CS 422 HW 3

Abdullah Moizuddin

Recitation Exercises

Exercise 5.2

2. Consider the data set shown in Table 5.20.

Table 5.20. Example of market basket transactions.

Customer ID	Transaction ID	Items Bought
1	0001	{a, d, e}
1	0024	{a, b, c, e}
2	0012	{a, b, d, e}
2	0031	{a, c, d, e}
3	0015	{b, c e}
3	0022	{b, d, e}
4	0029	{c d}
4	0040	{a, b, c}
5	0033	{a, d, e}
5	0038	{a, b, e}

a. Compute the support for itemsets {e}, {b, d}, and {b, d, e} by treating each transaction ID as a market basket.

b. Use the results in part (a) to compute the confidence for the association rules $\{b, d\} \rightarrow \{e\}$ and $\{e\} \rightarrow \{b, d\}$. Is confidence a symmetric measure?

c. Repeat part (a) by treating each customer ID as a market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer, and 0 otherwise).

d. Use the results in part (c) to compute the confidence for the association rules $\{b, d\} \rightarrow \{e\}$ and $\{e\} \rightarrow \{b, d\}$.

e. Suppose s_1 and c_1 are the support and confidence values of an association rule r when treating each transaction ID as a market basket. Also, let s_2 and c_2 be the support and confidence values of r when treating each customer ID as a market basket. Discuss whether there are any relationships between s_1 and s_2 or c_1 and c_2.

In [2]:
```
supp = lambda supp_count, count : supp_count/count
conf = lambda suppA,suppB: suppA/suppB
```

A.

In [3]:
```
print("support of itemset {e}:", supp(8,10))
print("support of itemset {b,d}:", supp(2,10))
print("support of itemset {b,d,e}:", supp(2,10))
```

```
support of itemset {e}: 0.8
support of itemset {b,d}: 0.2
support of itemset {b,d,e}: 0.2
```

B.

In [4]:
```
print("Confidence of {b,d} -> {e}  =  support(b,d,e)/support(b,d):",
conf(supp(2,10),supp(2,10)))
print("Confidence of {e} -> {b,d}  =  support(b,d,e)/support(e):", co
nf(supp(2,10),supp(8,10)))
```

```
Confidence of {b,d} -> {e}  =  support(b,d,e)/support(b,d): 1.0
Confidence of {e} -> {b,d}  =  support(b,d,e)/support(e): 0.25
```

- It is not symmetric. For example, if set {b,d} was purchased then we can confidently say {e} will be purchased. However,if {e} is purchaes we cant be too sure that {b,d} will be purchased.

C.

In [5]:
```
print("support for {e}:", supp(4,5))
print("support for {b,d}:", supp(5,5))
print("support for {b,d,e}:", supp(4,5))
```

```
support for {e}: 0.8
support for {b,d}: 1.0
support for {b,d,e}: 0.8
```

D.

In [6]:
```
print("Confidence of {b,d} -> {e}  =  support(b,d,e)/support(b,d):",
conf(supp(4,5),supp(4,5)))
print("Confidence of {e} -> {b,d}  =  support(b,d,e)/support(e):", co
nf(supp(4,5),supp(5,5)))
```

```
Confidence of {b,d} -> {e}  =  support(b,d,e)/support(b,d): 1.0
Confidence of {e} -> {b,d}  =  support(b,d,e)/support(e): 0.8
```

E.

- when we use the transactionID as a market basket we look to predict an association between two itemsets for a transaction. When we use customerID, we are are trying to find out what products a customer will buy if they already bought another product.

Exercise 5.6

6. Consider the market basket transactions shown in Table 5.21.

a. What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?

b. What is the maximum size of frequent itemsets that can be extracted (assuming $minsup > 0$)?

Table 5.21. Market basket transactions.

Transaction ID	Items Bought
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

c. Write an expression for the maximum number of size-3 itemsets that can be derived from this data set.

d. Find an itemset (of size 2 or larger) that has the largest support.

e. Find a pair of items, *a* and *b*, such that the rules $\{a\} \rightarrow \{b\}$ and $\{b\} \rightarrow \{a\}$ have the same confidence.

A.

In [7]:
```
total_possible_rules = lambda items : 3**items - 2**(items+1) + 1
print("maximum number of association rules:", total_possible_rules(6
))
```

```
maximum number of association rules: 602
```

B.

When minsup > 0, then any rule would work regardless. So the maximum size of the frequent itemsets is just 4.

C.

With max size = 3, for a data set with 6 items, $\binom{6}{3} = 20$

D.

In [8]:
```
print("Max support is {bread, butter} with support:", supp(5, 10))
```

```
Max support is {bread, butter} with support: 0.5
```

E.

Find confidence where conf(a,b) = conf(b,a)
This simplifies to supportCount(b) == supportCount(a)
So, {Beer} and {Cookies} have 4 occurances each and {Beer,Cookies} has 2 occurances. Therefore, {Beer,Cookies} holds.

In [9]:
```
print("Confidence of {A} -> {B} and {B} -> {A} =", conf(supp(2, 10), supp(4, 10)))
```

```
Confidence of {A} -> {B} and {B} -> {A} = 0.5
```

Exercise 5.8

8. Consider the following set of frequent 3-itemsets:

{1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}, {2, 3, 5}, {3, 4, 5}.

Assume that there are only five items in the data set.

a. List all candidate 4-itemsets obtained by a candidate generation procedure using the $F_{k-1} \times F_1$ merging strategy.

b. List all candidate 4-itemsets obtained by the candidate generation procedure in *Apriori*.

c. List all candidate 4-itemsets that survive the candidate pruning step of the *Apriori* algorithm.

A.

Frequent 2-Itemsets	Frequent 3-Itemsets	4-Itemset Candidate Generation
{1, 2}	{1, 2, 3}	{1, 2, 3, 4}
{1, 3}	{1, 2, 4}	{1, 2, 3, 5}
{2, 3}	{1, 2, 5}	{1, 2, 4, 5}
{1, 4}	{1, 3, 4}	{1, 3, 4, 5}
{2, 4}	{1, 3, 5}	{2, 3, 4, 5}
{1, 5}	{2, 3, 4}	
{2, 5}	{3, 4, 5}	
{3, 5}		
{3, 4}		
{4, 5}		

B.

Frequent 3-Itemsets	4-Itemset Candidate Generation
{1, 2, 3}	{1, 2, 3, 4}
{1, 2, 4}	{1, 2, 3, 5}
{1, 2, 5}	{1, 2, 4, 5}
{1, 3, 4}	{1, 3, 4, 5}
{1, 3, 5}	{2, 3, 4, 5}
{2, 3, 4}	
{3, 4, 5}	

C:

The pruning process of apriori is to remove all candidates are not made up of entriely of frequent subsets. In this case, each 4-Itmeset candidate has 3 subsets, so it needs to be comprised of 3 subsets which are frequent. out of all of these candidates, the only subset that survives the pruning process is:

- {1, 2, 3, 4}

Exercise 5.9

9. The *Apriori* algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size $k+1$ are created by joining a pair of frequent itemsets of size k (this is known as the candidate generation step). A candidate is discarded if any one of its subsets is found to be infrequent during the candidate pruning step. Suppose the *Apriori* algorithm is applied to the data set shown in Table 5.22 with $minsup = 30\%$, i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

Table 5.22. Example of market basket transactions.

Transaction ID	Items Bought
1	{a, b, d, e}
2	{b, c d}
3	{a, b, d, e}
4	{a, c, d, e}
5	{b, c, d, e}
6	{b, d, e}
7	{c, d}
8	{a, b, c}
9	{a, d, e}
10	{b, d}

a. Draw an itemset lattice representing the data set given in Table 5.22. Label each node in the lattice with the following letter(s):

