**pROJECT**

**Note: Webmlxtend is a Python library that provides useful tools and extensions for data science tasks, such as classifiers, data loading, and plotting. Learn how to use mlxtend with examples, documentation, and citation information.**

**START:**

**Step1: Import Libraries**

**import pandas as pd**

**retail\_dataset = pd.read\_csv(r"C:\Users\GT-499\Downloads\Groceries\_dataset.csv")**

**retail\_dataset**

**STEP2:**

**pip install mlxtend**

**Step3:EDA**

**retail\_dataset.info()**

**retail\_dataset.head()**

**STEP4:**

**unique\_values = retail\_dataset['itemDescription'].unique()**

**unique\_values.size**

**retail\_dataset[retail\_dataset.duplicated]**

**#Dataset contains duplicated values**

**unique\_combinations = retail\_dataset.groupby(['Member\_number', 'Date']).agg({'itemDescription': list}).reset\_index()**

**unique\_combinations**

**step5:**

**# create a list using each transations. This will later encode using transactionencoder for apriori algorithm.**

**transaction = unique\_combinations['itemDescription'].tolist()**

**transaction[0]**

**step6:**

**# Install packages for apriori algorithm**

**# pip install mlxtend**

**from mlxtend.preprocessing import TransactionEncoder**

**from mlxtend.frequent\_patterns import apriori, association\_rules**

**step7:**

**# Converting transactions to columns using transacationEncoder**

**te = TransactionEncoder()**

**te\_ary = te.fit\_transform(transaction)**

**dataset = pd.DataFrame(te\_ary, columns=te.columns\_)**

**dataset**

**STEP8:**

**# Convert dataset into 1-0 encoding**

**def encode\_units(x):**

**if x == False:**

**return 0**

**if x == True:**

**return 1**

**dataset = dataset.applymap(encode\_units)**

**dataset.head(10)**

**STEP9:**

**# Apply apriori algorithm on transformed dataset with given confidence and support value**

**frequent\_itemsets = apriori(dataset, min\_support=.10,max\_len=2, use\_colnames=True)**

**rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=.20)**

**rules.head(20)**

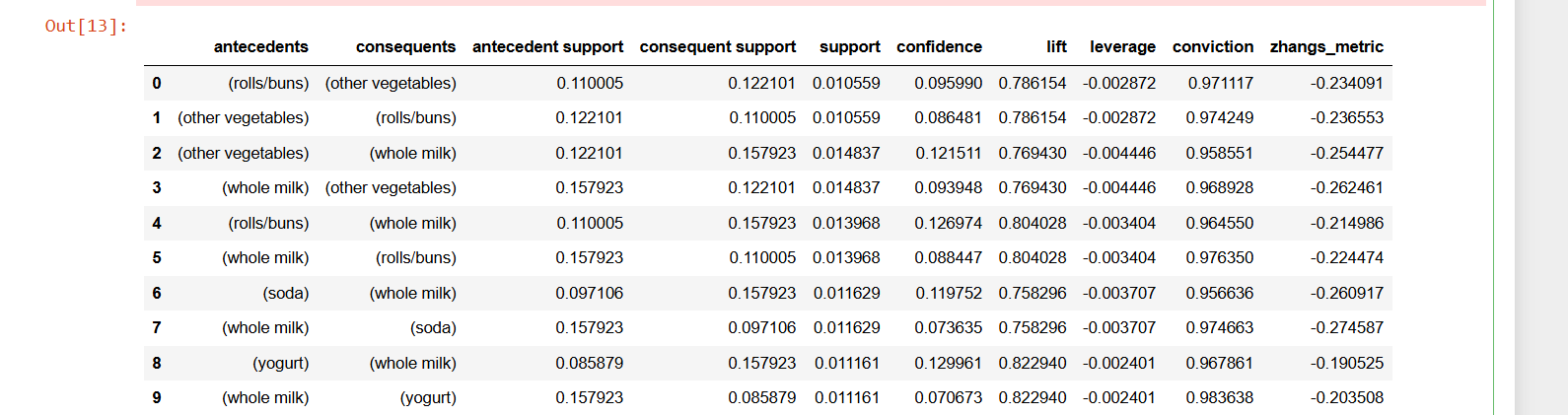
**Last Step**

**frequent\_itemsets = apriori(dataset, min\_support=.01,max\_len=2, use\_colnames=True)**

**rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0)**

**rules**

**Output**

****

**………………x end of the projectx…………………….**

**Terminologies used in the project**

 Pandas **groupby** is used for grouping the data according to the categories and applying a function to the categories.

Unique Combination:

The agg() method allows you to apply a function or a list of function names to be executed along one of the axis of the DataFrame, default 0, which is the index (row) axis.

The reset\_index() method allows you reset the index back to the default 0, 1, 2 etc indexes. By default this method will keep the "old" idexes in a column named "index", to avoid this, use the drop parameter.

Example:

TransactionEncoder: Convert item lists into transaction data for frequent itemset mining. Encoder class for transaction data in Python lists.

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

Association Rule Learning

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

The association rule learning is one of the very important concepts of [machine learning](https://www.javatpoint.com/machine-learning), and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put togethe

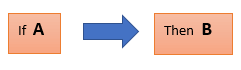
## **How does Association Rule Learning work?**

Note: What do you mean by cardinality?

Cardinality is a mathematical term. It translates into the number of elements in a set. In databases, cardinality refers to the relationships between the data in two database tables

Association rule

Association rule learning works on the concept of If and Else Statement, such as if A then B.



Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known as single cardinality. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

* **Support**
* **Confidence**
* **Lift**

**Let's understand each of them:**

### **Support**

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

Association Rule Learning

### **Confidence**

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

Association Rule Learning

### **Lift**

It is the strength of any rule, which can be defined as below formula:

Association Rule Learning

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

* If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
* **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
* **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

NOTE:Using and TransactionEncoder object, we can transform this dataset into an array format suitable for typical machine learning APIs. Via the fit method, the TransactionEncoder learns the unique labels in the dataset, and via the transform method, it transforms the input dataset (a Python list of lists) into a one-hot encoded NumPy boolean

## Note: **What is Apriori Algorithm?**

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example,

## **Components of Apriori algorithm**

The given three components comprise the aprori algorithm.

1. Support
2. Confidence
3. Lift
4. Let's take an example to understand this concept.
5. We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.
6. Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

### **Support**

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

### **Confidence**

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 percent of customers who bought biscuits bought chocolates also.

### **Lift**

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

=50/10 = 5

It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

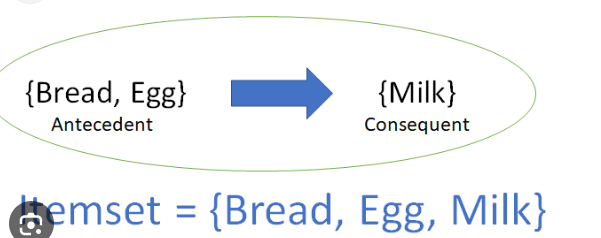
NOTE: What is Min\_support in Apriori?

Minimum-Support is a parameter supplied to the Apriori algorithm in order to prune candidate rules by specifying a minimum lower bound for the Support measure of resulting association rules.

COLUMN\_NAME= True

By default, apriori returns the column indices of the items, which may be useful in downstream operations such as association rule mining. For better readability, we can set use\_colnames=True to convert these integer values into the respective item names:

By default, apriori returns the column indices of the items, which may be useful in downstream operations such as association rule mining. For better readability, we can set use\_colnames=True to convert these integer values into the respective item names:

Note: 

from mlxtend.frequent\_patterns import association\_rules

## **Overview**

Rule generation is a common task in the mining of frequent patterns. An association rule is an implication expression of the form X→Y𝑋→𝑌, where X𝑋 and Y𝑌 are disjoint itemsets [1]. A more concrete example based on consumer behaviour would be {Diapers}→{Beer}{𝐷𝑖𝑎𝑝𝑒𝑟𝑠}→{𝐵𝑒𝑒𝑟} suggesting that people who buy diapers are also likely to buy beer. To evaluate the "interest" of such an association rule, different metrics have been developed. The current implementation make use of the confidence and lift metrics.

### **Metrics**

The currently supported metrics for evaluating association rules and setting selection thresholds are listed below. Given a rule "A -> C", A stands for antecedent and C stands for consequent.

#### **'support':**

support(A→C)=support(A∪C),range: [0,1]

Support displays antecedent support—that is, the proportion of IDs for which the antecedents are true, based on the training data. For example, if 50% of the training data includes the purchase of bread, then the rule bread -> cheese will have an antecedent support of 50%.

leverage(X⇒Y)=support(X∪Y)−support(X)⋅support(Y). This definition is almost the same as for the lift, except that the difference is used instead of the ratio. Thus, there is a close relationship between lift and leverage. In general, leverage slightly favors itemsets with larger supportConviction is the measure of dependence of consequent on antecedent

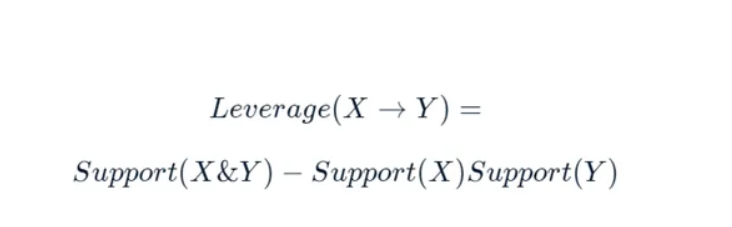
Conviction is defined as, Conviction ( X → Y ) = 1 − Support ( Y ) 1 − Confidence ( X → Y ) = P ( X ) ∗ P ( ˆ Y ) P ( X ∪ ˆ Y ) where, P(ˆY) is the probability that Y does not appear in a transaction.

