Development Part 2

Using Association Rules:

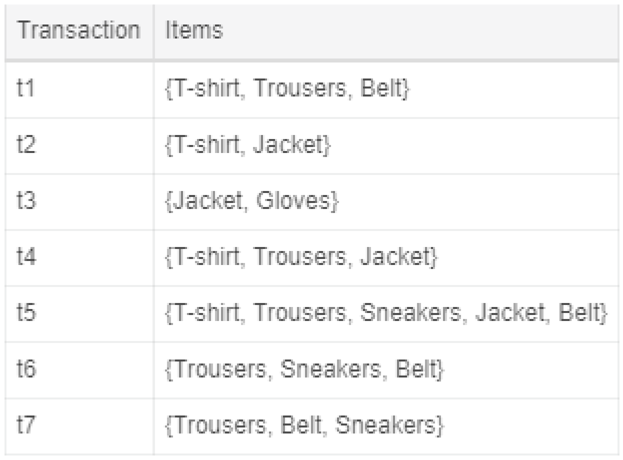
Association Rules are widely used to analyse retail basket or transaction data, and are intended to identify strong rules discovered in transaction data using measures of interestingness, based on the concept of strong rules. The outcome of this type of technique is, in simple terms, a set of **rules** that can be understood as **“if this, then that”**.

An example of Association Rules

- Assume there are 100 customers
- 10 of them bought milk, 8 bought butter and 6 bought both of them.
- bought milk => bought butter
- support = P(Milk & Butter) = 6/100 = 0.06
- confidence = support/P(Butter) = 0.06/0.08 = 0.75
- lift = confidence/P(Milk) = 0.75/0.10 = 7.5
- Note: This example is extremely small. In practice, a rule needs the support of several hundred transactions, before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

Association rules

The Apriori algorithm generates association rules for a given data set. An association rule implies that if an item A occurs, then item B also occurs with a certain probability. Let's see an example,



Transaction	Items
t1	{T-shirt, Trousers, Belt}
t2	{T-shirt, Jacket}
t3	{Jacket, Gloves}
t4	{T-shirt, Trousers, Jacket}
t5	{T-shirt, Trousers, Sneakers, Jacket, Belt}
t6	{Trousers, Sneakers, Belt}
t7	{Trousers, Belt, Sneakers}

In the table above we can see seven transactions from a clothing store. Each transaction shows items bought in that transaction. We can represent our items as an **item set** as follows:

$$I = \{i_1, i_2, \ldots, i_k\}$$

In our case it corresponds to:

$$I = \{T\text{-}shirt, Trousers, Belt, Jacket, Gloves, Sneakers\}$$

A **transaction** is represented by the following expression:

$$T = \{t_1, t_2, \ldots, t_n\}$$

For example,

$$t_1 = \{T\text{-}shirt, Trousers, Belt\}$$

Then, an **association rule** is defined as an implication of the form:

$$X \Rightarrow Y \text{, where } X \subset I, Y \subset I \text{ and } X \cap Y = 0$$

For example,

$$\{T\text{-}shirt, Trousers\} \Rightarrow \{Belt\}$$

In the following sections we are going to define four metrics to measure the precision of a rule.

Support

Support is an indication of how frequently the item set appears in the data set.

$$supp(X \Rightarrow Y) = \frac{|X \cup Y|}{n}$$

In other words, it's the number of transactions with both X and Y divided by the total number of transactions. The rules are not useful for low support values. Let's see different examples using the clothing store transactions from the previous table.

- $supp(T\text{-}shirt \Rightarrow Trousers) = \frac{3}{7} = 43\%$
- $supp(Trousers \Rightarrow Belt) = \frac{4}{7} = 57\%$
- $supp(T\text{-}shirt \Rightarrow Belt) = \frac{2}{7} = 28\%$
- $supp(\{T\text{-}shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{2}{7} = 28\%$

Confidence

For a rule $X \Rightarrow Y$, confidence shows the percentage in which Y is bought with X. It's an indication of how often the rule has been found to be true.

