Apriori algorithm

It is also helpful in data mining.

If you want to find if a person buy x --→ also buy y

e.g if person like movie 1 -→also like move2

This algorithm helps to find these association rules.

Example

Bear ---→ diaper algorithm

| User ID | Movies liked |
|---|---|
| 46578 | Movie1, Movie2, Movie3, Movie4 |
| 98989 | Movie1, Movie2 |
| 71527 | Movie1, Movie2, Movie4 |
| 78981 | Movie1, Movie2 |
| 89192 | Movie2, Movie4 |
| 61557 | Movie1, Movie3 |

Potential Rules:

Movie1 ⇨ Movie2

Movie2 ⇨ Movie4

Movie1 ⇨ Movie3

| Transaction ID | Products purchased |
|---|---|
| 46578 | Burgers, French Fries, Vegetables |
| 98989 | Burgers, French Fries, Ketchup |
| 71527 | Vegetables, Fruits |
| 78981 | Pasta, Fruits, Butter, Vegetables |
| 89192 | Burgers, Pasta, French Fries |
| 61557 | Fruits, Orange Juice, Vegetables |
| 87923 | Burgers, French Fries, Ketchup, Mayo |

Potential Rules:

Burgers ⇨ French Fries

Vegetables ⇨ Fruits

Burgers, French Fries ⇨ Ketchup

While finding these rules we need to find strong support for these rules

For this rule finding we can use following measures

1. Support
2. Confidence

3. Lift
Formulea:

Support: users who watched movie m/total number of users

Movie Recommendation: $$\text{support}(M) = \frac{\#\text{ user watchlists containing } M}{\#\text{ user watchlists}}$$

Market Basket Optimisation: $$\text{support}(I) = \frac{\#\text{ transactions containing } I}{\#\text{ transactions}}$$

If there are total 100 people and out of those if 10 has seen movie1
Then support for movie1 = 10/100

Confidence: people who have seen movie 1 also has seen movie2

Movie Recommendation: $$\text{confidence}(M_1 \rightarrow M_2) = \frac{\#\text{ user watchlists containing } M_1 \text{ and } M_2}{\#\text{ user watchlists containing } M_1}$$

Market Basket Optimisation: $$\text{confidence}(I_1 \rightarrow I_2) = \frac{\#\text{ transactions containing } I_1 \text{ and } I_2}{\#\text{ transactions containing } I_1}$$

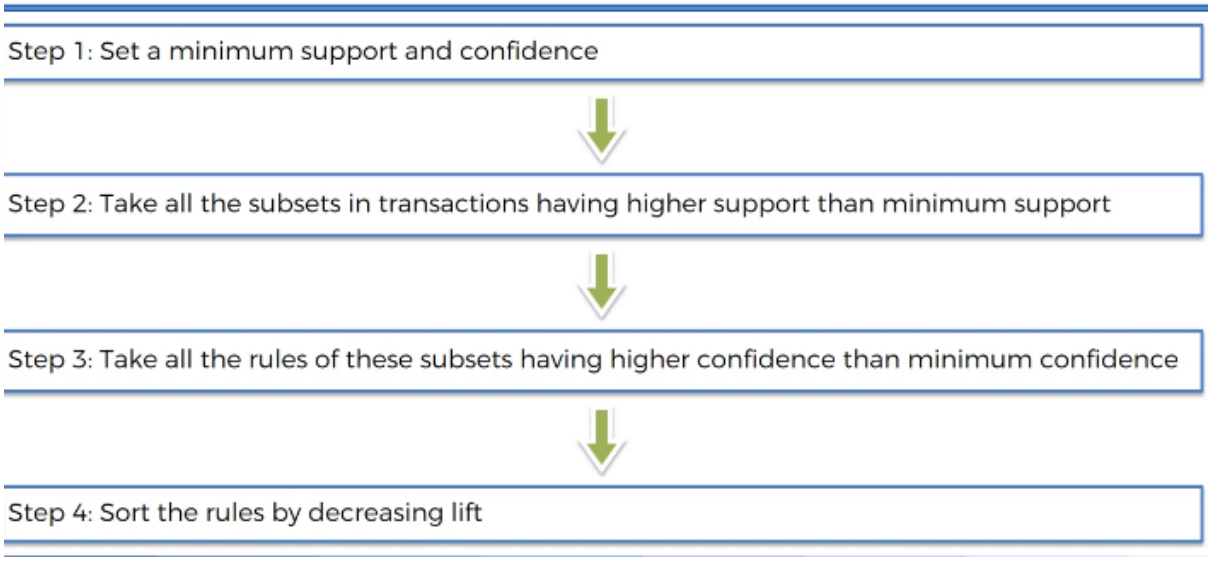
Example if 70 people have seen movie1 and out of them 10 has seen movie 2 also
Then confidence for movie 1 is 10/70
Lift : It is usually similar to naïve bayes algorithm
Example :
If we have 100 people
Suppose 40 have seen movie 1 and out of that 7 people have seen Movie 2 also
3 more people have seen Movie 2
So if we recommend Movie 1 to these people what will be the likelihood that they will like to see the movie.

i.e called as lift
basically lift is

Movie Recommendation: $$\text{lift}(M_1 \rightarrow M_2) = \frac{\text{confidence}(M_1 \rightarrow M_2)}{\text{support}(M_2)}$$

Market Basket Optimisation: $$\text{lift}(I_1 \rightarrow I_2) = \frac{\text{confidence}(I_1 \rightarrow I_2)}{\text{support}(I_2)}$$

Step by step




In this we are going to use apriory.py file downloaded from python foundation
We are not using any predefined package
Basically this file contains some classes and functions

The file we will use is market-basket-optimization.csv. In the file no header is available.
Every row shows one transaction
Each column contains separate product available in the store

But the apriory.py file need i/p in list of list fomat. So first we need to convert it into List of list and then pass it to class
So use nested for loop

Transactions=[]

For I in range(7501) : ---------- because total number of rows are 7500
Transaction.append(str(dataset.values[I,j]) for j in range(20)])
Str will put values in " (double quotes)
.values is needed to retrieve the value.

Rules=apriori(transaction, min-support= ,min-confidence= , min-lift=, min-length=2)

Min_length-→
Consider only transaction with minimum 2 items in it.
Min-support-→ we want product atleast bought 3 times a day(3*4)=28/total no of transactions i.e 7500 we can decide such values based on our application and based on
So support = (7*4)/7500=0.003 by experience we may modify it.
Min-confidence--→0.20 by experienced we find better value
Min-lift--→3
We need to find these values by trial error