# Report

## Associate Rule Mining: Market Basket Analysis

**Name:** Keya Shukla

- **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.

  - Analyzed a store dataset to summarize their main characteristics using Pandas, NumPy and implemented graphs using Matplotlib and seaborn to identify the underlying relations between different items.
  - Executed an association rule mining, Apriori algorithm, to find the rules that describes the relation between each of the products that were bought by the customers.
  - Discovered a pattern between products and buying patterns of customers by finding out the Life, Support and Confidence scores of the applied algorithm and prepared a 3D graph to visualize the scores per product relation.

- **Prerequisites**

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)
  - Tools:
    - Pandas
    - Numpy
    - Matplotlib
    - Apyori
  - Dataset: Store data from Google drive

- **Method Used**

  Association rule mining is a technique to identify underlying relations between different items. Take an example of a Super Market where customers can buy variety of items. Usually, there is a pattern in what the customers buy. For instance, mothers with babies buy baby products such as milk and diapers. Damsels may buy makeup items whereas bachelors may buy beers and chips etc. In short, transactions involve a pattern. More profit can be generated if the relationship between the items purchased in different transactions can be identified.

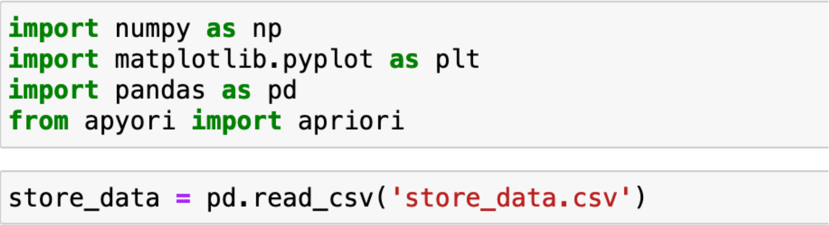
For instance, if item A and B are bought together more frequently then several steps can be taken to increase the profit. For example:

1. A and B can be placed together so that when a customer buys one of the products, he doesn't have to go far away to buy the other product.
2. People who buy one of the products can be targeted through an advertisement campaign to buy the other.
3. Collective discounts can be offered on these products if the customer buys both of them.
4. Both A and B can be packaged together.

The process of identifying an association between products is called association rule mining.

- **Implementation:**

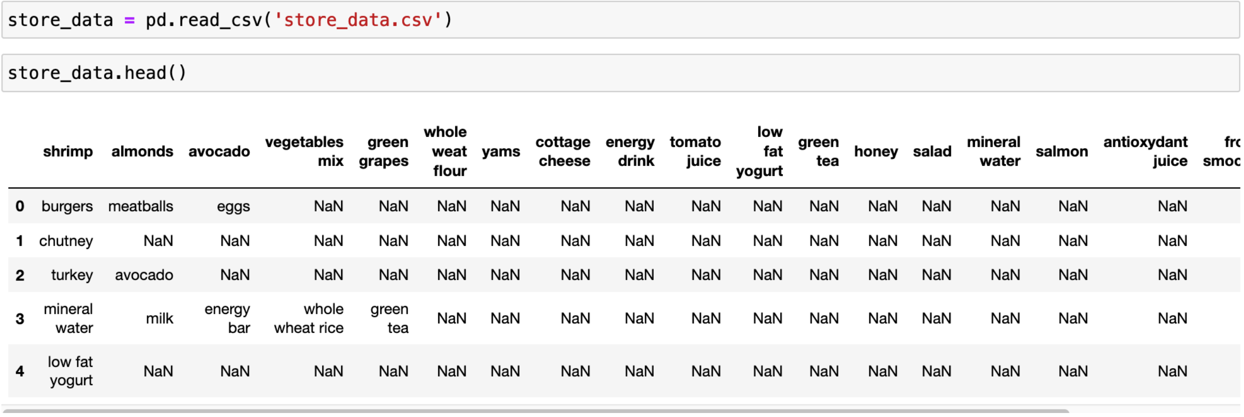
**1.** Load all required libraries



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

```
store_data = pd.read_csv('store_data.csv')
```

**2.** Calling dataset and checking the first five rows of the dataset

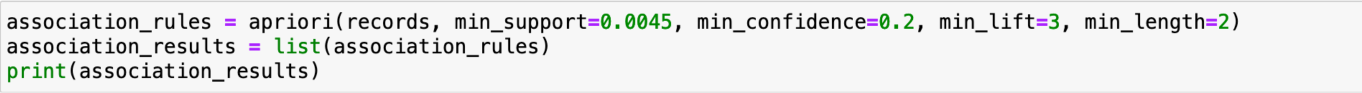


```
store_data = pd.read_csv('store_data.csv')
```

```
store_data.head()
```

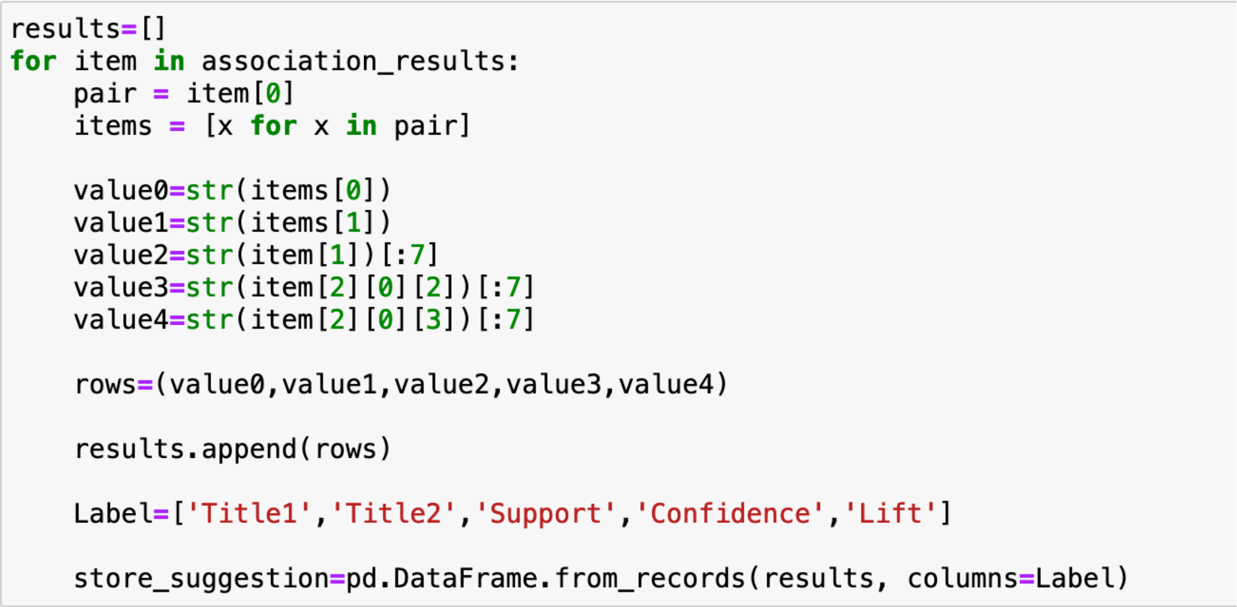
| | shrimp | almonds | avocado | vegetables mix | green grapes | whole weat flour | yams | cottage cheese | energy drink | tomato juice | low fat yogurt | green tea | honey | salad | mineral water | salmon | antioxydant juice | fro smoc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | burgers | meatballs | eggs | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1** | chutney | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **2** | turkey | avocado | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **3** | mineral water | milk | energy bar | whole wheat rice | green tea | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **4** | low fat yogurt | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

**3.** Implementing Apriori method



```
association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
association_results = list(association_rules)
print(association_results)
```

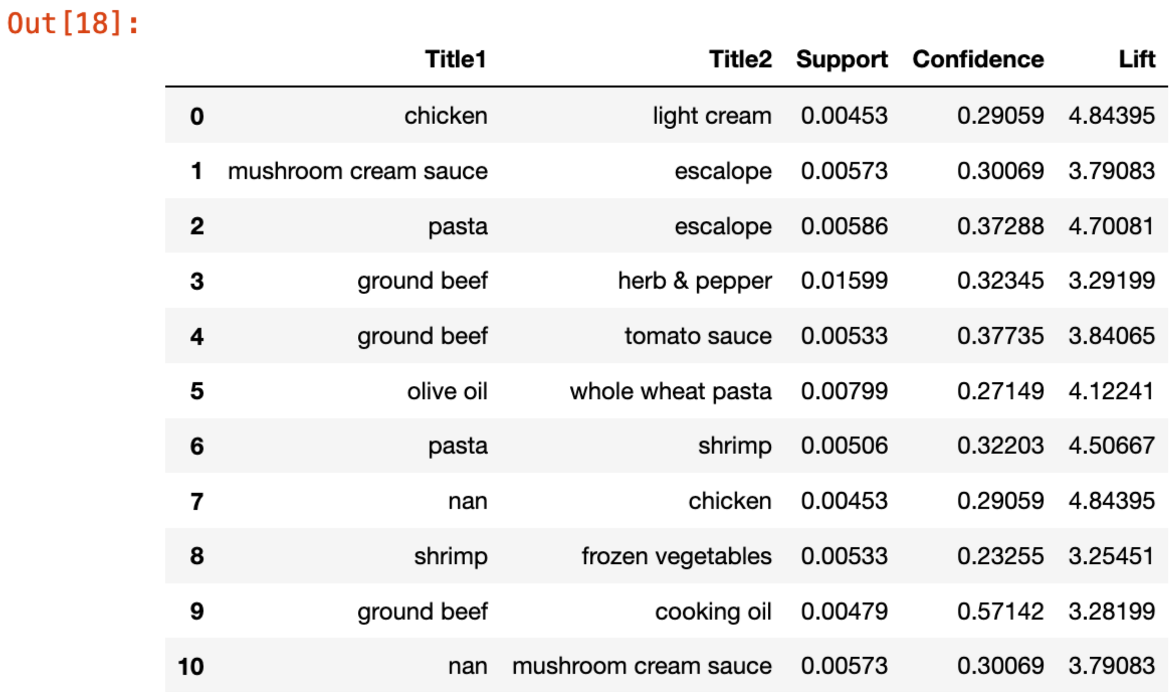
4. Calculating Support, Confidence and Lift of the result



```
results=[]
for item in association_results:
    pair = item[0]
    items = [x for x in pair]

    value0=str(items[0])
    value1=str(items[1])
    value2=str(item[1])[:7]
    value3=str(item[2][0][2])[:7]
    value4=str(item[2][0][3])[:7]

    rows=(value0,value1,value2,value3,value4)

    results.append(rows)

    Label=['Title1','Title2','Support','Confidence','Lift']

    store_suggestion=pd.DataFrame.from_records(results, columns=Label)
```

- **Results:**

1. Result of association mining



Out[18]:

| | Title1 | Title2 | Support | Confidence | Lift |
|---|---|---|---|---|---|
| **0** | chicken | light cream | 0.00453 | 0.29059 | 4.84395 |
| **1** | mushroom cream sauce | escalope | 0.00573 | 0.30069 | 3.79083 |
| **2** | pasta | escalope | 0.00586 | 0.37288 | 4.70081 |
| **3** | ground beef | herb & pepper | 0.01599 | 0.32345 | 3.29199 |
| **4** | ground beef | tomato sauce | 0.00533 | 0.37735 | 3.84065 |
| **5** | olive oil | whole wheat pasta | 0.00799 | 0.27149 | 4.12241 |
| **6** | pasta | shrimp | 0.00506 | 0.32203 | 4.50667 |
| **7** | nan | chicken | 0.00453 | 0.29059 | 4.84395 |
| **8** | shrimp | frozen vegetables | 0.00533 | 0.23255 | 3.25451 |
| **9** | ground beef | cooking oil | 0.00479 | 0.57142 | 3.28199 |
| **10** | nan | mushroom cream sauce | 0.00573 | 0.30069 | 3.79083 |

- **Support**

  Support refers to the default popularity of an item and can be calculated by finding number of transactions containing a particular item divided by total number of transactions.

- **Confidence**

  Confidence refers to the likelihood that an item B is also bought if item A is bought. It can be calculated by finding the number of transactions where A and B are bought together, divided by total number of transactions where A is bought.

- **Lift**

  Lift(A -> B) refers to the increase in the ratio of sale of B when A is sold. Lift(A –> B) can be calculated by dividing Confidence(A -> B) divided by Support(B).