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

For example, the rule $T\text{-}shirt \Rightarrow Trousers$ has a confidence of 3/4, which means that for 75% of the transactions containing a t-shirt the rule is correct (75% of the times a customer buys a t-shirt, trousers are bought as well). Three more examples:

- $conf(Trousers \Rightarrow Belt) = \frac{4/7}{5/7} = 80\%$
- $conf(T\text{-}shirt \Rightarrow Belt) = \frac{2/7}{4/7} = 50\%$
- $conf(\{T\text{-}shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{2/7}{3/7} = 66\%$

Lift

The lift of a rule is the ratio of the observed support to that expected if X and Y were independent, and is defined as

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)}$$

Greater lift values indicate stronger associations. Let's see some examples:

- $lift(T\text{-}shirt \Rightarrow Trousers) = \frac{3/7}{(4/7)(5/7)} = 1.05$
- $lift(Trousers \Rightarrow Belt) = \frac{4/7}{(5/7)(4/7)} = 1.4$
- $lift(T\text{-}shirt \Rightarrow Belt) = \frac{2/7}{(4/7)(4/7)} = 0.875$
- $lift(\{T\text{-}shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{2/7}{(3/7)(4/7)} = 1.17$

Conviction

The conviction of a rule is defined as

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)}$$

It can be interpreted as the ratio of the expected frequency that X occurs without Y if X and Y were independent divided by the observed frequency of incorrect predictions. A high value means that the consequent depends strongly on the antecedent. Let's see some examples:

- $conv(T\text{-}shirt \Rightarrow Trousers) = \frac{1 - 5/7}{1 - 3/4} = 1.14$
- $conv(Trousers \Rightarrow Belt) = \frac{1 - 4/7}{1 - 4/5} = 2.14$
- $conv(T\text{-}shirt \Rightarrow Belt) = \frac{1 - 4/7}{1 - 1/2} = 0.86$
- $conv(\{T\text{-}shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{1 - 4/7}{1 - 2/3} = 1.28$

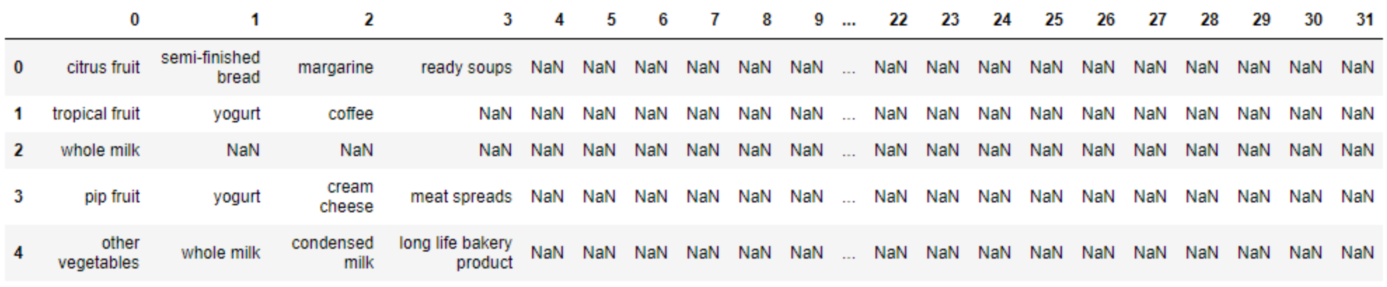
LET'S CODE

Importing Dependencies

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

Loading CSV file

```
df=pd.read_csv(r'groceries.csv',header=None)

df.head()
```



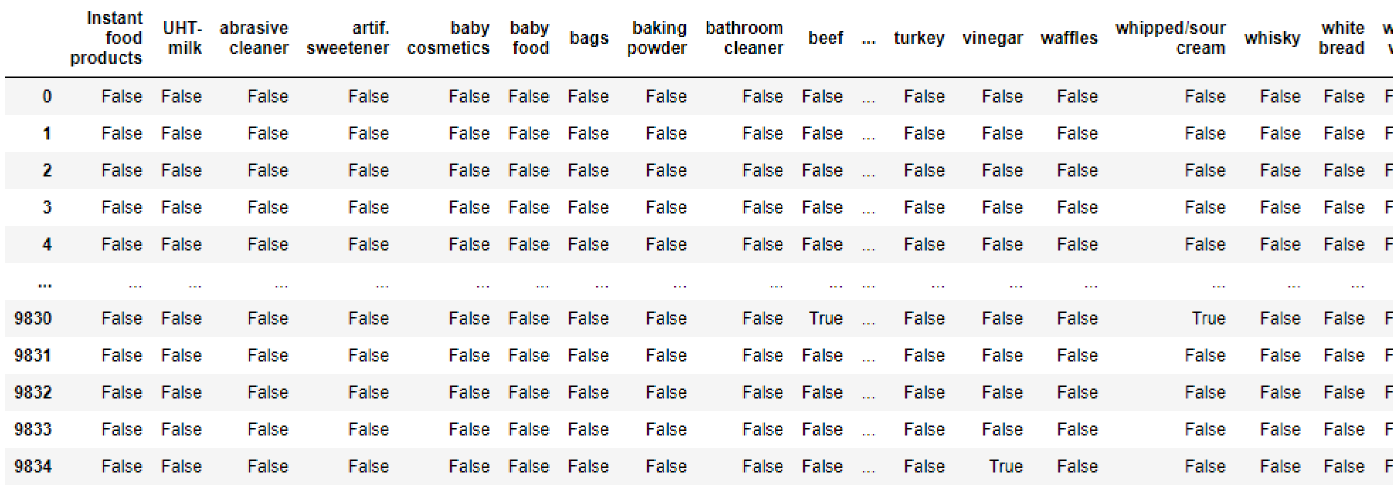
	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30	31
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Data Preparation

