Market Basket Analysis

INTRODUCTION:

Market basket analysis is a strategic data mining technique used by retailers to enhance sales by gaining a deeper understanding of customer purchasing patterns. This method entails the examination of substantial datasets, such as historical purchase records, in order to unveil inherent product groupings and identify items that tend to be bought together.

PROBLEM THINKING:

Nowadays people buy daily goods from super market nearby. There are many supermarkets that provide goods to their customer. The problem many retailers face is the placement of the items. They are unaware of the purchasing habits of the customer so they don't know which items should be placed together in their store. With the help of this application shop managers can determine the strong relationships between the items which ultimately helps them to put products that co-occur together close to one another. Also decisions like which item to stock more, cross selling, up selling, store shelf arrangement are determined

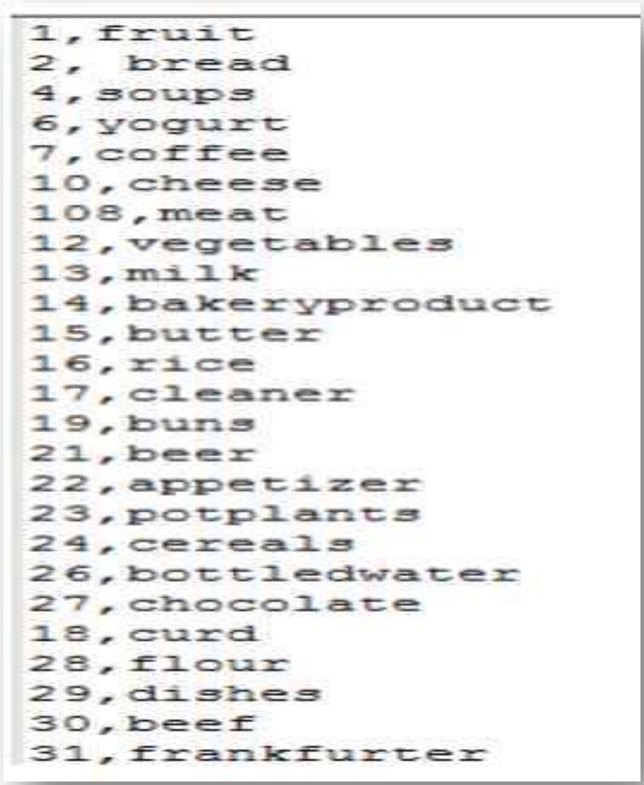
Market Basket Example

Where should detergents be placed in the Store to maximize their sales?
Are window cleaning products purchased when detergents and orange juice are bought together?
Is soda typically purchased with bananas? Does the brand of soda make a difference?
How are the demographics of the neighborhood affecting what customers are buying?
Window Cleaner
FLORIDA Orange Juice
Dish SOAP
Image source: deepclimate.org

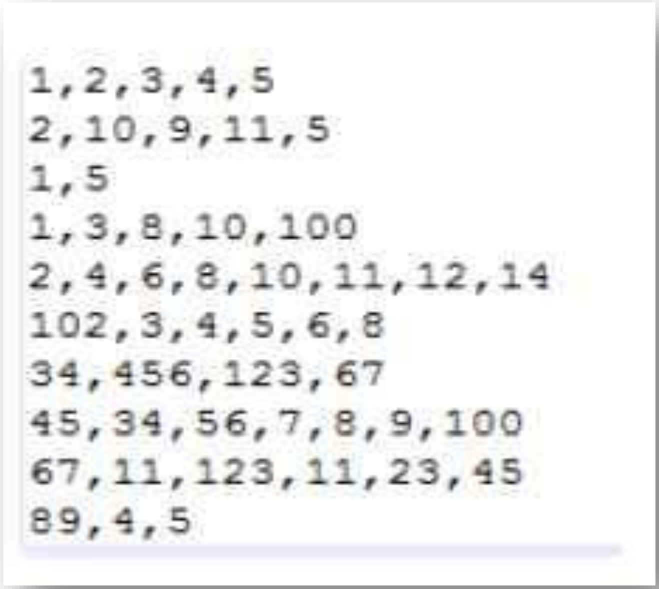
DESIGN THINKING:

DATA PREPROCESSING:

The data collected was mapped manually as integer values as shown in Figure . For example the "Fruit" was labeled as 1, "Bread" as 2 "Soups"as 4 and so on.


```
1,fruit
2,  bread
4,soups
6,yogurt
7,coffee
10,cheese
108,meat
12,vegetables
13,milk
14,bakeryproduct
15,butter
16,rice
17,cleaner
19,buns
21,beer
22,appetizer
23,potplants
24,cereals
26,bottledwater
27,chocolate
18,curd
28,flour
29,dishes
30,beef
31,frankfurter
```

The mapped integer's values were then saved in a text file and given as the input to the system.


```
1,2,3,4,5
2,10,9,11,5
1,5
1,3,8,10,100
2,4,6,8,10,11,12,14
102,3,4,5,6,8
34,456,123,67
45,34,56,7,8,9,100
67,11,123,11,23,45
89,4,5
```

APRIORI ALGORITHM:

Association rule mining finds interesting associations and/or correlation relationships among large set of data items. Association rules shows attribute value conditions that occur frequently together in a given dataset. A typical and widely used example of association rule mining is Market Basket Analysis. For example, data are collected from the supermarkets. Such market basket databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Association rules provide information of this type in the form of "IF-THEN" statements. The rules are computed from the data, an association rule has two numbers that express the degree of uncertainty about the rule.

a. Support

b. Confidence

SUPPORT

The support of an item is the number of transaction containing the item. Those items that do not meet the minimum support are excluded from the further processing. Support determines how often a rule is applicable to a given data set. Support $(X \cup Y) = \min(\text{Support}(X), \text{Support}(Y))$

CONFIDENCE

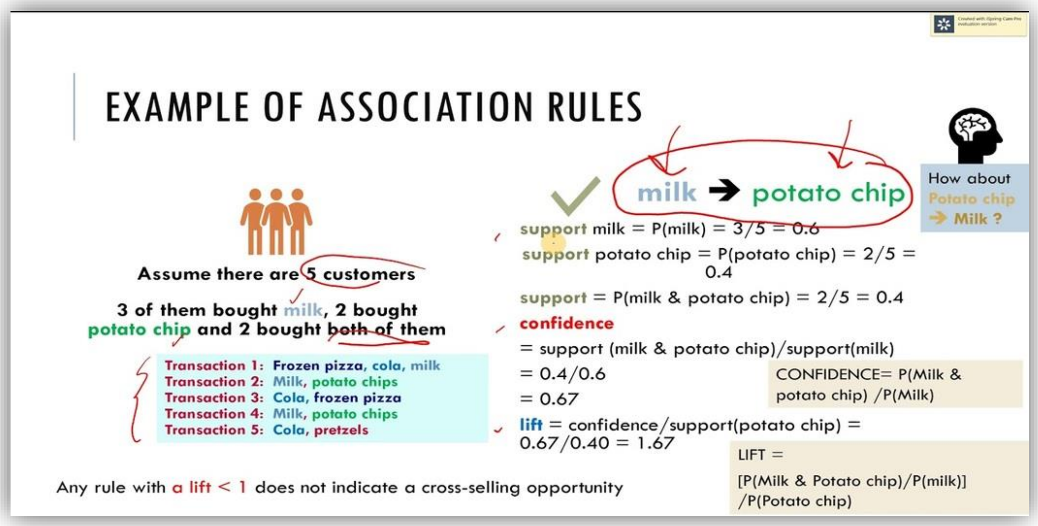
Confidence is defined as the conditional probability that a transaction containing the LHS will also contain the RHS.

Confidence (LHS->RHS-> P(RHS/LHS)=P(RHS∩LHS)/P(LHS)=support(RHS∩LHS)/support(LHS).

Confidence determines how frequently item in RHS appears in the transaction that Contain LHS. While determining the rules we must measure these two components as it is very important to us. A rule that has very low support may occur simply by chance.

PSEUDOCODE

```
//Find all frequent itemset
Apriori(database D of transaction, min_support){
F1={frequent 1-itemset}
K=2
While Fk-1≠ Empty Set
Ck=AprioriGeneration (Fk-1)//Generate candidate item sets.
For each transaction in the database D {
Ct=subset (Ck, t)
For each candidate c in C
t{
Count c++
 }
 Fk={c in Ck such that countc>min_support}
 K++
}
 F=U K>Fk
 }
//prune the candidate item sets
 Apriori generation (Fk-1) {
//Insert into Ck all combination of elements in Fk-1 obtained by self-joining item
 sets in Fk-1
//Delete all item sets c in Ck such that some (K-1) subset of c is not in Lk-1
}
//find all subsets of candidate contained in t
 Subset (Ck, t)
}
```



EXAMPLE OF ASSOCIATION RULES

Assume there are 5 customers

3 of them bought milk, 2 bought potato chip and 2 bought both of them

Transaction 1: Frozen pizza, cola, milk
Transaction 2: Milk, potato chips
Transaction 3: Cola, frozen pizza
Transaction 4: Milk, potato chips
Transaction 5: Cola, pretzels

milk ➔ potato chip

How about Potato chip ➔ Milk ?

support milk = P(milk) = 3/5 = 0.6

support potato chip = P(potato chip) = 2/5 = 0.4

support = P(milk & potato chip) = 2/5 = 0.4

confidence
= support (milk & potato chip)/support(milk)
= 0.4/0.6
= 0.67

CONFIDENCE= P(Milk & potato chip) /P(Milk)

lift = confidence/support(potato chip) = 0.67/0.40 = 1.67

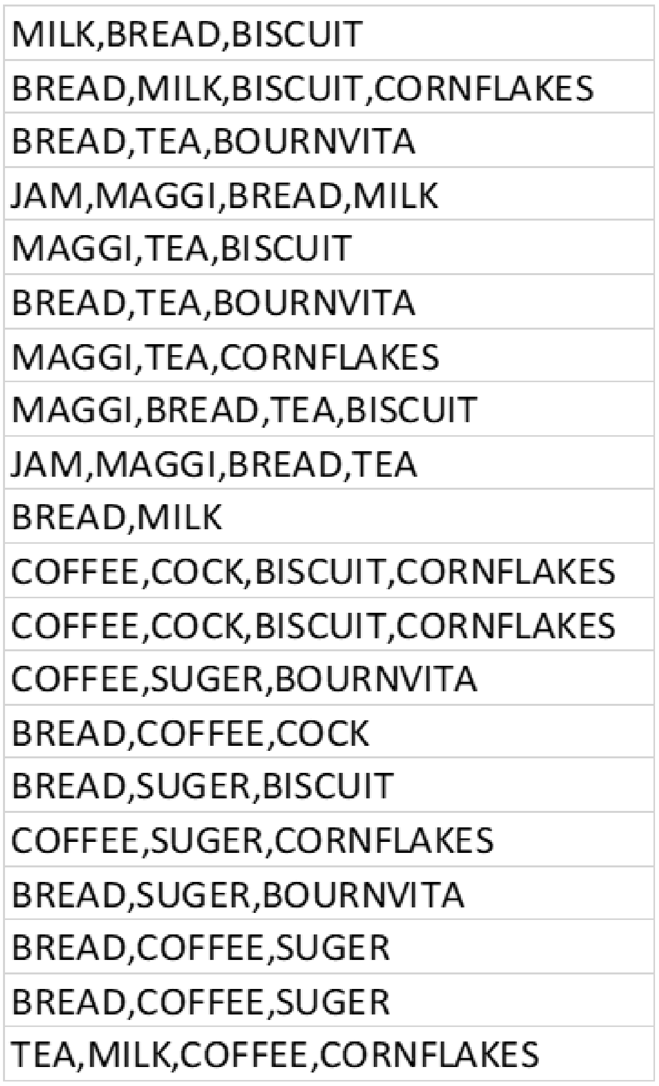
LIFT = [P(Milk & Potato chip)/P(milk)] /P(Potato chip)

Any rule with a lift < 1 does not indicate a cross-selling opportunity

PHASES OF DEVELOPMENT:

This dataset contains 11 items: JAM, MAGGI, SUGAR, COFFEE, CHEESE, TEA, BOURNVITA, CORNFLAKES, BREAD, BISCUIT, and MILK.

There are 19 transactions and 1 row represents 1 transaction



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

STEPS SUMMARY

1. Import libraries and load data
2. One-hot encoding
3. Find the frequent itemsets using Apriori
4. Association rule

1. IMPORT LIBRARIES

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rulesdf =
pd.read_csv("GroceryStoreDataSet.csv", names = ['transaction'], sep = ',')
```

- pandas for dataframe management
- mlxtend (machine learning extensions) is a Python library of useful tools for day-to-day data science tasks.
- mlxtend.preprocessing for data preprocessing before performing the task.
- mlxtend.frequent_patterns for association rule task.
- load data as a variable df

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

2. ONE-HOT ENCODING

*We have to change the data to the appropriate format before inputting it into the Apriori algorithm. The goal is a **one-hot DataFrame**.*

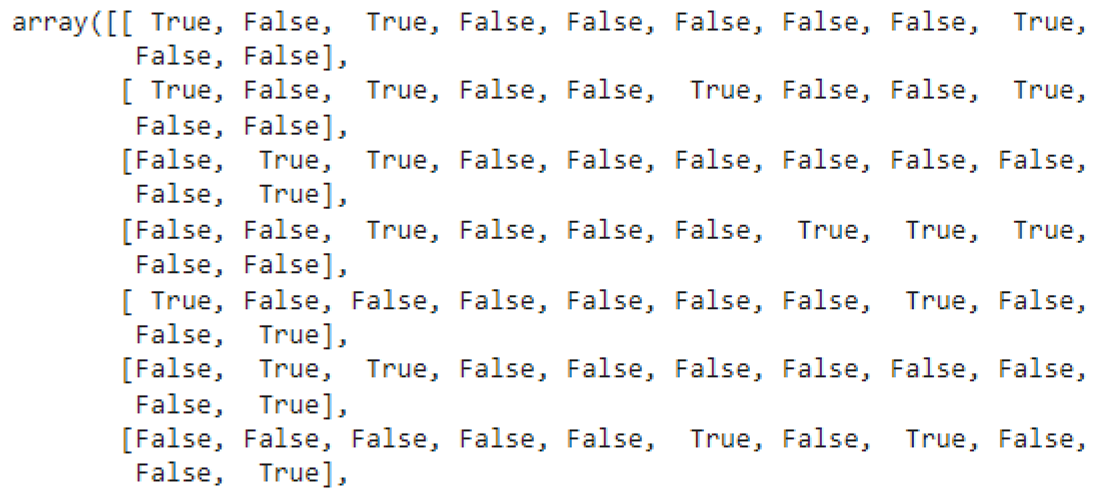
- *Change the DataFrame to a list of lists.*

```
df = list(df["transaction"].apply(lambda x:x.split(",")))
```

```
[['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']]
```

- *Use TransactionEncoder from the mlxtend library to change the list of transactions to a one-hot array.*

```
one_hot_transformer = TransactionEncoder()df_transform =
one_hot_transformer.fit_transform(df)
```

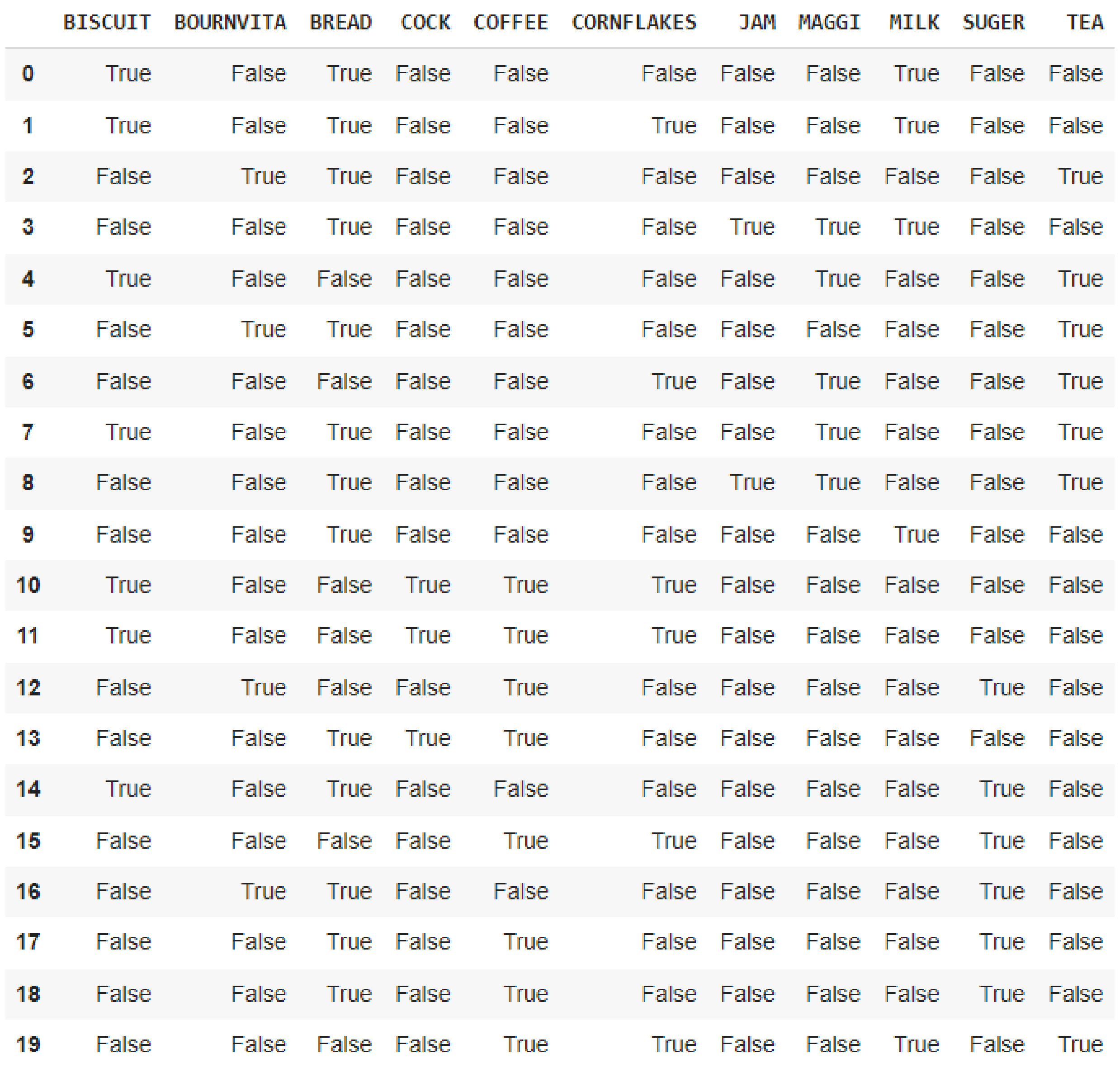


```
array([[ True, False,  True, False, False, False, False, False,  True,
         False, False],
       [ True, False,  True, False, False,  True, False, False,  True,
         False, False],
       [False,  True,  True, False, False, False, False, False, False,
         False,  True],
       [False, 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],
```

It is still hard to read, so we will change it to a dataframe with column names.

- *Change the one-hot array to a DataFrame.*

```
df = pd.DataFrame(df_transform,columns=one_hot_transformer.columns_)
```



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

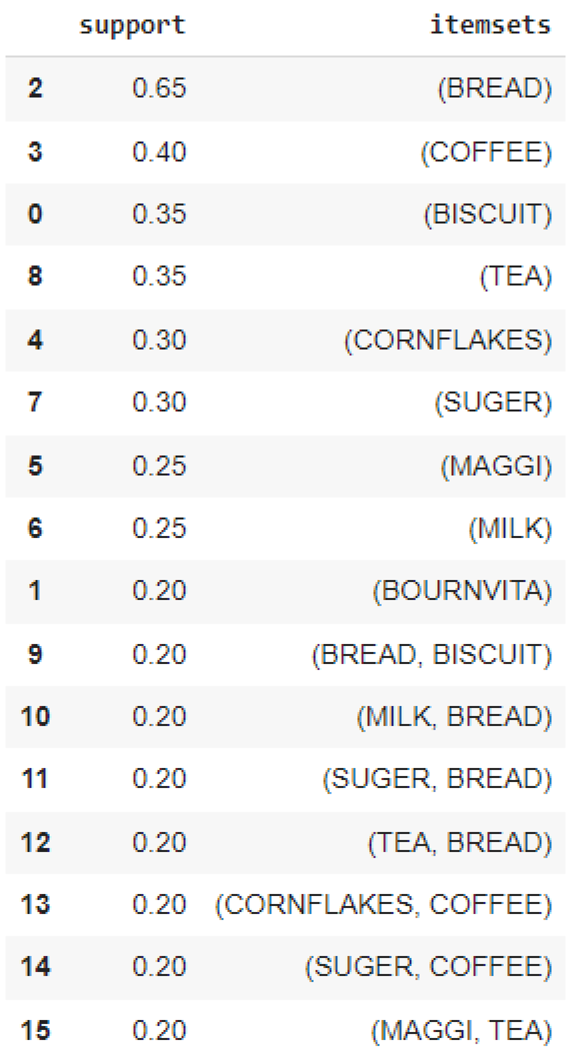
This is our desired format, one-hot. The columns are items in the store and each row represents a transaction. If the value is True, that item is sold in that transaction. Now, the data is ready to be fed to the algorithm.

3. FIND THE FREQUENT ITEMSETS USING APRIORI

We will use Apriori to find the frequent itemsets from the one-hot transaction DataFrame. This step's objective is to decrease the computational workload in the association rule.

Frequent itemsets' supports are higher than minimum support. You can adjust min_support to suit your dataset.

```
df = apriori(df, min_support = 0.2, use_colnames = True)
df.sort_values(['support'],ascending=False, inplace = True)
```



	support	itemsets
2	0.65	(BREAD)
3	0.40	(COFFEE)
0	0.35	(BISCUIT)
8	0.35	(TEA)
4	0.30	(CORNFLAKES)
7	0.30	(SUGER)
5	0.25	(MAGGI)
6	0.25	(MILK)
1	0.20	(BOURNVITA)
9	0.20	(BREAD, BISCUIT)
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 this example, I set the minimum support to 0.2, so it will filter only a set of item(s) whose support is greater than 0.2 (or more than 20% from 19 transactions).

Only these itemsets which are considered important will proceed to the association rule.

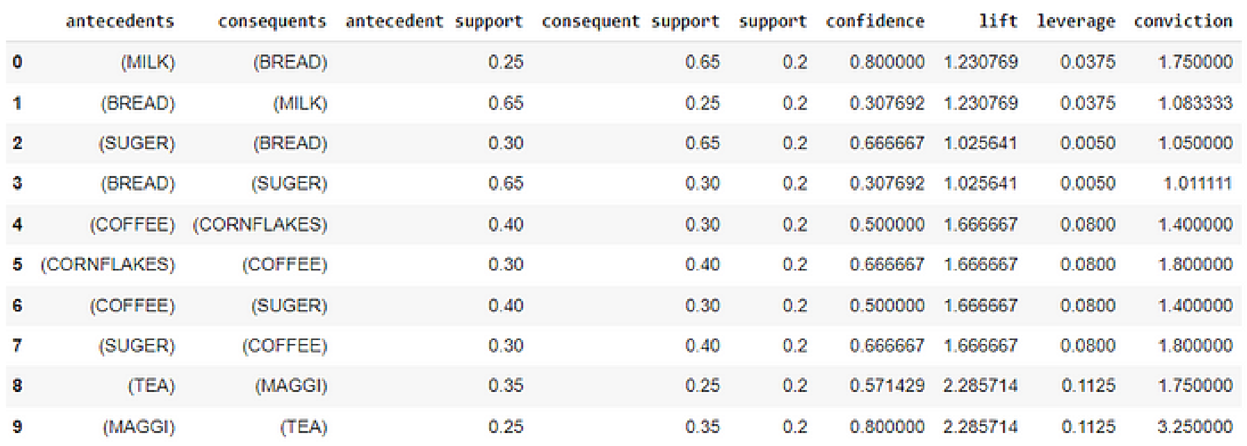
4. ASSOCIATION RULE

It is time for the main step!!

The association_rules function will automatically calculate key metrics of our transaction data including support, confidence, lift, leverage, and conviction.

```
df_ar = association_rules(df, metric="lift", min_threshold=1)
```

The result is shown below.



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.800000	1.230769	0.0375	1.750000
1	(BREAD)	(MILK)	0.65	0.25	0.2	0.307692	1.230769	0.0375	1.083333
2	(SUGER)	(BREAD)	0.30	0.65	0.2	0.666667	1.025641	0.0050	1.050000
3	(BREAD)	(SUGER)	0.65	0.30	0.2	0.307692	1.025641	0.0050	1.011111
4	(COFFEE)	(CORNFLAKES)	0.40	0.30	0.2	0.500000	1.666667	0.0800	1.400000
5	(CORNFLAKES)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.800000
6	(COFFEE)	(SUGER)	0.40	0.30	0.2	0.500000	1.666667	0.0800	1.400000
7	(SUGER)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.800000
8	(TEA)	(MAGGI)	0.35	0.25	0.2	0.571429	2.285714	0.1125	1.750000
9	(MAGGI)	(TEA)	0.25	0.35	0.2	0.800000	2.285714	0.1125	3.250000

The result is required interpretation for further business actions.

INTERPRETATION EXAMPLES

- *From the highest confidence in index 0, the confidence is 0.8. It means that the customer who buys milk will buy bread for 80%. However, you have to keep in mind that confidence is not everything. The high confidence in this row is due to the high support of bread (consequence support 0.65) which means bread occurs in many transactions, so it will not make a business impact if we try to sell bread with milk anyway.*
- *The better metric is lift or leverage. Index 8 has the highest lift and leverage at 2.28 and 0.1125 respectively. It means that the customers who buy tea are likely to buy Maggi as well.*

DATASET:

For my Data Mining lab where we had to execute algorithms like apriori, it was very difficult to get a small data set with only a few transactions. It was infeasible to run the algorithm with datasets containing over 10000 transactions. This dataset contains 11 items : JAM, MAGGI, SUGAR, COFFEE, CHEESE, TEA, BOURNVITA, CORNFLAKES, BREAD, BISCUIT and MILK.

FEATURE EXTRACTION

Product level features

Aisle and department features

User level features

User and Product level features

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import gc

import category_encoders as ce

root = 'C:/Data/instacart-market-basket-analysis/'
aisles = pd.read_csv(root + 'aisles.csv')

departments = pd.read_csv(root + 'departments.csv')

orders = pd.read_csv(root + 'orders.csv',
            dtype={
                'order_id': np.int32,
                'user_id': np.int64,
                'eval_set': 'category',
                'order_number': np.int16,
                'order_dow': np.int8,
                'order_hour_of_day': np.int8,
                'days_since_prior_order': np.float32})

order_products_prior = pd.read_csv(root + 'order_products__prior.csv',
                    dtype={
                        'order_id': np.int32,
                        'product_id': np.uint16,
                        'add_to_cart_order': np.int16,
                        'reordered': np.int8})
```

```
order_products_train = pd.read_csv(root + 'order_products__train.csv',
                        dtype={
                            'order_id': np.int32,
                            'product_id': np.uint16,
                            'add_to_cart_order': np.int16,
                            'reordered': np.int8})
products = pd.read_csv(root + 'products.csv')
```

MACHINE LEARNING ALGORITHM:

Machine Learning is helping the Retail Industry in many different ways. You can imagine that from forecasting the performance of sales to identifying the buyers, there are many applications of AI and ML in the retail industry. Market basket analysis is a data mining technique retailers use to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings and products likely to be purchased together. In this article, we will comprehensively cover the topic of Market Basket Analysis and its various components and then dive deep into the ways of implementing it in machine learning, including how to perform it in Python on a real-world dataset.

METRICS FOR EVALUATING ASSOCIATION RULES

There are a few different interestingness metrics you may apply to your association rules:

- ***Support***: *This is the easiest metric to calculate, as it's simply the proportion of all your transactions that contain an association rule.*
  - *In our dataset above, we find support for {green beans}* ➡ *{french fried onions} is 0.5 (2 transactions out of 4). Higher numbers closer to 1 are better here.*
  - *Support is easy to calculate, but imagine trying to do this for more popular items in the store. How many people buy {bread, eggs} when they shop? Probably a lot. You may get a high support metric for that association rule, but it won't add much nuance to your understanding of your customers' habits.*
  - ***Confidence***: *Confidence brings a bit more specificity to your judgment of this association rule. In this case, it's the proportion of all the transactions that contain all the items in the itemset over the proportion of transactions containing just one of them. (Yes, this is the same as dividing the support metric for {green beans}* ➡ *{french fried onions} by the support metric for just {green beans} alone.)*
  - *In our dataset above, 2 of 4 transactions included both items, and 3 of 4 included green beans. That's 0.5 / 0.75, or 0.67. Again, higher numbers closer to 1 are better here.*

- Confidence gives us the probability that a customer will purchase the consequent, the item on the right of our association rule — the french fried onions — given that they purchased green beans, our antecedent. As you can see, this metric provides a different and perhaps more useful insight into the nature of customers' behavior; we are getting not just frequency, but also a measure of likelihood.

- **Lift**: Some people will buy green beans. Some will buy french fried onions. Some will buy both. If we imagine there's no relationship between the two items, then we can see by how much we actually exceed that expectation when people do buy both. That calculation is called lift.

For our mini dataset, this comes out to 0.5 / (0.75 * 0.5) or 1.33. Here's how you can assess lift:

- If lift is greater than 1, the antecedent is in fact increasing the likelihood of the consequent also appearing in a transaction (yes to green beans, more likely a yes to french fried onions, which is our case here).
- If lift is below 1, then it's the opposite; the antecedent decreases the likelihood of the consequent (yes to green beans, more likely a no to french fried onions). This might be the case with products filling the same need; for example, if I buy a bottle of my usual brand of shampoo during a shopping trip, odds are I won't buy a bottle of another brand, too.

- *If lift equals 1, then the antecedent isn't affecting the chance of buying the consequent.*

CONCLUSION:

The Apriori algorithm effectively generates highly informative frequent itemsets and association rules for the data of the supermarket. The frequent data items are generated from the given input data and based on the frequent item stets strong association rules were generated.