

Assignment No-3  
Title: Apriori Algorithm.

Problem Statement: Apply a-priori algorithm to find frequently occurring items from given data and generate strong association rules using support and confidence thresholds. For Example: Market Basket Analysis.

S/W & H/W : Jupyter Notebook, 2 GB RAM, 500 GB HDD.  
requirements

Learning Objectives: Use Apriori functions to implement Apriori Algorithm.

Learning Outcomes: Mining of frequent itemsets and create or generate relevant association rules.

Theory:

With the quick growth in e-commerce applications, there is an accumulation vast quantity of data in months not in years. Data Mining, also known as Knowledge Discovery in Databases (KDD), to find anomalies, correlations, patterns, and trends to predict outcomes.

Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets & relevant association rules. It is devised to operate

on a database containing a lot of transactions, for instance, items brought by customers in a store.

### ~~Algorithm~~ Apriori Algorithm - The Theory.

Three significant components comprise the apriori algorithm.

- Support
- Confidence
- Lift

eg. As mentioned earlier, you need big DB.

Suppose you have 2000 customers transactions in a store. You have to find support, confidence, & lift of two items, say bread & jam. It is because people frequently bundle these two items together.

Out of 200 transactions, 200 contain jam whereas 300 contain bread. These 300 transactions include a 100 that includes bread as well as jam. Using this data, we shall find out the support, confidence & lift.

• **Support** - Support is the default popularity of any item. You calculate Support as a quotient of the division of the number of transactions containing that item by total number of transactions.



$$\text{Support (Jam)} = (\text{Transactions involving jam}) / (\text{Total Transactions})$$

$$= 200/2000 = 10\%$$

• Confidence - It is the likelihood that customers bought both bread & jam. Dividing number of transactions that include both bread and jam by total number of transactions will give the Confidence figure.

$$\text{Confidence} = (\text{Transactions having both bread \& jam}) / (\text{Total Transactions having jam})$$

$$= 100/200 = 50\%$$

It implies 50% of customers who bought jam bought bread as well.

• Lift - It is the increase in the ratio of sale of bread when you sell jam.

$$\text{Lift} = (\text{confidence (Jam - Bread)}) / (\text{Support (Jam)})$$

$$= 50/10 = 5$$

It says chance of customer buying both jam & bread together is 5 times more than jam alone. If lift value is less than 1, customers are unlikely to buy both items together.

Algorithm-

- (1) Apriori (PharmGKB,  $\epsilon$ )
- (2)  $L_1 \leftarrow$  (frequent genes in drug class for Alzheimer's disease)
- (3)  $K \leftarrow 2$
- (4) while  $L_{K-1} \neq \phi$

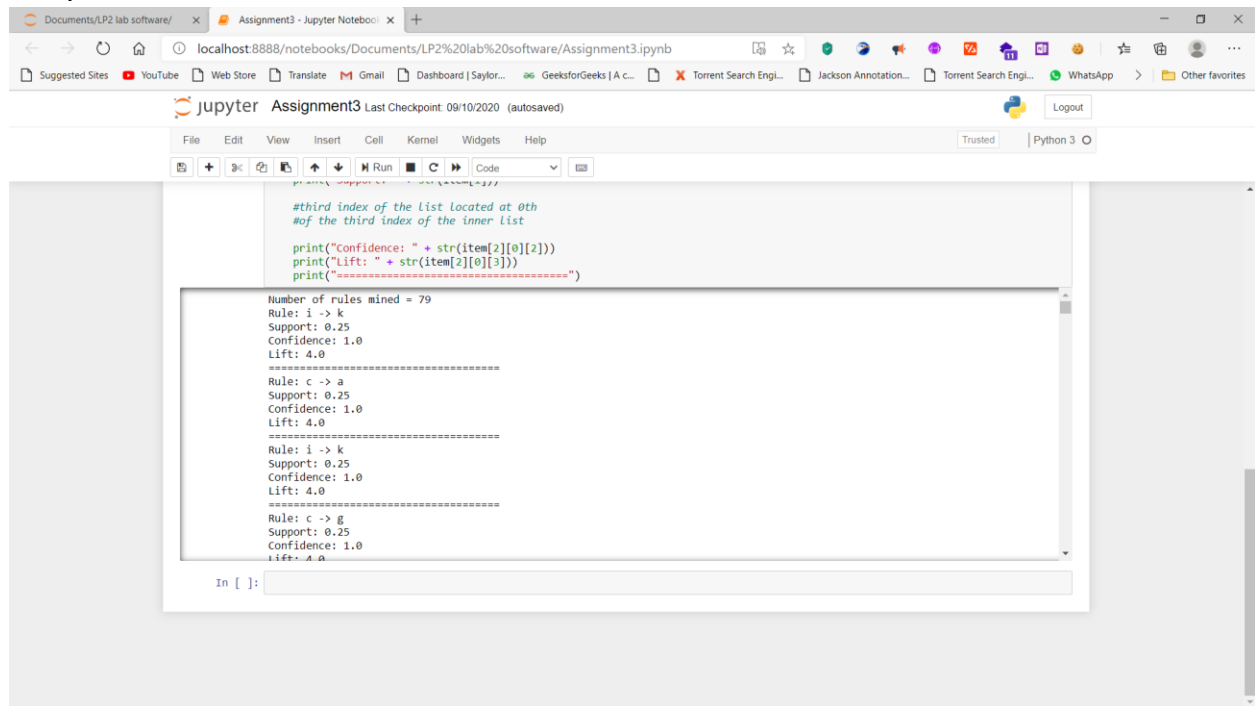
(5)  $C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \in U L_{k-1} \wedge b \notin a\}$   
 (6) for each drug class  $\in$  PharmGKB  
 (7)  $C_k \leftarrow \{gene \mid gene \in C_k \wedge gene \subseteq \text{drug class}\}$   
 (8) for each candidate gene  $\in C_k$   
 (9)  $\text{count}[gene] \leftarrow \text{count}[gene] + 1$   
 (10) end for  
 (11) end for  
 (12)  $L_k \leftarrow \{gene \mid gene \in C_k \wedge \text{count}[gene] > \epsilon\}$   
 (13)  $K \leftarrow K + 1$   
 (14) end while  
 (15) return  $\bigcup_{k=1}^K L_k$

### Test Cases

Sr.no	Description	Expected o/p	Actual o/p
1)	Install Apriori library	success	success
2)	Preprocess data, find the total transactions & append them.	success	success
3)	Training of Apriori on the dataset.	success	success
4)	Visualise the results	success	success
5)	Create association Rules	success	success

Conclusion: Thus, we implemented apriori algorithm using python for given transaction input.

## Output:



The screenshot shows a Jupyter Notebook interface in a web browser. The browser's address bar displays the URL `localhost:8888/notebooks/Documents/LP2%20lab%20software/Assignment3.ipynb`. The notebook's title bar indicates the file is `Assignment3` and it was last checkpointed on 09/10/2020. The interface includes a menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations and a 'Run' button. The code editor contains the following Python code:

```
print("Support: " + str(item[2][0][2]))  
  
#third index of the List Located at 0th  
#of the third index of the inner List  
  
print("Confidence: " + str(item[2][0][2]))  
print("Lift: " + str(item[2][0][3]))  
print("=====")
```

The output area below the code shows the results of the execution:

```
Number of rules mined = 79  
Rule: i -> k  
Support: 0.25  
Confidence: 1.0  
Lift: 4.0  
=====  
Rule: c -> a  
Support: 0.25  
Confidence: 1.0  
Lift: 4.0  
=====  
Rule: i -> k  
Support: 0.25  
Confidence: 1.0  
Lift: 4.0  
=====  
Rule: c -> g  
Support: 0.25  
Confidence: 1.0  
Lift: 4.0
```

At the bottom of the output area, there is a prompt `In [ ]:` followed by an empty input field.