True if transaction occur and false if not

records = []
for i in range (0, 9835):
records.append([str(df.values[i,j]) for j in range(0, 20)])

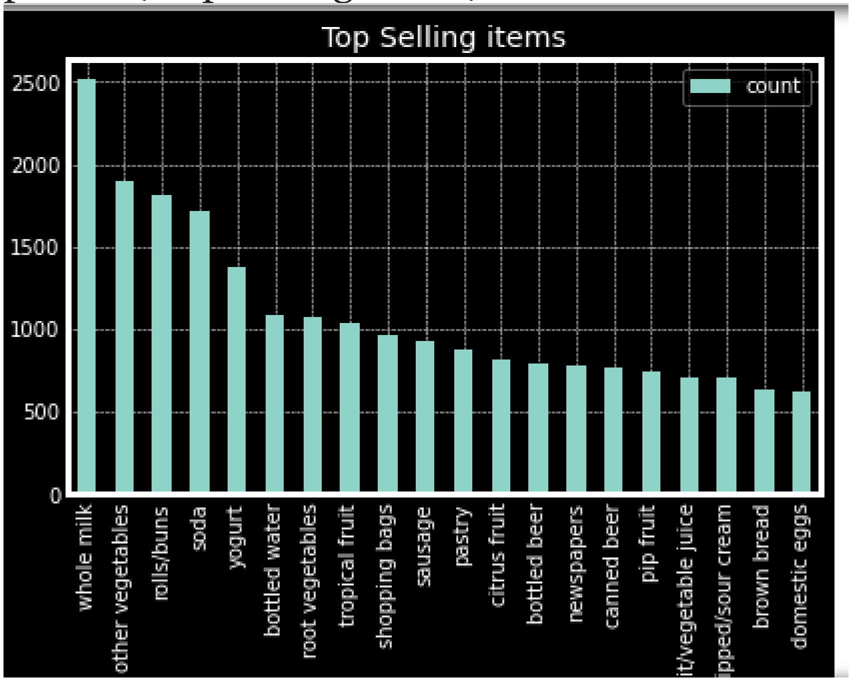
TE = TransactionEncoder()
array = TE.fit(records).transform(records)
#building the data frame rows are logical and columns are the items have been purchased
df1 = pd.DataFrame(array, columns = TE.columns_)
df1



	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef	...	turkey	vinegar	waffles	whipped/sour cream	whisky	white bread	w v
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9830	False	False	False	False	False	False	False	False	False	True	...	False	False	False	True	False	False	F
9831	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
9832	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
9833	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	F
9834	False	False	False	False	False	False	False	False	False	False	...	False	True	False	False	False	False	F

