

```
In [2]: # Importing libraries
```

```
In [3]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

```
In [4]: # Loading the dataset
```

```
In [5]: store_data = pd.read_csv('I:\\Datasets\\store_data.csv')
```

```
In [6]: #checking the dataset
```

```
In [7]: #check its dimension
store_data.shape
```

```
Out[7]: (7500, 20)
```

```
In [8]: #check null values
store_data.isnull()
```

```
Out[8]:
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	g
0	False	False	False	True	True	True	True	True	True	True	True	
1	False	True	True	True	True	True	True	True	True	True	True	
2	False	False	True	True	True	True	True	True	True	True	True	
3	False	False	False	False	False	True	True	True	True	True	True	
4	False	True	True	True	True	True	True	True	True	True	True	
...	...	...	...	...	...	...	...	...	...	...	...	
7495	False	False	False	True	True	True	True	True	True	True	True	
7496	False	False	False	False	False	False	True	True	True	True	True	
7497	False	True	True	True	True	True	True	True	True	True	True	
7498	False	False	True	True	True	True	True	True	True	True	True	
7499	False	False	False	False	True	True	True	True	True	True	True	

7500 rows × 20 columns



```
In [28]: #count null values and convert them to NaN default value
store_data.isnull().sum()
```

```
Out[28]: 0          0
1       1754
2       3112
3       4156
4       4972
5       5637
6       6132
7       6520
8       6847
9       7106
10      7245
11      7347
12      7414
13      7454
14      7476
15      7493
16      7497
17      7497
18      7498
dtype: int64
```

```
In [10]: #for our processing we do not need a header row
```

```
In [11]: store_data = pd.read_csv('I:\\Datasets\\store_data.csv', header=None)
```

```
In [12]: store_data.head()
```

```
Out[12]:
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
In [13]: #as we can see from the result the 20th column has all null values so we can elim
store_data.drop(store_data.columns[[19]], axis=1, inplace=True)
```


```
In [14]: store_data.isnull().sum()
```

```
Out[14]: 0      0
1     1754
2     3112
3     4156
4     4972
5     5637
6     6132
7     6520
8     6847
9     7106
10    7245
11    7347
12    7414
13    7454
14    7476
15    7493
16    7497
17    7497
18    7498
dtype: int64
```

```
In [15]: store_data.head()
```

```
Out[15]:
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN



```
In [17]: #Data Proprocessing
#The Apriori Library we are going to use requires our dataset to be in the form of a list
#where the whole dataset is a big list and each transaction in the dataset is an element in the list
#Currently we have data in the form of a pandas dataframe.
#To convert our pandas dataframe into a list of lists, we execute the script below
```

```
In [20]: records = []
for i in range(0, 7501):
    records.append([str(store_data.values[i,j]) for j in range(0, 19)])
```

```
In [21]: #Applying Apriori
# The first parameter is the list of list that you want to extract rules from.
# The second parameter is the min_support parameter.
# This parameter is used to select the items with support values greater than the
# Next, the min_confidence parameter filters those rules that have confidence greater than the
# Similarly, the min_lift parameter specifies the minimum lift value for the short rules
# Finally, the min_length parameter specifies the minimum number of items that you want to extract
```

```
In [22]: # Let's suppose that we want rules for only those items that are purchased at least 3 times
# since our dataset is for a one-week time period.
# The support for those items can be calculated as 35/7500 = 0.0045.
# The minimum confidence for the rules is 20% or 0.2.
# Similarly, we specify the value for lift as 3 and finally min_length is 2 since we want to extract rules with at least 2 items
```

```
In [23]: association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
```

```
In [24]: # Viewing the Results
```

```
In [25]: association_results = list(association_rules)
```

```
In [26]: print(association_results)

[RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.84395061728395)]), RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}), support=0.005732568990801226, ordered_statistics=[OrderedStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), confidence=0.3006993006993007, lift=3.790832696715049)]), RelationRecord(items=frozenset({'pasta', 'escalope'}), support=0.005865884548726837, ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lift=4.700811850163794)]), RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, ordered_statistics=[OrderedStatistic(items_base=frozenset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence=0.3234501347708895, lift=3.2919938411349285)]), RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}), support=0.005332622317024397, ordered_statistics=[OrderedStatistic(items_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}), confidence=0.3773584905660377, lift=3.840659481324083)]), RelationRecord(items=frozenset({'olive oil', 'whole wheat pasta'}), support=0.007998933475536596, ordered_statistics=[OrderedStatistic(items_base=frozenset({'olive oil'}), items_add=frozenset({'whole wheat pasta'}), confidence=0.3773584905660377, lift=3.840659481324083)])]
```

```
In [27]: print(association_results[0])

RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.84395061728395)])
```

```
In [18]: for item in association_results:

    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====")
```

```
Rule: chicken -> light cream
Support: 0.004532728969470737
Confidence: 0.29059829059829057
Lift: 4.84395061728395
=====
Rule: mushroom cream sauce -> escalope
Support: 0.005732568990801226
Confidence: 0.3006993006993007
Lift: 3.790832696715049
=====
Rule: pasta -> escalope
Support: 0.005865884548726837
Confidence: 0.3728813559322034
Lift: 4.700811850163794
=====
Rule: herb & pepper -> ground beef
Support: 0.015997866951073192
Confidence: 0.3234501347708895
Lift: 3.2919938411349285
```

In [ ]: