Ivan Zepeda

C0883949

IN CLASS ASSIGNMENT

Apriori

In [1]:
```
import pandas as pd

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
['Corn' 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)

df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[1]:

	Apple	Corn	CornOnion	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogur
0	False	False	False	False	True	False	True	True	True	True	False	Tru
1	False	False	False	True	True	False	True	False	True	True	False	Tru
2	True	False	False	False	True	False	True	True	False	False	False	Fals
3	False	True	False	False	False	False	True	True	False	False	True	Tru
4	False	False	True	False	True	True	True	False	False	True	False	Fals

In [2]:
```
from mlxtend.frequent_patterns import apriori

frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

Out[2]:

	support	itemsets
0	0.8	(Eggs)
1	1.0	(Kidney Beans)
2	0.6	(Milk)
3	0.6	(Onion)
4	0.6	(Yogurt)
5	0.8	(Eggs, Kidney Beans)
6	0.6	(Eggs, Onion)
7	0.6	(Milk, Kidney Beans)
8	0.6	(Onion, Kidney Beans)
9	0.6	(Yogurt, Kidney Beans)
10	0.6	(Eggs, Onion, Kidney Beans)

In [3]:

```
from mlxtend.frequent_patterns import association_rules
res = association_rules(frequent_itemsets, metric= "confidence", min_threshold
res
```

Out[3]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	c
0	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.00	1.00	0.00	
1	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.80	1.00	0.00	
2	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	
3	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	
4	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	
5	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	
6	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	
7	(Eggs, Onion)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00	
8	(Eggs, Kidney Beans)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12	
9	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12	
10	(Eggs)	(Onion, Kidney Beans)	0.8	0.6	0.6	0.75	1.25	0.12	
11	(Onion)	(Eggs, Kidney Beans)	0.6	0.8	0.6	1.00	1.25	0.12	

In [4]:

```
res1 = res[['antecedents','consequents','support','confidence','lift']]
res1
```

Out[4]:

	antecedents	consequents	support	confidence	lift
0	(Eggs)	(Kidney Beans)	0.8	1.00	1.00
1	(Kidney Beans)	(Eggs)	0.8	0.80	1.00
2	(Eggs)	(Onion)	0.6	0.75	1.25
3	(Onion)	(Eggs)	0.6	1.00	1.25
4	(Milk)	(Kidney Beans)	0.6	1.00	1.00
5	(Onion)	(Kidney Beans)	0.6	1.00	1.00
6	(Yogurt)	(Kidney Beans)	0.6	1.00	1.00
7	(Eggs, Onion)	(Kidney Beans)	0.6	1.00	1.00
8	(Eggs, Kidney Beans)	(Onion)	0.6	0.75	1.25
9	(Onion, Kidney Beans)	(Eggs)	0.6	1.00	1.25
10	(Eggs)	(Onion, Kidney Beans)	0.6	0.75	1.25
11	(Onion)	(Eggs, Kidney Beans)	0.6	1.00	1.25

In [5]:
```
res2 = res1[res1['confidence'] >= 1]
res2
```

Out[5]:

	antecedents	consequents	support	confidence	lift
0	(Eggs)	(Kidney Beans)	0.8	1.0	1.00
3	(Onion)	(Eggs)	0.6	1.0	1.25
4	(Milk)	(Kidney Beans)	0.6	1.0	1.00
5	(Onion)	(Kidney Beans)	0.6	1.0	1.00
6	(Yogurt)	(Kidney Beans)	0.6	1.0	1.00
7	(Eggs, Onion)	(Kidney Beans)	0.6	1.0	1.00
9	(Onion, Kidney Beans)	(Eggs)	0.6	1.0	1.25
11	(Onion)	(Eggs, Kidney Beans)	0.6	1.0	1.25

In [ ]:
```
#  rules['antecedents_len'] = rules['antecedents'].apply(lambda x: len(x))
```

FP growth

// Video on minute 18

In [6]:
```
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules
```

In [8]:
```
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
```

```
['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
['Corn' 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]
```