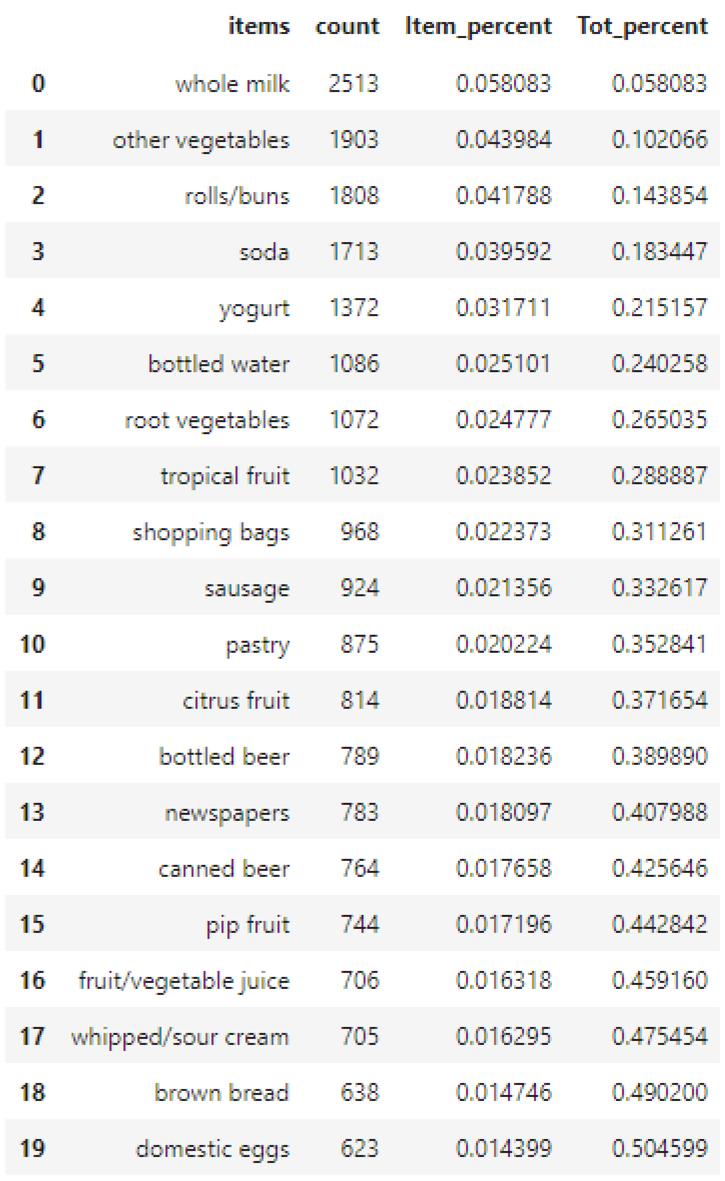
Now Visualize Top 20 Selling Items

```
count = df_clean.loc[:,:].sum()
df2 = count.sort_values(0, ascending = False).head(20)
df2 = df2.to_frame()
df2 = df2.reset_index()
df2 = df2.rename(columns = {“index”: “items”,0: “count”})

plt.style.use(‘dark_background’)
ax = df2.plot.bar(x = ‘items’, y = ‘count’)
plt.title(‘Top Selling items’)
```



Now find the item percentage and cummulative percentage

```
tot_item = sum(df_clean.sum())

df2[‘Item_percent’] = df2[‘count’]/tot_item
df2[‘Tot_percent’] = df2.Item_percent.cumsum()
df2.head(20)
```



	items	count	Item_percent	Tot_percent
0	whole milk	2513	0.058083	0.058083
1	other vegetables	1903	0.043984	0.102066
2	rolls/buns	1808	0.041788	0.143854
3	soda	1713	0.039592	0.183447
4	yogurt	1372	0.031711	0.215157
5	bottled water	1086	0.025101	0.240258
6	root vegetables	1072	0.024777	0.265035
7	tropical fruit	1032	0.023852	0.288887
8	shopping bags	968	0.022373	0.311261
9	sausage	924	0.021356	0.332617
10	pastry	875	0.020224	0.352841
11	citrus fruit	814	0.018814	0.371654
12	bottled beer	789	0.018236	0.389890
13	newspapers	783	0.018097	0.407988
14	canned beer	764	0.017658	0.425646
15	pip fruit	744	0.017196	0.442842
16	fruit/vegetable juice	706	0.016318	0.459160
17	whipped/sour cream	705	0.016295	0.475454
18	brown bread	638	0.014746	0.490200
19	domestic eggs	623	0.014399	0.504599

