Assignment Project Report

Association Rule Mining: Market Basket Analysis

Name: Pavan Tiwari
Course: AI and ML
(Batch 4)

- **Problem Statement**

  Apriori is a statistical algorithm for implementing associate rule mining, that primarily relies on three components: Life, Support and Confidence. Using this algorithm try to find the rules that describe the relation between each of the products that were brought by the customers as described in

  Prerequisites

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)
  - Tools:
    - Pandas
    - Numpy
    - Matplotlib
    - Seaborn
  - Dataset: : Store Data
    https://drive.google.com/file/d/1y5DYn0dGoSbC22xowBq2d4po6h1JxcTQ/view?usp=sharing

- **Method Used**

  Association Rule Mining is used when we want to find an association between different objects in a set or find frequent patterns in a transaction database or relational databases. The applications of Association Rule Mining are found in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering and classification. It can be used to find what items do customers frequently buy together by generating a set of rules called Association Rules.

- **Implementation:**

1. Load all required libraries and Dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('store_data.csv')
data.head()
```

2. Preprocessing data

In [5]:
```
transactions = []
for i in range(0, data.shape[0]):
    transactions.append([str(data.values[i, j]) for j in range(0, 20)])

print(transactions[0])
```

```
['burgers', 'meatballs', 'eggs', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'na
n', 'nan', 'nan', 'nan']
```

3. Using Model

In [6]:
```
from apyori import apriori

rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_le ngth = 2)
# Support: number of transactions containing set of times / total number of transactions
# .      --> products that are bought at least 3 times a day --> 21 / 7501 = 0.0027
# Confidence: Should not be too high, as then this wil lead to obvious rules

#Try many combinations of values to experiment with the model.

#viewing the rules
results = list(rules)
```

4. Output

In [7]:
```
#Transferring the list to a table

results = pd.DataFrame(results)
results.head(5)
```

Out[7]:

	items	support	ordered_statistics
0	(chicken, light cream)	0.004533	[((light cream), (chicken), 0.290598290598290...
1	(mushroom cream sauce, escalope)	0.005733	[((mushroom cream sauce), (escalope), 0.300699...
2	(pasta, escalope)	0.005867	[((pasta), (escalope), 0.37288135593220345, 4....
3	(honey, fromage blanc)	0.003333	[((fromage blanc), (honey), 0.2450980392156863...
4	(herb & pepper, ground beef)	0.016000	[((herb & pepper), (ground beef), 0.3234501347...

In [9]:
```
results['ordered_statistics'][0]
```

Out[9]:
```
[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.2905982905982906, lift=
4.843304843304844)]
```