

PA 3: Association Analysis - Apriori Algorithm

Apriori Algorithm:

The Apriori algorithm is used for mining frequent itemsets and devising association rules from a transactional dataset. Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given dataset. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. The parameters used in this algorithm are support and confidence

Support: Item's frequency of occurrence

Support is calculated as follows:

$$\text{Support}(i(1^{\text{st}} \text{ item})) = \text{Item's frequency} / \text{total number of transactions}$$

Confidence: It is a conditional probability

Confidence is calculated as follows:

$$\text{Confidence} = \text{support of itemset} / \text{support}(i(1^{\text{st}} \text{ item}))$$

Task1: Data Pre-processing

We read the given dataset dataset_group.csv into data frame and we give the headers 'Date', 'id', 'tran' for the dataset

```
Out[125]:
```

	Date	id	tran
0	2000-01-01	1	yogurt
1	2000-01-01	1	pork
2	2000-01-01	1	sandwich bags
3	2000-01-01	1	lunch meat
4	2000-01-01	1	all- purpose

Then we drop the date column as we no need this in apriori algorithm

```
Out[126]:
```

	id	tran
0	1	yogurt
1	1	pork
2	1	sandwich bags
3	1	lunch meat
4	1	all- purpose

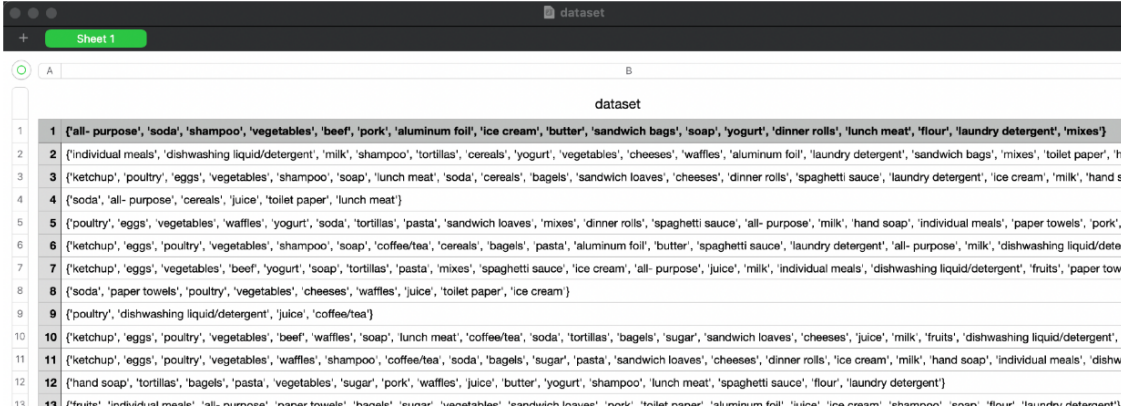
- Since the algorithm takes set of values we apply a set for the values in the dataset using groupby of ID and Tran columns
- We extract 5% of the given dataset and update the dataset with the 75 rows to dataset.csv

Out[8]:

	ID	Transaction
0	1	{'aluminum foil', 'lunch meat', 'pork', 'all- purpose', 'dinner rolls', 'beef', 'mixes', 'soda', 'soap', 'vegetables', 'butter', 'flour', 'yogurt', 'sandwich bags', 'laundry detergent', 'shampoo', 'ice cream'}
1	2	{'milk', 'aluminum foil', 'yogurt', 'cereals', 'tortillas', 'dishwashing liquid/detergent', 'toilet paper', 'mixes', 'cheeses', 'hand soap', 'vegetables', 'individual meals', 'sandwich bags', 'laundry detergent', 'shampoo', 'waffles'}
2	3	{'spaghetti sauce', 'sandwich loaves', 'vegetables', 'soda', 'milk', 'bagels', 'soap', 'cheeses', 'ketchup', 'lunch meat', 'cereals', 'hand soap', 'eggs', 'poultry', 'pork', 'dinner rolls', 'toilet paper', 'laundry detergent', 'shampoo', 'ice cream'}
3	4	{'lunch meat', 'all- purpose', 'cereals', 'toilet paper', 'juice', 'soda'}
4	5	{'spaghetti sauce', 'all- purpose', 'sandwich loaves', 'vegetables', 'pasta', 'flour', 'soda', 'yogurt', 'milk', 'paper towels', 'tortillas', 'hand soap', 'waffles', 'eggs', 'mixes', 'poultry', 'pork', 'dinner rolls', 'toilet paper', 'individual meals'}
5	6	{'spaghetti sauce', 'all- purpose', 'vegetables', 'pasta', 'aluminum foil', 'paper towels', 'milk', 'bagels', 'soap', 'coffee/tea', 'cereals', 'butter', 'eggs', 'poultry', 'toilet paper', 'laundry detergent', 'shampoo', 'dishwashing liquid/detergent'}
6	7	{'spaghetti sauce', 'all- purpose', 'vegetables', 'juice', 'pasta', 'yogurt', 'milk', 'paper towels', 'tortillas', 'soap', 'sandwich bags', 'ketchup', 'fruits', 'beef', 'dishwashing liquid/detergent', 'individual meals', 'eggs', 'mixes', 'toilet paper', 'ice cream'}

- The screenshot of the dataset.csv is given below

Out[9]:



	A	B
		dataset
1	1	{'all- purpose', 'soda', 'shampoo', 'vegetables', 'beef', 'pork', 'aluminum foil', 'ice cream', 'butter', 'sandwich bags', 'soap', 'yogurt', 'dinner rolls', 'lunch meat', 'flour', 'laundry detergent', 'mixes'}
2	2	{'individual meals', 'dishwashing liquid/detergent', 'milk', 'shampoo', 'tortillas', 'cereals', 'yogurt', 'vegetables', 'cheeses', 'waffles', 'aluminum foil', 'laundry detergent', 'sandwich bags', 'mixes', 'toilet paper', 'h
3	3	{'ketchup', 'poultry', 'eggs', 'vegetables', 'shampoo', 'soap', 'lunch meat', 'soda', 'cereals', 'bagels', 'sandwich loaves', 'cheeses', 'dinner rolls', 'spaghetti sauce', 'laundry detergent', 'ice cream', 'milk', 'hand s
4	4	{'soda', 'all- purpose', 'cereals', 'juice', 'toilet paper', 'lunch meat'}
5	5	{'poultry', 'eggs', 'vegetables', 'waffles', 'yogurt', 'soda', 'tortillas', 'pasta', 'sandwich loaves', 'mixes', 'dinner rolls', 'spaghetti sauce', 'all- purpose', 'milk', 'hand soap', 'individual meals', 'paper towels', 'pork,
6	6	{'ketchup', 'eggs', 'poultry', 'vegetables', 'shampoo', 'soap', 'coffee/tea', 'cereals', 'bagels', 'pasta', 'aluminum foil', 'butter', 'spaghetti sauce', 'laundry detergent', 'all- purpose', 'milk', 'dishwashing liquid/deter
7	7	{'ketchup', 'eggs', 'vegetables', 'beef', 'yogurt', 'soap', 'tortillas', 'pasta', 'mixes', 'spaghetti sauce', 'ice cream', 'all- purpose', 'juice', 'milk', 'individual meals', 'dishwashing liquid/detergent', 'fruits', 'paper tow
8	8	{'soda', 'paper towels', 'poultry', 'vegetables', 'cheeses', 'waffles', 'juice', 'toilet paper', 'ice cream'}
9	9	{'poultry', 'dishwashing liquid/detergent', 'juice', 'coffee/tea'}
10	10	{'ketchup', 'eggs', 'poultry', 'vegetables', 'beef', 'waffles', 'soap', 'lunch meat', 'coffee/tea', 'soda', 'tortillas', 'bagels', 'sugar', 'sandwich loaves', 'cheeses', 'juice', 'milk', 'fruits', 'dishwashing liquid/detergent', 'h
11	11	{'ketchup', 'eggs', 'poultry', 'vegetables', 'waffles', 'shampoo', 'coffee/tea', 'soda', 'bagels', 'sugar', 'pasta', 'sandwich loaves', 'cheeses', 'dinner rolls', 'ice cream', 'milk', 'hand soap', 'individual meals', 'dishw
12	12	{'hand soap', 'tortillas', 'bagels', 'pasta', 'vegetables', 'sugar', 'pork', 'waffles', 'juice', 'butter', 'yogurt', 'shampoo', 'lunch meat', 'spaghetti sauce', 'flour', 'laundry detergent'}
13	13	{'fruits', 'individual meals', 'all- purpose', 'paper towels', 'bagels', 'sugar', 'vegetables', 'sandwich loaves', 'pork', 'toilet paper', 'aluminum foil', 'juice', 'ice cream', 'shampoo', 'soap', 'flour', 'laundry detergent'}