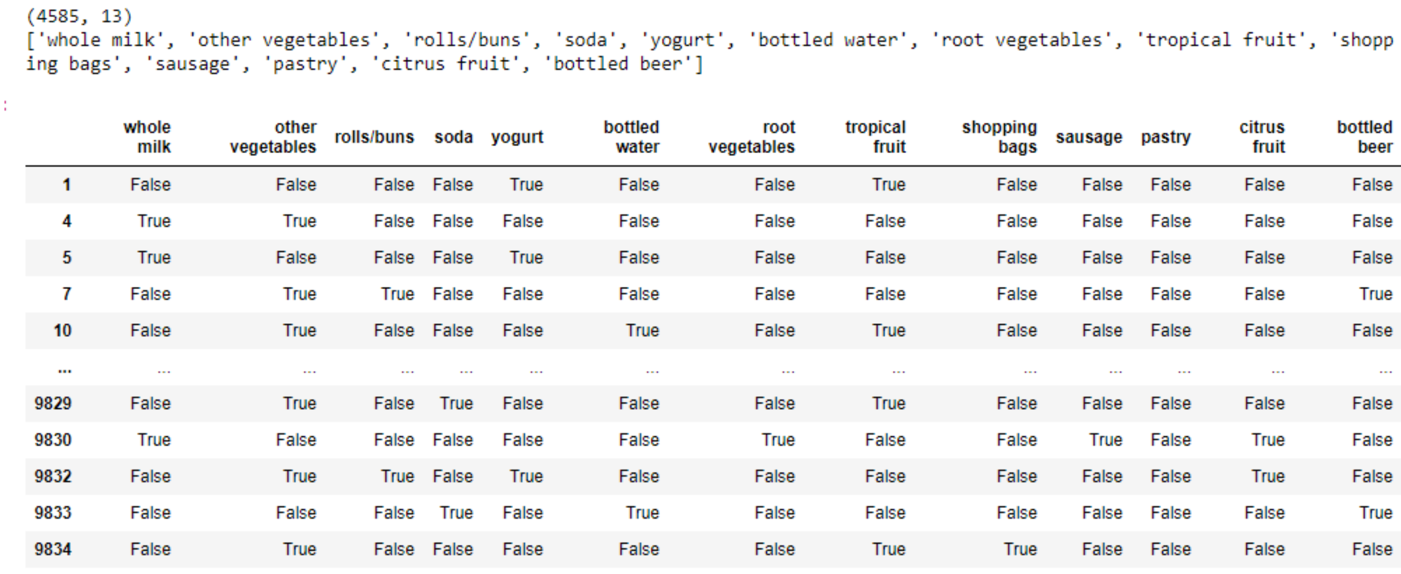
as we can see 50% of sold items are top 20 items. so now we will remove less frequently sold items

```
def prune_dataset(olddf, len_transaction, tot_sales_percent):
# Delete the last column tot_items if present
if ‘tot_items’ in olddf.columns:
del(olddf[‘tot_items’])
#Finding the item_count for each item and total number of items.
#This is the same code as in step 3
```

```
Item_count = olddf.sum().sort_values(ascending =
False).reset_index()
tot_items = sum(olddf.sum().sort_values(ascending = False))
Item_count.rename(columns={Item_count.columns[0]:’Item_nam
e’,Item_count.columns[1]:’Item_count’}, inplace=True)

# Code from Step 3 to find Item Percentage and Total Percentage.
Item_count[‘Item_percent’] = Item_count[‘Item_count’]/tot_items
Item_count[‘Tot_percent’] = Item_count.Item_percent.cumsum()

# Taking items that fit the condition/ minimum threshold for total
sales percentage.
selected_items = list(Item_count[Item_count.Tot_percent <
tot_sales_percent].Item_name)
olddf[‘tot_items’] = olddf[selected_items].sum(axis = 1)

# Taking items that fit the condition/ minimum threshold for length
of transaction or number of items in a row.
olddf = olddf[olddf.tot_items >= len_transaction]
del(olddf[‘tot_items’])

#Return pruned dataframe.
return olddf[selected_items], Item_count[Item_count.Tot_percent
< tot_sales_percent]

output_df, item_counts = prune_dataset(df_clean,
2,0.4)
print(output def shape)
```

print(list(output def columns))

output def



```
(4585, 13)
['whole milk', 'other vegetables', 'rolls/buns', 'soda', 'yogurt', 'bottled water', 'root vegetables', 'tropical fruit', 'shopp
ing bags', 'sausage', 'pastry', 'citrus fruit', 'bottled beer']
```

	whole milk	other vegetables	rolls/buns	soda	yogurt	bottled water	root vegetables	tropical fruit	shopping bags	sausage	pastry	citrus fruit	bottled beer
1	False	False	False	False	True	False	False	True	False	False	False	False	False
4	True	True	False	False	False	False	False	False	False	False	False	False	False
5	True	False	False	False	True	False	False	False	False	False	False	False	False
7	False	True	True	False	False	False	False	False	False	False	False	False	True
10	False	True	False	False	False	True	False	True	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9829	False	True	False	True	False	False	False	True	False	False	False	False	False
9830	True	False	False	False	False	False	True	False	False	True	False	True	False
9832	False	True	True	False	True	False	False	False	False	False	False	True	False
9833	False	False	False	True	False	True	False	False	False	False	False	False	True
9834	False	True	False	False	False	False	False	True	True	False	False	False	False

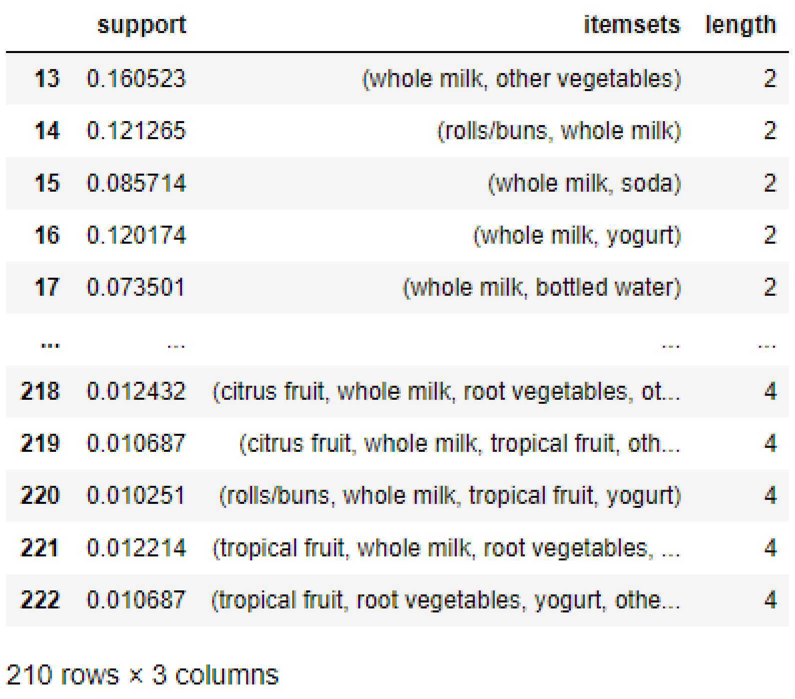
Now we use a priori algorithm

Frequent item sets = a priori(output def, min support=0.01, use col names=True)

frequent item sets['length'] = frequent item sets['item sets'].apply(lambda x:len(x))

frequent item sets=frequent item sets[(frequent item sets['length'] >= 2)]

frequent item sets



	support	itemsets	length
13	0.160523	(whole milk, other vegetables)	2
14	0.121265	(rolls/buns, whole milk)	2
15	0.085714	(whole milk, soda)	2
16	0.120174	(whole milk, yogurt)	2
17	0.073501	(whole milk, bottled water)	2
...	...	...	...
218	0.012432	(citrus fruit, whole milk, root vegetables, ot...	4
219	0.010687	(citrus fruit, whole milk, tropical fruit, oth...	4
220	0.010251	(rolls/buns, whole milk, tropical fruit, yogurt)	4
221	0.012214	(tropical fruit, whole milk, root vegetables, ...	4
222	0.010687	(tropical fruit, root vegetables, yogurt, othe...	4

210 rows × 3 columns

Association Rules

we want antecedent length more then 2 , confidence more then or equal to 0.3 , lift greater or equal 1and support greater or equal to 0.04.

```
rules_mlxtend[“antecedent_len”] =
rules_mlxtend[“antecedents”].apply(lambda x: len(x))

rules_mlxtend[ (rules_mlxtend[‘antecedent_len’] >= 2) &
(rules_mlxtend[‘confidence’] >= 0.3) &
(rules_mlxtend[‘lift’] >= 1) &
(rules_mlxtend[‘support’]>=0.04)]
```

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
(whole milk, yogurt)	(other vegetables)	0.120174	0.352454	0.047764	0.397459	1.127692	0.005409	1.074693	2
(yogurt, other vegetables)	(whole milk)	0.093130	0.442748	0.047764	0.512881	1.158403	0.006531	1.143974	2
(whole milk, root vegetables)	(other vegetables)	0.104907	0.352454	0.049727	0.474012	1.344893	0.012752	1.231106	2
(whole milk, other vegetables)	(root vegetables)	0.160523	0.207852	0.049727	0.309783	1.490402	0.016362	1.147679	2
(root vegetables, other vegetables)	(whole milk)	0.101636	0.442748	0.049727	0.489270	1.105076	0.004728	1.091090	2

Conclusion

- The most popular item in this data set is whole milk followed by vegetables and rolls/buns.
- By applying the Apriori algorithm and association rules we can have a better insight on what items are more likely to be bought together.