In [9]:
```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[9]:

	Apple	Corn	CornOnion	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogur
0	False	False	False	False	True	False	True	True	True	True	False	Tru
1	False	False	False	True	True	False	True	False	True	True	False	Tru
2	True	False	False	False	True	False	True	True	False	False	False	Fals
3	False	True	False	False	False	False	True	True	False	False	True	Tru
4	False	False	True	False	True	True	True	False	False	True	False	Fals

In [10]:
```
fpgrowth(df, min_support=0.6)
```

Out[10]:

	support	itemsets
0	1.0	(6)
1	0.8	(4)
2	0.6	(11)
3	0.6	(9)
4	0.6	(7)
5	0.8	(4, 6)
6	0.6	(11, 6)
7	0.6	(9, 4)
8	0.6	(9, 6)
9	0.6	(9, 4, 6)
10	0.6	(6, 7)

In [11]:
```
frequent_itemsets = fpgrowth(df, min_support=0.6, use_colnames=True)
frequent_itemsets
```

Out[11]:

	support	itemsets
0	1.0	(Kidney Beans)
1	0.8	(Eggs)
2	0.6	(Yogurt)
3	0.6	(Onion)
4	0.6	(Milk)
5	0.8	(Eggs, Kidney Beans)
6	0.6	(Yogurt, Kidney Beans)
7	0.6	(Eggs, Onion)
8	0.6	(Onion, Kidney Beans)
9	0.6	(Eggs, Onion, Kidney Beans)
10	0.6	(Milk, Kidney Beans)

In [13]:
```
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold
rules
```

Out[13]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	co
0	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.0	1.00	0.00	
1	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.8	1.00	0.00	
2	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	0.00	
3	(Onion)	(Eggs)	0.6	0.8	0.6	1.0	1.25	0.12	
4	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	0.00	
5	(Eggs, Onion)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	0.00	
6	(Onion, Kidney Beans)	(Eggs)	0.6	0.8	0.6	1.0	1.25	0.12	
7	(Onion)	(Eggs, Kidney Beans)	0.6	0.8	0.6	1.0	1.25	0.12	
8	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.0	1.00	0.00	

In [14]:
```
from mlxtend.frequent_patterns import apriori
%timeit apriori(df, min_support=0.6) #magic function, you can find the exceutic
```