Zhang's metric is defined as the difference between the confidence metrics of "if A then B" and "if NOT A then B," divided by the maximum of the confidence of "if A then B" and of "if NOT A then B."

Note: Leverage and conviction are some other metrics used to assess the strength and significance of association rules in market basket analysis. Leverage helps identify deviations from independence, while conviction quantifies the dependency between items in an association rule. These metrics, along with support, confidence, and lift, provide a more comprehensive understanding of the relationships between items in transaction data, helping businesses make informed decisions about product placement, cross-selling, and marketing strategies.

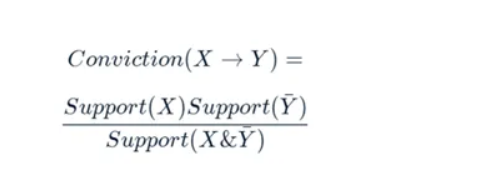
.

Leverage measures the difference between the observed frequency of co-occurrence of two items (X and Y) in transactions and the frequency that would be expected if X and Y were independent of each other. In other words, it quantifies how much the occurrence of X and Y together deviates from what would be expected by chance.



Interpretation:  
— A positive leverage value indicates that the occurrence of X and Y together is more frequent than what would be expected by chance, suggesting a positive association.  
— A leverage value of 0 suggests that the co-occurrence of X and Y is as expected under independence, meaning there is no association.  
— A negative leverage value indicates that the occurrence of X and Y together is less frequent than expected, suggesting a negative association or avoidance.

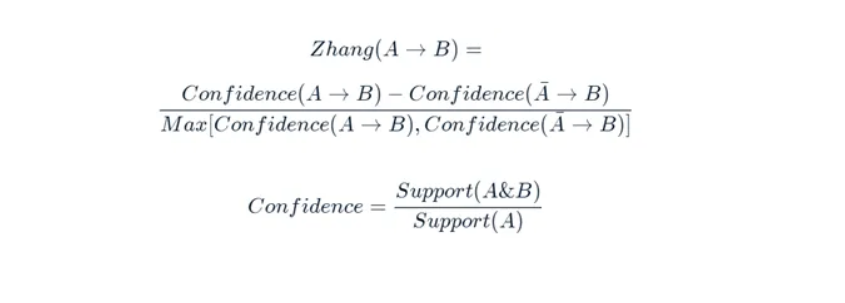
**Conviction**  
Conviction measures the ratio of the expected frequency that X occurs without Y to the observed frequency that X occurs without Y. In essence, it quantifies how much the presence of X implies the absence of Y. X high conviction value indicates that the rule X => Y is highly dependent, and A rarely occurs without B



— A high conviction value (> 1) indicates that the presence of X is strongly dependent on the presence of Y, implying a strong association.  
— A low conviction value (< 1) suggests that the presence of X is not strongly dependent on the presence of Y, implying a weaker association.

**Zhang’s Metric**

Zhang’s metric is a measure designed to assess the strength of association (positive or negative) between two items, taking into account both their co-occurrence and their non-co-occurrence. It is particularly useful when we want to understand how the presence or absence of item A affects the likelihood of item B being present in a transaction.  
In practical terms, Zhang’s metric can be used to assess the relationship between two items or itemsets in transaction data, considering both their co-occurrence and their non-co-occurrence. Positive values indicate a positive association, while negative values suggest a negative association or avoidance. Values close to zero indicate no significant association.



A summary of how Zhang’s metric is calculated:  
Numerator: The numerator of Zhang’s metric is calculated as follows:  
— Compute the support of both items A and B.  
— Subtract the product of the support of A and the support of B.  
— This numerator represents the difference between the observed co-occurrence of A and B and the expected co-occurrence under the assumption of independence.

Denominator: The denominator is a bit more complex and involves multiple terms:  
— Compute the support of both items A and B, multiplied by one minus the support of A.  
— Compute the support of item A, multiplied by the support of item B, minus the support of both items A and B.  
— Determine the maximum value between the two computed terms above.

Zhang’s Metric Calculation: Finally, Zhang’s metric is calculated by dividing the numerator by the denominator. This ensures that the metric’s value falls within the range of -1 (indicating strong negative association) to 1 (indicating strong positive association), with 0 indicating no association.  
Keep in mind that Zhang’s metric offers a nuanced way to measure association and dissociation between items and can be a valuable tool in certain contexts where traditional metrics like confidence and lift may not capture the full extent of item relationships.