Recommender System Using Apriori

Import library

In [1]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

Data preprocessing

In [31]:
```
ds = pd.read_csv('Groceries_dataset.csv')
ds.head()
```

Out[31]:

	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

In [32]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    38765 non-null  int64 
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

In [33]:
```
ds.isna().any()
```

Out[33]:
```
Member_number      False
Date               False
itemDescription    False
dtype: bool
```

In [34]:
```
ds['customerID'] = ds.groupby(['Member_number','Date']).ngroup()
ds = ds.sort_values('customerID')
ds.head()
```

Out[34]:

	Member_number	Date	itemDescription	customerID
20992	1000	15-03-2015	semi-finished bread	0
8395	1000	15-03-2015	whole milk	0
24544	1000	15-03-2015	yogurt	0
4843	1000	15-03-2015	sausage	0
32851	1000	24-06-2014	salty snack	1

In [70]:
```
transaction = [list(ds[ds['customerID'] == i]['itemDescription']) for i in set(ds.customerID)]
transaction[:10]
```

Out[70]:
```
[['semi-finished bread', 'whole milk', 'yogurt', 'sausage'],
 ['salty snack', 'whole milk', 'pastry'],
 ['canned beer', 'misc. beverages'],
 ['sausage', 'hygiene articles'],
 ['pickled vegetables', 'soda'],
 ['curd', 'frankfurter'],
 ['sausage', 'rolls/buns', 'whole milk'],
 ['soda', 'whole milk'],
 ['white bread', 'beef'],
 ['frankfurter', 'soda', 'whipped/sour cream']]
```

Create recommender system from the dataset

In [65]:
```
from apyori import apriori

rules = apriori(transactions = transaction, min_support = 0.001, min_confidence = 0.01, max_length = 3)
results = list(rules)
results
```

```
 RelationRecord(items=frozenset({'yogurt', 'butter milk'}), support=0.0012697988371315912, ordered_statistics=[Order
edStatistic(items_base=frozenset({'butter milk'}), items_add=frozenset({'yogurt'}), confidence=0.07224334600760456,
 lift=0.8412273823438031), OrderedStatistic(items_base=frozenset({'yogurt'}), items_add=frozenset({'butter milk'}),
 confidence=0.014785992217898832, lift=0.8412273823438031)]),
 RelationRecord(items=frozenset({'citrus fruit', 'candy'}), support=0.0010024727661565194, ordered_statistics=[Order
edStatistic(items_base=frozenset({'candy'}), items_add=frozenset({'citrus fruit'}), confidence=0.06976744186046512,
 lift=1.3131197893813076), OrderedStatistic(items_base=frozenset({'citrus fruit'}), items_add=frozenset({'candy'}),
 confidence=0.018867924528301886, lift=1.3131197893813076)]),
 RelationRecord(items=frozenset({'other vegetables', 'candy'}), support=0.0011361358016440553, ordered_statistics=[O
rderedStatistic(items_base=frozenset({'candy'}), items_add=frozenset({'other vegetables'}), confidence=0.07906976744
186046, lift=0.6475757691475413)]),
 RelationRecord(items=frozenset({'rolls/buns', 'candy'}), support=0.0014702933903628951, ordered_statistics=[Ordered
Statistic(items_base=frozenset({'candy'}), items_add=frozenset({'rolls/buns'}), confidence=0.10232558139534884, lift
=0.9301929978241826), OrderedStatistic(items_base=frozenset({'rolls/buns'}), items_add=frozenset({'candy'}), confide
nce=0.013365735115431349, lift=0.9301929978241826)]),
 RelationRecord(items=frozenset({'soda', 'candy'}), support=0.0012697988371315912, ordered_statistics=[OrderedStatis
tic(items_base=frozenset({'candy'}), items_add=frozenset({'soda'}), confidence=0.08837209302325581, lift=0.910056178
8761025), OrderedStatistic(items_base=frozenset({'soda'}), items_add=frozenset({'candy'}), confidence=0.013076393668
27254, lift=0.9100561788761025)]),
 RelationRecord(items=frozenset({'whole milk', 'candy'}), support=0.002138608567800575, ordered_statistics=[OrderedS
```

In [66]:
```
def inspect(results):
    items       = [tuple(result[0]) for result in results]
    supports    = [result[1] for result in results]
    confidences = [result[2][0][2] for result in results]
    lifts       = [result[2][0][3] for result in results]
    return list(zip(items, supports, confidences, lifts))

resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Items combination', 'Support', 'Confidence', 'Lift'])
```

In [67]:
```
resultsinDataFrame
```

Out[67]:

	Items combination	Support	Confidence	Lift
0	(UHT-milk,)	0.021386	0.021386	1.000000
1	(beef,)	0.033950	0.033950	1.000000
2	(berries,)	0.021787	0.021787	1.000000
3	(beverages,)	0.016574	0.016574	1.000000
4	(bottled beer,)	0.045312	0.045312	1.000000
...	...	...	...	...
660	(rolls/buns, whole milk, sausage)	0.001136	0.010328	1.153275
661	(rolls/buns, whole milk, soda)	0.001002	0.010323	0.739091
662	(rolls/buns, yogurt, whole milk)	0.001337	0.012151	1.088685
663	(whole milk, soda, sausage)	0.001069	0.017719	1.523708
664	(yogurt, whole milk, sausage)	0.001470	0.024363	2.182917

665 rows × 4 columns

In [68]:
```
resultsinDataFrame.nlargest(n = 10, columns = 'Lift')
```

Out[68]:

	Items combination	Support	Confidence	Lift
664	(yogurt, whole milk, sausage)	0.001470	0.024363	2.182917
301	(citrus fruit, specialty chocolate)	0.001403	0.026415	1.653762
373	(tropical fruit, flour)	0.001069	0.109589	1.617141
109	(beverages, sausage)	0.001537	0.092742	1.536764
663	(whole milk, soda, sausage)	0.001069	0.017719	1.523708
487	(napkins, pastry)	0.001738	0.078550	1.518529
576	(processed cheese, root vegetables)	0.001069	0.105263	1.513019
439	(hard cheese, pip fruit)	0.001069	0.072727	1.482586
635	(yogurt, soft cheese)	0.001270	0.126667	1.474952
339	(curd, sausage)	0.002941	0.087302	1.446615

In [ ]:
```
```