```
3 ms ± 212 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [15]:
```
#from mlxtend.fequent_patterns import fpgrowth
%timeit fpgrowth(df, min_support=0.6)
```

```
1.09 ms ± 41.2 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

FPGrowth is faster than apriori, but its downside, you have to build the construction of the 'tree'?

In [ ]:

In [ ]:

In class activities

WHat are some applications of association rules

Market Basket Analysis: Association rules are extensively used in retail and e-commerce for market basket analysis. By analyzing customer purchase patterns, retailers can identify which items are frequently purchased together. This information helps in cross-selling, inventory management, and optimizing product placement.

Recommender Systems: Association rules play a vital role in recommender systems. By capturing item associations, these systems can suggest related or complementary items to users. Collaborative filtering and content-based recommendation techniques often employ association rules to provide personalized recommendations.

Customer Behavior Analysis: Association rules are employed in customer behavior analysis to gain insights into purchasing habits, preferences, and product affinities. Marketers can identify customer segments, understand cross-category relationships, and personalize marketing campaigns based on these associations.

Web Usage Mining: Association rules help analyze web browsing patterns and user navigation on websites. This information can be utilized for website optimization, content placement, targeted advertising, and improving user experience.

Healthcare and Medical Research: In healthcare settings, association rules help discover relationships between medical conditions, symptoms, treatments, and patient attributes. They assist in identifying co-occurring diseases, predicting disease outcomes, understanding drug interactions, and supporting medical decision-making.

Fraud Detection: Association rules are valuable in detecting fraudulent activities or suspicious behavior in various domains, such as credit card fraud, insurance claims, and network intrusion detection. Associations among unusual patterns can highlight potential fraudulent or anomalous transactions.

Bioinformatics: Association rules are used in analyzing genetic datasets to discover patterns and associations among genetic markers, diseases, and phenotypes. This aids in

understanding genetic predispositions, identifying disease markers, and facilitating personalized medicine.

Social Network Analysis: Association rules are useful in analyzing social media networks to uncover relationships, patterns, and trends. They help in understanding social connections, recommending friends or connections, and detecting communities of interest.

What are some terminologies and definitions of association rule

Here are some terminologies and definitions related to association rules:

1. Association Rule: An association rule describes the relationships or patterns between items in a dataset. It consists of an antecedent (or left-hand side) and a consequent (or right-hand side). For example, "If {milk, bread} then {eggs}" is an association rule.
2. Support: Support measures the frequency or occurrence of an itemset in a dataset. It indicates how frequently an itemset appears in the dataset. It is calculated as the ratio of transactions containing the itemset to the total number of transactions.
3. Confidence: Confidence measures the reliability or strength of an association rule. It indicates the likelihood that the consequent will appear in a transaction given that the antecedent is present. It is calculated as the ratio of the number of transactions containing both the antecedent and consequent to the number of transactions containing the antecedent.
4. Lift: Lift measures the strength of association between the antecedent and consequent. It compares the confidence of a rule with the expected confidence if the antecedent and consequent were independent. A lift greater than 1 indicates a positive relationship, while a lift less than 1 indicates a negative or coincidental relationship.
5. Itemset: An itemset refers to a collection of one or more items in a transaction. It can be a single item (singleton itemset) or a combination of multiple items (multi-itemset).
6. Frequent Itemset: A frequent itemset is an itemset whose support exceeds a predefined minimum support threshold. It represents the sets of items that occur together frequently in the dataset.
7. Apriori Algorithm: Apriori algorithm is a popular algorithm for mining frequent itemsets and generating association rules. It works by iteratively discovering frequent itemsets and constructing association rules based on the discovered itemsets.
8. MinSup: MinSup (minimum support) is a user-defined threshold used in association rule mining to determine the minimum support level required for an itemset to be considered frequent.

These terminologies and definitions are commonly used in association rule mining to analyze and identify patterns in datasets. 😉

What is one-dimensional,multi-dimensioan! and hybric-dimensional association rule

One-Dimensional Association Rule: In a broad sense, a one-dimensional association rule could refer to a rule that involves a single attribute or dimension of the dataset. For example, in a dataset containing customer purchase transactions, a one-dimensional association rule might be "If {Product: Milk} then {Product: Bread}"—focusing on a single product dimension.

Multi-Dimensional Association Rule: Similarly, a multi-dimensional association rule may refer to a rule that involves multiple attributes or dimensions of the dataset. In the customer purchase transaction dataset example, a multi-dimensional association rule could be "If {Product: Milk, Location: Store A} then {Product: Bread, Location: Store B}"—considering both the product and location dimensions.

Hybrid-Dimensional Association Rule: A hybrid-dimensional association rule might imply a combination of different types of dimensions or attributes in a rule. It could involve both categorical and numerical attributes or a mix of discrete and continuous dimensions. This type of rule could capture more complex relationships and patterns within a dataset.

Explain support, confidence and lift

Support: Support is a measure of the frequency or occurrence of an itemset in a dataset. It indicates how often a particular combination of items appears together. The support of an itemset is calculated as the ratio of the number of transactions containing the itemset to the total number of transactions in the dataset. For example, if we have a dataset of 1000 transactions, and the itemset {milk, bread} appears in 150 transactions, then the support of {milk, bread} would be 150/1000 = 0.15 or 15%.

High support values indicate that the itemset is frequently occurring or popular in the dataset.

Confidence: Confidence is a measure of the reliability or strength of an association rule. It indicates the likelihood that the consequent will appear in a transaction given that the antecedent is present. Confidence is calculated as the ratio of the number of transactions containing both the antecedent and the consequent to the number of transactions containing only the antecedent. For example, if we have an association rule {milk} -> {bread} with an antecedent support of 300 transactions and a joint support of 200 transactions, then the confidence of the rule would be 200/300 = 0.67 or 67%.

High confidence values indicate that the rule is dependable and the consequent often occurs when the antecedent is present.

Lift: Lift measures the strength of association between the antecedent and the consequent of an association rule. It compares the observed confidence of the rule with the expected confidence if the antecedent and consequent were independent. Lift is calculated as the

ratio of the confidence of the rule to the expected confidence based on the individual support of the antecedent and consequent.

For example, if the confidence of {milk} -> {bread} is 0.67 and the individual supports of {milk} and {bread} are 0.4 and 0.5, respectively, then the lift would be (0.67 / (0.4 * 0.5)) = 3.35.

A lift value greater than 1 indicates that the antecedent and consequent are positively correlated, meaning they appear together more often than expected by chance. A lift value less than 1 indicates a negative or coincidental relationship, while a lift value equal to 1 suggests independence.

These measures—support, confidence, and lift—are important tools to evaluate the significance and usefulness of association rules discovered in a dataset. They assist in identifying the most interesting and reliable patterns. 😉

What is apriori principle? How is it used to filter itemsets and rules

The Apriori principle is a fundamental concept in association rule mining, specifically in the Apriori algorithm. It aids in efficiently identifying frequent itemsets and generating association rules from large transactional datasets. The principle is based on the observation that if an itemset is infrequent, then its supersets (larger itemsets containing the same items) are also infrequent.

The Apriori algorithm follows a level-wise approach to progressively generate frequent itemsets with increasing sizes. Here's how the Apriori principle is used within the algorithm:

Generating frequent itemsets: The algorithm begins by scanning the dataset to determine the support of individual items (singleton itemsets). Then, it applies the Apriori principle by iteratively joining frequent itemsets of size k-1 to form candidate itemsets of size k. It prunes (filters out) candidate itemsets that contain a subset that is infrequent, as per the Apriori principle. For example, if {milk} and {bread} are both frequent itemsets, the algorithm can join them, yielding {milk, bread} as a candidate itemset. However, if {milk, bread} is found to be infrequent, the Apriori principle guarantees that any superset of {milk, bread} (e.g., {milk, bread, eggs}) will also be infrequent. Hence, it prunes {milk, bread, eggs} without calculating its actual support.

Generating association rules: Once the frequent itemsets are obtained, the Apriori algorithm generates association rules with high confidence. It applies the Apriori principle to filter out candidate rules that do not meet the minimum confidence threshold. For example, if {milk, bread} is a frequent itemset, the algorithm can generate the rule {milk} => {bread}. It checks the confidence of this rule, and if it falls below the minimum threshold, the Apriori principle ensures that any subset of the antecedent or any superset of the consequent will also fall below the threshold. Consequently, those candidate rules are pruned.

By utilizing the Apriori principle for filtering itemsets and rules, the Apriori algorithm significantly reduces the number of unnecessary computations, leading to more efficient and scalable association rule mining.

Remember, the Apriori principle states that if an itemset is infrequent, its supersets will also be infrequent. This property enables the algorithm to eliminate the generation and evaluation of non-frequent itemsets, improving the efficiency of the mining process. 😊

What are the steps 3 main steps in association rules

The main steps involved in generating association rules are as follows:

1. Data Preprocessing: This step involves preparing the dataset for association rule mining. It includes tasks such as data cleaning, data transformation, and handling missing values. Data preprocessing ensures that the dataset is in a suitable format for mining association rules.

2. Frequent Itemset Generation: In this step, frequent itemsets are identified from the dataset. Frequent itemsets are sets of items that appear together in transactions with a frequency greater than a specified minimum support threshold. The popular Apriori algorithm is often utilized for generating frequent itemsets.

    - Candidate Generation: Initially, frequent itemsets of size 1 (individual items) are identified by scanning the dataset. Then, using these frequent itemsets, the algorithm proceeds to generate larger candidate itemsets by joining frequent itemsets of size k-1.

    - Support Counting: After generating the candidate itemsets, the algorithm scans the dataset again to count the support (frequency) of these candidate itemsets. Only the candidate itemsets that meet the minimum support threshold are considered as frequent itemsets.

3. Rule Generation: In this final step, association rules are generated from the frequent itemsets. Association rules express relationships or patterns between different items in the dataset.

    - Confidence Calculation: For each frequent itemset, various possible rules (antecedent → consequent) can be generated. The confidence of a rule is calculated by dividing the support of the entire itemset by the support of the antecedent.

    - Rule Selection: Association rules are evaluated based on their confidence. Rules that satisfy a user-defined minimum confidence threshold are selected as significant rules. Additionally, other measures such as lift, leverage, or conviction can be considered to further evaluate and refine the generated rules based on specific requirements.

These three main steps—data preprocessing, frequent itemset generation, and rule generation—form the foundation of association rule mining. By following these steps, valuable insights and patterns can be discovered from transactional datasets. 😊

Consider dateStr (look below) as an input, write a python Program that