Task2: Run apriori.py and Evaluate Results

Case1:

For case1 to evaluate results we use minimum support = 0.1 & minimum confidence = 0.2

Reasoning:

According to Apriori algorithm, if we keep the support count to minimum, the more rules it produces. As we can see when we put the minimum support as 0.1 which is 10% and minimum confidence to 20% we get the huge number of rules and vice versa.

Case 1 Output:

```
-----ITEMS-----
item: (' \'waffles\'',) , 0.200
item: (" 'hand soap'", " 'poultry'") , 0.200
item: (" 'juice'", " 'fruits'") , 0.200
item: (" 'cheeses'", " 'fruits'") , 0.200
item: (" 'lunch meat'", " 'fruits'") , 0.200
item: (" 'vegetables'", ' \'ice cream\'') , 0.200
item: (" 'juice'", " 'cereals'") , 0.200
item: (" 'sugar'", " 'paper towels'") , 0.200
item: (" 'lunch meat'", " 'poultry'") , 0.200
item: (" 'toilet paper'", " 'juice'") , 0.200
item: (" 'cheeses'", " 'mixes'") , 0.200
item: (" 'poultry'", " 'dinner rolls'") , 0.200
item: (" 'cereals'", " 'mixes'") , 0.200
item: (" 'shampoo'", " 'lunch meat'") , 0.200
item: ('\'{\'spaghetti sauce\'', " 'hand soap'") , 0.200
item: (" 'hand soap'", " 'lunch meat'") , 0.200
```

```
-----RULES-----
Rule: (" 'vegetables'",) ==> (" 'ketchup'",) , 0.315
Rule: (" 'vegetables'",) ==> (" 'shampoo'", " 'laundry detergent'") , 0.315
Rule: (" 'vegetables'",) ==> (" 'poultry'", " 'soda'") , 0.315
Rule: (" 'vegetables'",) ==> (" 'sandwich bags'",) , 0.333
Rule: (" 'vegetables'",) ==> (" 'aluminum foil'",) , 0.333
Rule: (" 'vegetables'",) ==> ('\'{\'spaghetti sauce\'',) , 0.352
Rule: (" 'vegetables'",) ==> (" 'paper towels'",) , 0.352
Rule: (" 'vegetables'",) ==> (" 'toilet paper'",) , 0.352
Rule: (" 'vegetables'",) ==> (" 'all- purpose'",) , 0.352
Rule: (" 'vegetables'",) ==> (" 'soap'",) , 0.370
Rule: (" 'vegetables'",) ==> (" 'tortillas'",) , 0.370
Rule: (" 'vegetables'",) ==> (" 'pork'",) , 0.370
Rule: (" 'vegetables'",) ==> (" 'eggs'",) , 0.370
Rule: (" 'vegetables'",) ==> (" 'coffee/tea'",) , 0.389
Rule: (" 'vegetables'",) ==> (" 'bagels'",) , 0.389
Rule: (" 'vegetables'",) ==> (" 'dinner rolls'",) , 0.389
Rule: (" 'vegetables'",) ==> (" 'mixes'",) , 0.389
Rule: (" 'vegetables'",) ==> (" 'beef'",) , 0.389
```

Case2:

For Case2 to evaluate results we use minimum support = 0.2 & minimum confidence = 0.4

Reasoning

In case 2, we tried to increase the minimum support to 0.2 and minimum confidence to 0.4, thereby decreasing the rules when compared to case 1. Moreover, it would be very difficult to analyse the data and background calculations for larger datasets, so we kept the reasoning to understanding the working functionality of algorithm.

item: (" 'eggs'",) , 0.343
item: (" 'sandwich loaves'",) , 0.347
item: (" 'pork'",) , 0.355
item: (" 'spaghetti sauce'",) , 0.356
item: (" 'poultry'",) , 0.358
item: (" 'juice'",) , 0.362
item: (" 'butter'",) , 0.364
item: (" 'ketchup'",) , 0.365
item: (" 'toilet paper'",) , 0.366
item: (" 'beef'",) , 0.369
item: (" 'all- purpose'",) , 0.370
item: (" 'mixes'",) , 0.371
item: (" 'dinner rolls'",) , 0.377
item: (" 'cereals'",) , 0.378
item: (" 'milk'",) , 0.378
item: (" 'laundry detergent'",) , 0.378
item: (" 'dishwashing liquid/detergent'",) , 0.381
item: (" 'cheeses'",) , 0.384
item: (" 'yogurt'",) , 0.384
item: (" 'soda'",) , 0.387
item: (" 'ice cream'",) , 0.392
item: (" 'vegetables'",) , 0.702

Rule: (" 'ketchup'",) ==> (" 'vegetables'",) , 0.752
Rule: (" 'soda'",) ==> (" 'vegetables'",) , 0.760
Rule: (" 'dishwashing liquid/detergent'",) ==> (" 'vegetables'",) , 0.760
Rule: (" 'laundry detergent'",) ==> (" 'vegetables'",) , 0.761
Rule: (" 'dinner rolls'",) ==> (" 'vegetables'",) , 0.762
Rule: (" 'cereals'",) ==> (" 'vegetables'",) , 0.763
Rule: (" 'aluminum foil'",) ==> (" 'vegetables'",) , 0.765
Rule: (" 'cheeses'",) ==> (" 'vegetables'",) , 0.767
Rule: (" 'beef'",) ==> (" 'vegetables'",) , 0.767
Rule: (" 'shampoo'",) ==> (" 'vegetables'",) , 0.770
Rule: (" 'spaghetti sauce'",) ==> (" 'vegetables'",) , 0.778
Rule: (" 'hand soap'",) ==> (" 'vegetables'",) , 0.778
Rule: (" 'sandwich loaves'",) ==> (" 'vegetables'",) , 0.780
Rule: (" 'soap'",) ==> (" 'vegetables'",) , 0.780
Rule: (" 'poultry'",) ==> (" 'vegetables'",) , 0.784
Rule: (" 'individual meals'",) ==> (" 'vegetables'",) , 0.787
Rule: (" 'sandwich bags'",) ==> (" 'vegetables'",) , 0.788
Rule: (" 'paper towels'",) ==> (" 'vegetables'",) , 0.795
Rule: (" 'yogurt'",) ==> (" 'vegetables'",) , 0.801
Rule: (" 'sugar'",) ==> (" 'vegetables'",) , 0.804
Rule: (" 'eggs'",) ==> (" 'vegetables'",) , 0.808

Case3:

For Case2 to evaluate results we use minimum support = 0.35 & minimum confidence = 0.45

Reasoning:

After observing the support counts of each item in the given dataset we concluded that all the items are in range 0 to 0.35 support count and if the min-support exceeds 35% then none of the items will be considered as frequent set item and no rules will generated after on. so, to skip this case we have considered the support values which are less than 35 as minimum support. According to algorithm we can conclude that higher the support count will give no rules.

Case 3 Output:

```
-----ITEMS-----
item: (" 'pork'",) , 0.355
item: (" 'spaghetti sauce'",) , 0.356
item: (" 'poultry'",) , 0.358
item: (" 'juice'",) , 0.362
item: (" 'butter'",) , 0.364
item: (" 'ketchup'",) , 0.365
item: (" 'toilet paper'",) , 0.366
item: (" 'beef'",) , 0.369
item: (" 'all- purpose'",) , 0.370
item: (" 'mixes'",) , 0.371
item: (" 'dinner rolls'",) , 0.377
item: (" 'cereals'",) , 0.378
item: (" 'milk'",) , 0.378
item: (" 'laundry detergent'",) , 0.378
item: (" 'dishwashing liquid/detergent'",) , 0.381
item: (" 'cheeses'",) , 0.384
item: (" 'yogurt'",) , 0.384
item: (" 'soda'",) , 0.387
item: (" 'ice cream'",) , 0.392
item: (" 'vegetables'",) , 0.702

-----RULES-----
```

AS we can conclude here that if we use support as 0.35 which is 35% of the data then there are no frequent items In the dataset hence no rules are generated.