	(1_yogurt)
15	0.028803	(2_citrus fruit)
16	0.058900	(2_other vegetables)
17	0.022977	(2_pip fruit)
18	0.040129	(2_rolls/buns)
19	0.036893	(2_root vegetables)
20	0.031068	(2_soda)
21	0.034628	(2_tropical fruit)
22	0.062136	(2_whole milk)
23	0.028479	(2_yogurt)
24	0.045955	(3_other vegetables)
25	0.033010	(3_rolls/buns)
26	0.024272	(3_soda)
27	0.057929	(3_whole milk)
28	0.023301	(3_yogurt)
29	0.020712	(4_other vegetables)
30	0.024595	(4_rolls/buns)
31	0.024272	(4_soda)
32	0.033333	(4_whole milk)
33	0.023301	(4_yogurt)

	support	itemsets
34	0.020065	(5_rolls/buns)
35	0.021036	(2_other vegetables, 3_whole milk)

In [11]:
```
df = basket_sets

frequent_itemsets = apriori(basket_sets, min_support=0.002, use_colnames=True)

frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x
frequent_itemsets
```

Out[11]:

	support	itemsets	length
0	0.006472	(1_UHT-milk)	1
1	0.030421	(1_beef)	1
2	0.011974	(1_berries)	1
3	0.008414	(1_beverages)	1
4	0.014887	(1_bottled beer)	1
...	...	...	...
844	0.002265	(5_other vegetables, 3_pip fruit, 6_whole milk)	3
845	0.002589	(5_whole milk, 4_other vegetables, 3_root vege...	3
846	0.002913	(5_yogurt, 4_curd, 3_whole milk)	3
847	0.003236	(4_root vegetables, 5_other vegetables, 6_whol...	3
848	0.002265	(5_other vegetables, 6_whole milk, 7_butter)	3

849 rows × 3 columns

In [12]:
```
frequent_itemsets[frequent_itemsets['length'] >= 3]
```

Out[12]:

	support	itemsets	length
820	0.002589	(2_root vegetables, 3_other vegetables, 1_beef)	3
821	0.002589	(2_other vegetables, 1_chicken, 3_whole milk)	3
822	0.002589	(2_other vegetables, 1_citrus fruit, 3_whole m...	3
823	0.003236	(2_tropical fruit, 3_pip fruit, 1_citrus fruit)	3
824	0.002589	(3_other vegetables, 4_whole milk, 1_citrus fr...	3
825	0.002265	(5_other vegetables, 1_frankfurter, 6_whole milk)	3
826	0.002265	(1_pork, 4_whole milk, 3_other vegetables)	3
827	0.003560	(1_root vegetables, 2_other vegetables, 3_whol...	3
828	0.002589	(1_sausage, 3_soda, 2_rolls/buns)	3
829	0.002265	(1_sausage, 4_whole milk, 3_other vegetables)	3
830	0.002265	(5_whole milk, 1_sausage, 4_other vegetables)	3
831	0.002913	(2_other vegetables, 3_whole milk, 1_tropical ...	3
832	0.002265	(5_whole milk, 4_other vegetables, 2_citrus fr...	3
833	0.002265	(4_butter, 2_other vegetables, 3_whole milk)	3
834	0.003560	(4_curd, 2_other vegetables, 3_whole milk)	3
835	0.003883	(4_yogurt, 2_other vegetables, 3_whole milk)	3
836	0.002265	(6_rolls/buns, 2_other vegetables, 3_whole milk)	3
837	0.003236	(2_pip fruit, 4_whole milk, 3_other vegetables)	3
838	0.005825	(2_root vegetables, 4_whole milk, 3_other vege...	3
839	0.002265	(4_other vegetables, 2_tropical fruit, 3_pip f...	3
840	0.003560	(5_butter, 4_whole milk, 3_other vegetables)	3
841	0.002913	(5_yogurt, 4_whole milk, 3_other vegetables)	3
842	0.003560	(6_yogurt, 4_whole milk, 3_other vegetables)	3
843	0.002265	(4_root vegetables, 5_other vegetables, 3_pip ...	3
844	0.002265	(5_other vegetables, 3_pip fruit, 6_whole milk)	3
845	0.002589	(5_whole milk, 4_other vegetables, 3_root vege...	3
846	0.002913	(5_yogurt, 4_curd, 3_whole milk)	3
847	0.003236	(4_root vegetables, 5_other vegetables, 6_whol...	3
848	0.002265	(5_other vegetables, 6_whole milk, 7_butter)	3

```
Association Rules

Confidence
```

In [13]:
```
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=
rules.head()
```

Out[13]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(2_sausage)	(1_frankfurter)	0.011327	0.064401	0.011327	1.000000	15.527638	0.010597
1	(7_pastry)	(1_frankfurter)	0.005178	0.064401	0.002589	0.500000	7.763819	0.002256
2	(2_ham)	(1_sausage)	0.007120	0.076052	0.004531	0.636364	8.367505	0.003989
3	(2_meat)	(1_sausage)	0.006796	0.076052	0.004854	0.714286	9.392097	0.004338
4	(3_beef)	(1_sausage)	0.004854	0.076052	0.002589	0.533333	7.012766	0.002220

Lift

In [15]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

Out[15]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(2_citrus fruit)	(1_beef)	0.028803	0.030421	0.005502	0.191011	6.278986	0.004625
1	(1_beef)	(2_citrus fruit)	0.030421	0.028803	0.005502	0.180851	6.278986	0.004625
2	(2_other vegetables)	(1_beef)	0.058900	0.030421	0.003236	0.054945	1.806173	0.001444
3	(1_beef)	(2_other vegetables)	0.030421	0.058900	0.003236	0.106383	1.806173	0.001444
4	(2_root vegetables)	(1_beef)	0.036893	0.030421	0.005502	0.149123	4.902016	0.004379

Lift and Confidence

In [16]:
```
rules[(rules['lift'] >= 5) & (rules['confidence']>= 0.5)]
```

Out[16]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
92	(2_sausage)	(1_frankfurter)	0.011327	0.064401	0.011327	1.000000	15.527638	0.01059
137	(7_pastry)	(1_frankfurter)	0.005178	0.064401	0.002589	0.500000	7.763819	0.00225
239	(2_ham)	(1_sausage)	0.007120	0.076052	0.004531	0.636364	8.367505	0.00398
243	(2_meat)	(1_sausage)	0.006796	0.076052	0.004854	0.714286	9.392097	0.00433
258	(3_beef)	(1_sausage)	0.004854	0.076052	0.002589	0.533333	7.012766	0.00222
...	...	...	...	...	...	...	...	.
958	(4_root vegetables, 5_other vegetables)	(6_whole milk)	0.005178	0.009385	0.003236	0.625000	66.594828	0.00318
959	(4_root vegetables, 6_whole milk)	(5_other vegetables)	0.003883	0.012621	0.003236	0.833333	66.025641	0.00318
965	(5_other vegetables, 7_butter)	(6_whole milk)	0.002589	0.009385	0.002265	0.875000	93.232759	0.00224
966	(6_whole milk, 7_butter)	(5_other vegetables)	0.002913	0.012621	0.002265	0.777778	61.623932	0.00222
969	(7_butter)	(5_other vegetables, 6_whole milk)	0.004207	0.007443	0.002265	0.538462	72.341137	0.00223

76 rows × 9 columns