```
can extract the following date formats:
datestr
'23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 oct
2002\n23
october 2002\noct 23, 2002\noctober 23, 2002\n'
```

In [18]:

```
import re
import time
times='23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 oct 2002\n23 october 20
month_map = {
        'January': 1,
        'February': 2,
        'March': 3,
        'April': 4,
        'May': 5,
        'June': 6,
        'July': 7,
        'August': 8,
        'September': 9,
        'October': 10,
        'November': 11,
        'December': 12
    }
month_map2 = {
        'Jan': 1,
        'Feb': 2,
        'Mar': 3,
        'Apr': 4,
        'May': 5,
        'Jun': 6,
        'Jul': 7,
        'Aug': 8,
        'Sep': 9,
        'Oct': 10,
        'Nov': 11,
        'Dec': 12
    }

time_list= times.split('\n')
for _time in time_list:
    if(_time==""):
        continue
    _time_or=_time
    separator=" "
    for c in _time:
        if(not c.isalnum()):
            separator=c
    _time=_time.replace(separator," ")
    _t=""
    for t in _time.split():
        t=t.capitalize()
```

```
        if(t in month_map): # USe only 3 chars, and reduce the if and dict, t[:
            _t+=str(month_map[t])
        elif(t in month_map2):
            _t+=str(month_map2[t])
        else:
            _t+=t
        _t+=separator

    print(_time_or,"->",_t[:-1])
```

```
23-10-2002 -> 23-10-2002
23/10/2002 -> 23/10/2002
23/10/02 -> 23/10/02
10/23/2002 -> 10/23/2002
23 oct 2002 -> 23 10 2002
23 october 2002 -> 23 10 2002
oct 23, 2002 -> 10 23, 2002
october 23, 2002 -> 10 23, 2002
```

In [ ]: