Name: Kaushik Kotian Class: D15B Roll No. 30 Batch:B

Business Intelligence Lab

Experiment 9

Aim: To implement Association Mining Algorithm (Apriori) using Rapid Miner and Python.

Theory:

Dataset: It is a basket analysis dataset from Kaggle. It contains around 1000 rows and 17 attributes.

What is Association Rule mining?

Association Rule Mining, as the name suggests, involves discovering relationships between seemingly independent relational databases or other data repositories through simple If/Then statements.

While many machine learning algorithms operate with numeric datasets, association rule mining is tailored for non-numeric, categorical data. It involves more than simple counting but is relatively straightforward compared to complex mathematical models.

what is association rule?

The Association rule is a learning technique that helps identify the dependencies between two data items. Based on the dependency, it then maps accordingly so that it can be more profitable. Association rule furthermore looks for interesting associations among the variables of the dataset. It is undoubtedly one of the most important concepts of Machine Learning and has been used in different cases such as association in data mining and continuous production, among others. However, like all other techniques, association in data mining, too, has its own set of disadvantages. The same has been discussed in brief in this article.

An association rule has 2 parts:

- **an antecedent (if) and**
- **a consequent (then)**

An antecedent is something that's found in data, and a consequent is an item that is found in combination with the antecedent. Have a look at this rule for instance:

"If a customer buys bread, he's 70% likely of buying milk."

In the above association rule, bread is the antecedent and milk is the consequent. Simply put, it can be understood as a retail store's association rule to target their customers better. If the above rule is a result of a thorough analysis of some data sets, it can be used to not only improve customer service but also improve the company's revenue.

Association rules are created by thoroughly analyzing data and looking for frequent if/then patterns. Then, depending on the following two parameters, the important relationships are observed:

1. **Support**: Support indicates how frequently the if/then relationship appears in the database.
2. **Confidence**: Confidence tells about the number of times these relationships have been found to be true.

Association Rule Mining is sometimes referred to as "Market Basket Analysis", as it was the first application area of association mining. The aim is to discover associations of items occurring together more often than you'd expect from randomly sampling all the possibilities. The classic anecdote of Beer and Diaper will help in understanding this better.

Apriori Algorithm:

An algorithm known as Apriori is a common one in data mining. It's used to identify the most frequently occurring elements and meaningful associations in a dataset.

In 1994, R. Agrawal and R. Srikant developed the Apriori method for identifying the most frequently occurring itemsets in a dataset using the boolean association rule. Since it makes use of previous knowledge about common itemset features, the method is referred to as Apriori. This is achieved by the use of an iterative technique or level-wise approach, in which k-frequent itemsets are utilized to locate k+1 itemsets.

An essential feature known as the Apriori property is utilized to boost the effectiveness of level-wise production of frequent itemsets. This property helps by minimizing the search area, which in turn serves to maximize the productivity of level-wise creation of frequent patterns.

Working of apriori algorithm:

The Apriori algorithm operates on a straightforward premise. When the support value of an item set exceeds a certain threshold, it is considered a frequent item set. Take into account the following steps. To begin, set the support criterion, meaning that only those things that have more than the support criterion are considered relevant.

- Step 1: Create a list of all the elements that appear in every transaction and create a frequency table.
- Step 2: Set the minimum level of support. Only those elements whose support exceeds or equals the threshold support are significant.
- Step 3: All potential pairings of important elements must be made, bearing in mind that AB and BA are interchangeable.
- Step 4: Tally the number of times each pair appears in a transaction. • Step 5:

Only those sets of data that meet the criterion of support are significant.

Implementation:

a) Using Rapid Miner

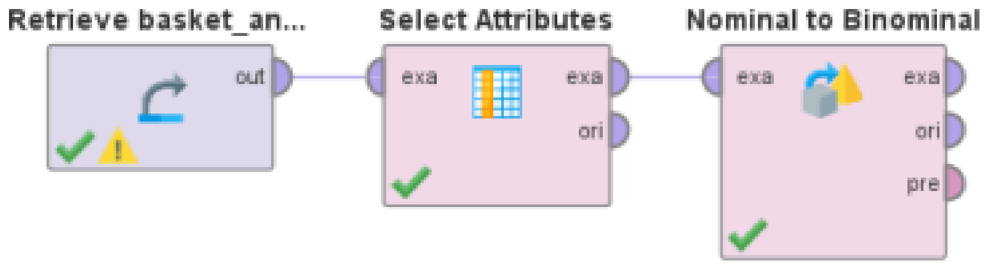
1. Before Preprocessing:

Open in Turbo Prep | Auto Model | Interactive Analysis   Filter (999 / 999 examples): all

Row No.	att1	Apple	Bread	Butter	Cheese	Corn	Dill	Eggs	Ice cream	Kidney B
1	0	False	True	False	False	True	True	False	True	False
2	1	False	False	False	False	False	False	False	False	False
3	2	True	False	True	False	False	True	False	True	False
4	3	False	False	True	True	False	True	False	False	False
5	4	True	True	False	False	False	False	False	False	False
6	5	True	True	True	True	False	True	False	True	False
7	6	False	False	True	False	False	False	True	True	True
8	7	True	False	False	True	False	False	True	False	False
9	8	True	False	False	False	True	True	True	True	False
10	9	True	False	False	False	False	True	True	True	False
11	10	True	True	False	True	False	False	False	True	False
12	11	True	True	False	True	False	True	True	True	True
13	12	False	False	False	False	False	False	False	False	False
14	13	False	False	False	False	False	False	False	True	False
15	14	False	False	True	False	False	False	False	False	True
16	15	False	False	False	False	False	False	False	False	False
17	16	True	True	True	True	False	False	False	True	False

ExampleSet (999 examples, 0 special attributes, 17 regular attributes)

2. Preprocessing the dataset:



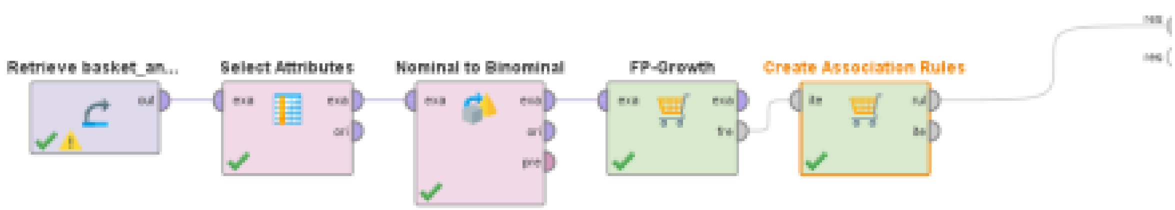
3. After Preprocessing:

Open in Turbo Prep Auto Model Interactive Analysis Filter (999 / 999 examples): all

Row No.	Apple	Bread	Butter	Cheese	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk
1	False	True	False	False	True	True	False	True	False	False
2	False	False	False	False	False	False	False	False	False	True
3	True	False	True	False	False	True	False	True	False	True
4	False	False	True	True	False	True	False	False	False	True
5	True	True	False	False	False	False	False	False	False	False
6	True	True	True	True	False	True	False	True	False	False
7	False	False	True	False	False	False	True	True	True	True
8	True	False	False	True	False	False	True	False	False	False
9	True	False	False	False	True	True	True	True	False	True
10	True	False	False	False	False	True	True	True	False	True
11	True	True	False	True	False	False	False	True	False	False
12	True	True	False	True	False	True	True	True	True	False
13	False	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	True	False	False
15	False	False	True	False	False	False	False	False	True	False
16	False	False	False	False	False	False	False	False	False	False
17	True	True	True	True	False	False	False	True	False	False

ExampleSet (999 examples,0 special attributes,10 regular attributes)

4. Applying apriori algorithm and generating association rules:



Confidence= 0.6

Support= 0.8

Output:

Association rules:

Around 400 rules were generated.

Show rules matching

all of these conclusions:

Bread
Dill
Corn
Sugar
Ice cream
Yogurt
chocolate

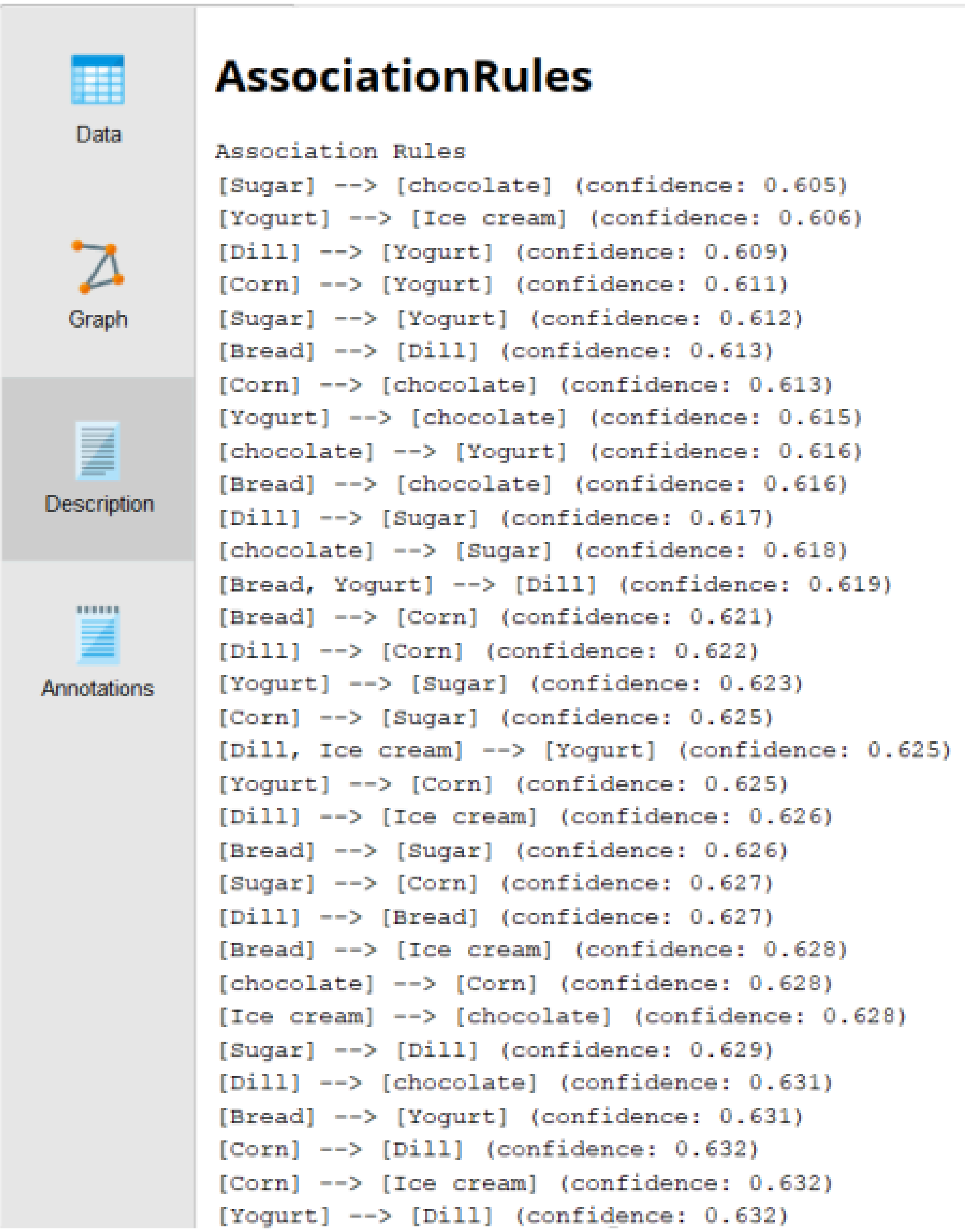
Min. Criterion:

confidence

Min. Criterion Value:

No.	Premises	Conclusion	Support	Confidence
11	Dill	Sugar	0.371	0.617
12	chocolate	Sugar	0.357	0.618
13	Bread, Yogurt	Dill	0.240	0.619
14	Bread	Corn	0.382	0.621
15	Dill	Corn	0.374	0.622
16	Yogurt	Sugar	0.361	0.623
17	Corn	Sugar	0.370	0.625
18	Dill, Ice cream	Yogurt	0.235	0.625
19	Yogurt	Corn	0.362	0.625
20	Dill	Ice cream	0.376	0.626
21	Bread	Sugar	0.385	0.626
22	Sugar	Corn	0.370	0.627
23	Dill	Sugar [illegible]ad	0.377	0.627
24	Bread	Ice cream	0.386	0.628
25	chocolate	Corn	0.363	0.628
26	Ice cream	chocolate	0.370	0.628
27	Sugar	Dill	0.371	0.629
28	Dill	chocolate	0.379	0.631

Support	Confidence	LaPlace	Gain	p-s	Lift	Convict...
0.371	0.617	0.856	-0.832	0.016	1.045	1.070
0.357	0.618	0.860	-0.800	0.016	1.046	1.071
0.240	0.619	0.893	-0.537	0.007	1.028	1.044
0.382	0.621	0.856	-0.849	0.018	1.048	1.075
0.374	0.622	0.858	-0.829	0.018	1.050	1.079
0.361	0.623	0.862	-0.798	0.019	1.056	1.087
0.370	0.625	0.860	-0.815	0.020	1.058	1.092
0.235	0.625	0.897	-0.51[illegible] -0.8148148148148148	[illegible]	[illegible].078	1.121
0.362	0.625	0.862	-0.797	0.019	1.055	1.087
0.376	0.626	0.859	-0.827	0.022	1.061	1.096
0.385	0.626	0.857	-0.846	0.022	1.060	1.095
0.370	0.627	0.862	-0.811	0.020	1.058	1.093
0.377	0.627	0.860	-0.826	0.007	1.019	1.031
0.386	0.628	0.858	-0.845	0.023	1.065	1.102
0.363	0.628	0.864	-0.794	0.021	1.060	1.095
0.370	0.628	0.862	-0.809	0.029	1.086	1.133
0.371	0.629	0.862	-0.810	0.016	1.045	1.073
0.379	0.631	0.861	-0.824	0.031	1.090	1.141



AssociationRules

```
Association Rules
[Sugar] --> [chocolate] (confidence: 0.605)
[Yogurt] --> [Ice cream] (confidence: 0.606)
[Dill] --> [Yogurt] (confidence: 0.609)
[Corn] --> [Yogurt] (confidence: 0.611)
[Sugar] --> [Yogurt] (confidence: 0.612)
[Bread] --> [Dill] (confidence: 0.613)
[Corn] --> [chocolate] (confidence: 0.613)
[Yogurt] --> [chocolate] (confidence: 0.615)
[chocolate] --> [Yogurt] (confidence: 0.616)
[Bread] --> [chocolate] (confidence: 0.616)
[Dill] --> [Sugar] (confidence: 0.617)
[chocolate] --> [Sugar] (confidence: 0.618)
[Bread, Yogurt] --> [Dill] (confidence: 0.619)
[Bread] --> [Corn] (confidence: 0.621)
[Dill] --> [Corn] (confidence: 0.622)
[Yogurt] --> [Sugar] (confidence: 0.623)
[Corn] --> [Sugar] (confidence: 0.625)
[Dill, Ice cream] --> [Yogurt] (confidence: 0.625)
[Yogurt] --> [Corn] (confidence: 0.625)
[Dill] --> [Ice cream] (confidence: 0.626)
[Bread] --> [Sugar] (confidence: 0.626)
[Sugar] --> [Corn] (confidence: 0.627)
[Dill] --> [Bread] (confidence: 0.627)
[Bread] --> [Ice cream] (confidence: 0.628)
[chocolate] --> [Corn] (confidence: 0.628)
[Ice cream] --> [chocolate] (confidence: 0.628)
[Sugar] --> [Dill] (confidence: 0.629)
[Dill] --> [chocolate] (confidence: 0.631)
[Bread] --> [Yogurt] (confidence: 0.631)
[Corn] --> [Dill] (confidence: 0.632)
[Corn] --> [Ice cream] (confidence: 0.632)
[Yogurt] --> [Dill] (confidence: 0.632)
```

b) Using python

1. Loading the dataset:

```python
import pandas as pd
df=pd.read_csv("basket_analysis.csv")
```

```python
[2] df.head()
```

	Unnamed: 0	Apple	Bread	Butter	Cheese	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Sugar	Unicorn	Yogurt	chocolate
0	0	False	True	False	False	True	True	False	True	False	False	False	False	True	False	True	True
1	1	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False
2	2	True	False	True	False	False	True	False	True	False	True	False	False	False	False	True	True
3	3	False	False	True	True	False	True	False	False	False	True	True	True	False	False	False	False
4	4	True	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False

2. Preprocessing

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    999 non-null    int64
 1   Apple         999 non-null    bool
 2   Bread         999 non-null    bool
 3   Butter        999 non-null    bool
 4   Cheese        999 non-null    bool
 5   Corn          999 non-null    bool
 6   Dill          999 non-null    bool
 7   Eggs          999 non-null    bool
 8   Ice cream     999 non-null    bool
 9   Kidney Beans  999 non-null    bool
 10  Milk          999 non-null    bool
 11  Nutmeg        999 non-null    bool
 12  Onion         999 non-null    bool
 13  Sugar         999 non-null    bool
 14  Unicorn       999 non-null    bool
 15  Yogurt        999 non-null    bool
 16  chocolate     999 non-null    bool
dtypes: bool(16), int64(1)
memory usage: 23.5 KB
```

```
[4] cols=['Unnamed: 0']
    df=df.drop(cols,axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 16 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Apple         999 non-null    bool
 1   Bread         999 non-null    bool
 2   Butter        999 non-null    bool
 3   Cheese        999 non-null    bool
 4   Corn          999 non-null    bool
 5   Dill          999 non-null    bool
 6   Eggs          999 non-null    bool
 7   Ice cream     999 non-null    bool
 8   Kidney Beans  999 non-null    bool
 9   Milk          999 non-null    bool
 10  Nutmeg        999 non-null    bool
 11  Onion         999 non-null    bool
 12  Sugar         999 non-null    bool
 13  Unicorn       999 non-null    bool
 14  Yogurt        999 non-null    bool
 15  chocolate     999 non-null    bool
dtypes: bool(16)
memory usage: 15.7 KB
```

3. Applying apriori algorithm and generating association rules.

```
[54] from mlxtend.frequent_patterns import apriori, association_rules
     frequent_itemsets = apriori(df, min_support=0.1, use_colnames=True)
```

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
```

```
[57] print("Frequent Itemsets:")
     print(frequent_itemsets.head())

     print("\nAssociation Rules:")
     print(rules.head(20))
```

Printing 20 association rules. There were a total 97 rules generated.

Output:

```
Frequent Itemsets:
    support  itemsets
0  0.383383   (Apple)
1  0.384384   (Bread)
2  0.420420  (Butter)
3  0.404404  (Cheese)
4  0.407407    (Corn)
```

```
Association Rules:
                   antecedents     consequents  antecedent support  \
0                      (Bread)        (Yogurt)            0.384384
1                  (Ice cream)        (Butter)            0.410410
2                       (Dill)     (chocolate)            0.398398
3                  (chocolate)          (Milk)            0.421421
4                       (Milk)     (chocolate)            0.405405
5              (Butter, Sugar)         (Apple)            0.196196
6              (Butter, Apple)         (Sugar)            0.188188
7               (Sugar, Apple)        (Butter)            0.182182
8               (Butter, Corn)     (Ice cream)            0.191191
9            (Corn, Ice cream)        (Butter)            0.192192
10       (Kidney Beans, Corn)        (Butter)            0.195195
11     (Kidney Beans, Butter)          (Corn)            0.202202
12              (Butter, Corn)  (Kidney Beans)            0.191191
13     (Kidney Beans, Butter)     (Ice cream)            0.202202
14  (Kidney Beans, Ice cream)        (Butter)            0.196196
15        (Butter, Ice cream)  (Kidney Beans)            0.207207
16           (Butter, Nutmeg)     (Ice cream)            0.198198
17        (Nutmeg, Ice cream)        (Butter)            0.187187
18            (Butter, Onion)     (Ice cream)            0.197197
19         (Onion, Ice cream)        (Butter)            0.192192
```

	consequent support	support	confidence	lift	leverage	conviction
0	0.420420	0.193193	0.502604	1.195480	0.031590	1.165228
1	0.420420	0.207207	0.504878	1.200889	0.034662	1.170579
2	0.421421	0.199199	0.500000	1.186461	0.031306	1.157157
3	0.405405	0.211211	0.501188	1.236263	0.040365	1.192021
4	0.421421	0.211211	0.520988	1.236263	0.040365	1.207857
5	0.383383	0.100100	0.510204	1.330793	0.024882	1.258926
6	0.409409	0.100100	0.531915	1.299225	0.023054	1.261716
7	0.420420	0.100100	0.549451	1.306907	0.023507	1.286384
8	0.410410	0.102102	0.534031	1.301213	0.023635	1.265299
9	0.420420	0.102102	0.531250	1.263616	0.021301	1.236436
10	0.420420	0.101101	0.517949	1.231978	0.019037	1.202319
11	0.407407	0.101101	0.500000	1.227273	0.018722	1.185185
12	0.408408	0.101101	0.528796	1.294772	0.023017	1.255489
13	0.410410	0.110110	0.544554	1.326853	0.027124	1.294534
14	0.420420	0.110110	0.561224	1.334913	0.027625	1.320902
15	0.408408	0.110110	0.531401	1.301151	0.025485	1.262469
16	0.410410	0.102102	0.515152	1.255211	0.020759	1.216029
17	0.420420	0.102102	0.545455	1.297403	0.023405	1.275075
18	0.410410	0.103103	0.522843	1.273951	0.022171	1.235629
19	0.420420	0.103103	0.536458	1.276004	0.022302	1.250329

Conclusion:

Hence, we have understood association rule mining by implementing it in Rapid Miner and Python. In python we got comparatively less rules.