

```
##Using Bread Basket Dataset develop a Association Rule Mining using
Apriori and FPGrowth algorithms.
```

```
In [1]: import pandas as pd
import numpy as np

from mlxtend.frequent_patterns import apriori, association_rules
```

```
In [2]: bbdata = pd.read_excel('BreadBasket.xlsx', names = ['Tx', 'products'])
bbdata.head(5)
```

```
Out[2]:
```

	Tx	products
0	0	MILK,BREAD,BISCUIT
1	1	BREAD,MILK,BISCUIT,CORNFLAKES
2	2	BREAD,TEA,BOURNVITA
3	3	JAM,MAGGI,BREAD,MILK
4	4	MAGGI,TEA,BISCUIT

```
In [3]: bbdata.shape
```

```
Out[3]: (20, 2)
```

```
In [4]: dataset = list(bbdata["products"].apply(lambda x:x.split(",") ))
dataset
```

```
Out[4]: [['MILK', 'BREAD', 'BISCUIT'],
['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
['BREAD', 'TEA', 'BOURNVITA'],
['JAM', 'MAGGI', 'BREAD', 'MILK'],
['MAGGI', 'TEA', 'BISCUIT'],
['BREAD', 'TEA', 'BOURNVITA'],
['MAGGI', 'TEA', 'CORNFLAKES'],
['MAGGI', 'BREAD', 'TEA', 'BISCUIT'],
['JAM', 'MAGGI', 'BREAD', 'TEA'],
['BREAD', 'MILK'],
['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
['COFFEE', 'SUGER', 'BOURNVITA'],
['BREAD', 'COFFEE', 'COCK'],
['BREAD', 'SUGER', 'BISCUIT'],
['COFFEE', 'SUGER', 'CORNFLAKES'],
['BREAD', 'SUGER', 'BOURNVITA'],
['BREAD', 'COFFEE', 'SUGER'],
['BREAD', 'COFFEE', 'SUGER'],
['TEA', 'MILK', 'COFFEE', 'CORNFLAKES']]
```

Apriori Algorithm

```
In [5]: from mlxtend.preprocessing import TransactionEncoder
```

```
te = TransactionEncoder()  
te_ary = te.fit(dataset).transform(dataset)  
te_ary
```

```
Out[5]: array([[ True, False,  True, False, False, False, False, False,  True,  
                False, False],  
              [ True, False,  True, False, False,  True, False, False,  True,  
                False, False],  
              [False,  True,  True, False, False, False, False, False, False,  
                False,  True],  
              [False, False,  True, False, False, False,  True,  True,  True,  
                False, False],  
              [ True, False, False, False, False, False, False,  True, False,  
                False,  True],  
              [False,  True,  True, False, False, False, False, False, False,  
                False,  True],  
              [False, False, False, False, False,  True, False,  True, False,  
                False,  True],  
              [ True, False,  True, False, False, False, False,  True, False,  
                False,  True],  
              [False, False,  True, False, False, False,  True,  True, False,  
                False,  True],  
              [False, False,  True, False, False, False, False, False,  True,  
                False, False],  
              [ True, False, False,  True,  True,  True, False, False, False,  
                False, False],  
              [ True, False, False,  True,  True,  True, False, False, False,  
                False, False],  
              [False,  True, False, False,  True, False, False, False, False,  
                True, False],  
              [False, False,  True,  True,  True, False, False, False, False,  
                False, False],  
              [ True, False,  True, False, False, False, False, False, False,  
                True, False],  
              [False, False, False, False,  True,  True, False, False, False,  
                True, False],  
              [False,  True,  True, False, False, False, False, False, False,  
                True, False],  
              [False, False,  True, False,  True, False, False, False, False,  
                True, False],  
              [False, False,  True, False,  True, False, False, False, False,  
                True, False],  
              [False, False, False, False,  True,  True, False, False,  True,  
                False,  True]])
```

```
In [6]: df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

```
Out[6]:
```

	BISCUIT	BOURNVITA	BREAD	COCK	COFFEE	CORNFLAKES	JAM	MAGGI	MILK	SUGAR
0	True	False	True	False	False	False	False	False	True	False
1	True	False	True	False	False	True	False	False	True	False
2	False	True	True	False	False	False	False	False	False	False
3	False	False	True	False	False	False	True	True	True	False
4	True	False	False	False	False	False	False	True	False	False
5	False	True	True	False	False	False	False	False	False	False
6	False	False	False	False	False	True	False	True	False	False
7	True	False	True	False	False	False	False	True	False	False
8	False	False	True	False	False	False	True	True	False	False
9	False	False	True	False	False	False	False	False	True	False
10	True	False	False	True	True	True	False	False	False	False
11	True	False	False	True	True	True	False	False	False	False
12	False	True	False	False	True	False	False	False	False	True
13	False	False	True	True	True	False	False	False	False	False
14	True	False	True	False	False	False	False	False	False	True
15	False	False	False	False	True	True	False	False	False	True
16	False	True	True	False	False	False	False	False	False	True
17	False	False	True	False	True	False	False	False	False	True
18	False	False	True	False	True	False	False	False	False	True
19	False	False	False	False	True	True	False	False	True	False



```
In [7]: from mlxtend.frequent_patterns import apriori  
apriori(df, min_support=0.2)
```

Out[7]:

	support	itemsets
0	0.35	(0)
1	0.20	(1)
2	0.65	(2)
3	0.40	(4)
4	0.30	(5)
5	0.25	(7)
6	0.25	(8)
7	0.30	(9)
8	0.35	(10)
9	0.20	(0, 2)
10	0.20	(8, 2)
11	0.20	(9, 2)
12	0.20	(2, 10)
13	0.20	(4, 5)
14	0.20	(9, 4)
15	0.20	(10, 7)

```
In [8]: apriori(df, min_support=0.2, use_colnames=True)
```

```
Out[8]:
```

	support	itemsets
0	0.35	(BISCUIT)
1	0.20	(BOURNVITA)
2	0.65	(BREAD)
3	0.40	(COFFEE)
4	0.30	(CORNFLAKES)
5	0.25	(MAGGI)
6	0.25	(MILK)
7	0.30	(SUGER)
8	0.35	(TEA)
9	0.20	(BISCUIT, BREAD)
10	0.20	(MILK, BREAD)
11	0.20	(SUGER, BREAD)
12	0.20	(TEA, BREAD)
13	0.20	(CORNFLAKES, COFFEE)
14	0.20	(SUGER, COFFEE)
15	0.20	(MAGGI, TEA)

```
In [9]: frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

```
Out[9]:
```

	support	itemsets	length
0	0.35	(BISCUIT)	1
1	0.20	(BOURNVITA)	1
2	0.65	(BREAD)	1
3	0.40	(COFFEE)	1
4	0.30	(CORNFLAKES)	1
5	0.25	(MAGGI)	1
6	0.25	(MILK)	1
7	0.30	(SUGER)	1
8	0.35	(TEA)	1
9	0.20	(BISCUIT, BREAD)	2
10	0.20	(MILK, BREAD)	2
11	0.20	(SUGER, BREAD)	2
12	0.20	(TEA, BREAD)	2
13	0.20	(CORNFLAKES, COFFEE)	2
14	0.20	(SUGER, COFFEE)	2
15	0.20	(MAGGI, TEA)	2

```
In [10]: frequent_itemsets[ (frequent_itemsets['length'] == 2) &
(frequent_itemsets['support'] >= 0.2) ]
```

```
Out[10]:
```

	support	itemsets	length
9	0.2	(BISCUIT, BREAD)	2
10	0.2	(MILK, BREAD)	2
11	0.2	(SUGER, BREAD)	2
12	0.2	(TEA, BREAD)	2
13	0.2	(CORNFLAKES, COFFEE)	2
14	0.2	(SUGER, COFFEE)	2
15	0.2	(MAGGI, TEA)	2

```
In [11]: frequent_itemsets[ frequent_itemsets['itemsets'] == {'BREAD', 'MILK'} ]
```

```
Out[11]:
```

	support	itemsets	length
10	0.2	(MILK, BREAD)	2

```
In [12]: from mlxtend.frequent_patterns import association_rules
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)
```

```
Out[12]:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.800000	1.230769	0.03
1	(SUGER)	(BREAD)	0.30	0.65	0.2	0.666667	1.025641	0.00
2	(CORNFLAKES)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.08
3	(SUGER)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.08
4	(MAGGI)	(TEA)	0.25	0.35	0.2	0.800000	2.285714	0.11

FP Growth Algorithm

```
In [13]: from mlxtend.frequent_patterns import fpgrowth
fpgrowth(df, min_support=0.3)
```

```
Out[13]:
```

	support	itemsets
0	0.65	(2)
1	0.35	(0)
2	0.30	(5)
3	0.35	(10)
4	0.40	(4)
5	0.30	(9)

```
In [14]: fpgrowth(df, min_support=0.3, use_colnames=True)
```

```
Out[14]:
```

	support	itemsets
0	0.65	(BREAD)
1	0.35	(BISCUIT)
2	0.30	(CORNFLAKES)
3	0.35	(TEA)
4	0.40	(COFFEE)
5	0.30	(SUGER)

```
In [15]: from mlxtend.frequent_patterns import association_rules
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
```

```
Out[15]:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.8	1.230769	0.0375
1	(MAGGI)	(TEA)	0.25	0.35	0.2	0.8	2.285714	0.1125

Comparative Study of Apriori and FP Growth

```
In [16]: import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

```
In [17]: from mlxtend.frequent_patterns import apriori

%timeit -n 100 -r 10 apriori(df, min_support=0.6)
```

978 μ s \pm 72.4 μ s per loop (mean \pm std. dev. of 10 runs, 100 loops each)

```
In [18]: %timeit -n 100 -r 10 apriori(df, min_support=0.6, low_memory=True)
```

1 ms \pm 41 μ s per loop (mean \pm std. dev. of 10 runs, 100 loops each)

```
In [19]: from mlxtend.frequent_patterns import fpgrowth

%timeit -n 100 -r 10 fpgrowth(df, min_support=0.6)
```

662 μ s \pm 120 μ s per loop (mean \pm std. dev. of 10 runs, 100 loops each)