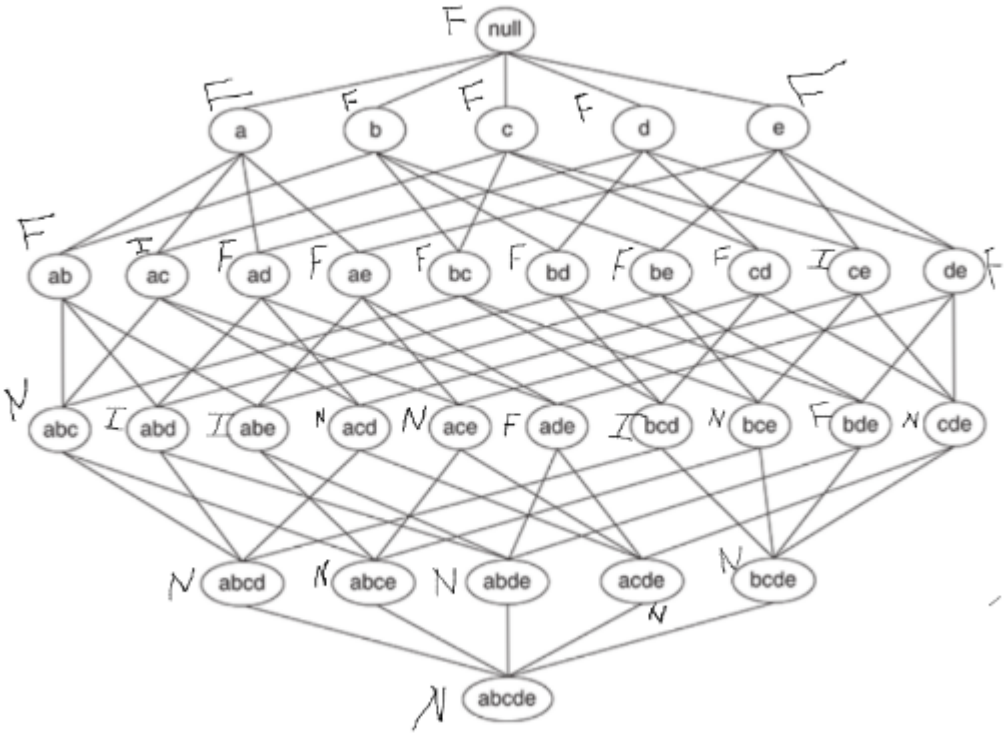
- **N**: If the itemset is not considered to be a candidate itemset by the *Apriori* algorithm. There are two reasons for an itemset not to be considered as a candidate itemset: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the candidate pruning step because one of its subsets is found to be infrequent.
- **F**: If the candidate itemset is found to be frequent by the *Apriori* algorithm.
- **I**: If the candidate itemset is found to be infrequent after support counting.

b. What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

c. What is the pruning ratio of the *Apriori* algorithm on this data set? (Pruning ratio is defined as the percentage of itemsets not considered to be a candidate because (1) they are not generated during candidate generation or (2) they are pruned during the candidate pruning step.)

d. What is the false alarm rate (i.e., percentage of candidate itemsets that are found to be infrequent after performing support counting)?

A.


null
a
b
c
d
e
ab
ac
ad
ae
bc
bd
be
cd
ce
de
abc
abd
abe
acd
ace
ade
bcd
bce
bde
cde
abcd
abce
abde
acde
bcde
abcde

B.

$\frac{16}{32} = 50\%$ of itemsets are freq.

C.

There were 11 'N's. $\frac{11}{32} = 34\%$ of itemsets were pruned

D.

There were 5 'I's. So, $\frac{5}{32} = 16\%$ were labled as infrequent

Exercise 5.12

12. Given the lattice structure shown in **Figure 5.33** and the transactions given in **Table 5.22**, label each node with the following letter(s):

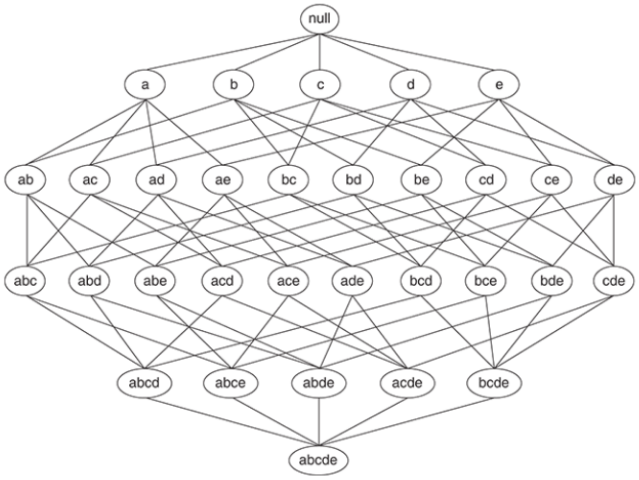
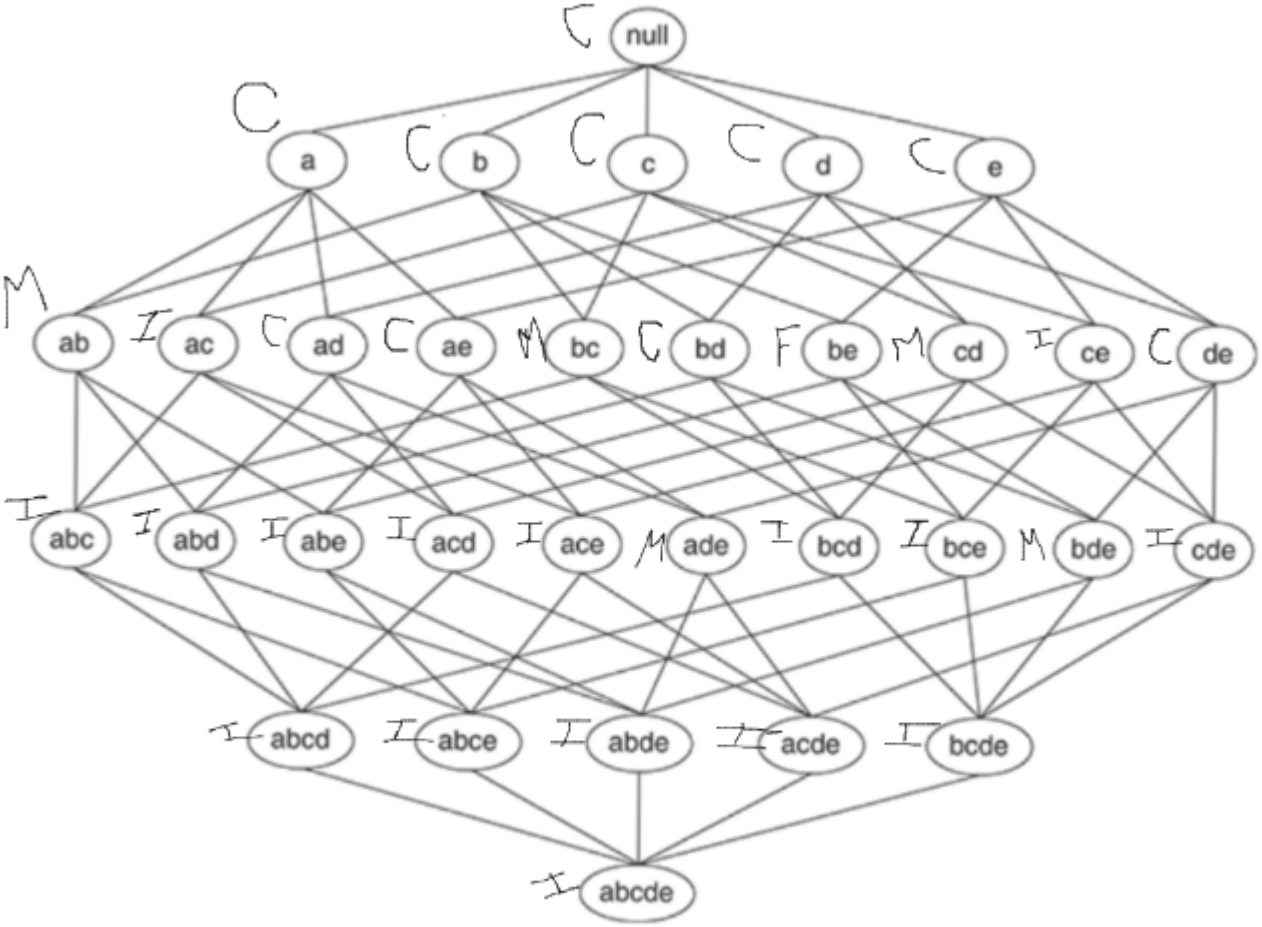


Figure 5.33.
An itemset lattice

- *M* if the node is a maximal frequent itemset,
- *C* if it is a closed frequent itemset,
- *N* if it is frequent but neither maximal nor closed, and
- *I* if it is infrequent

Assume that the support threshold is equal to 30%.


null
a
b
c
d
e
ab
ac
ad
ae
bc
bd
be
cd
ce
de
abc
abd
abe
acd
ace
ade
bcd
bce
bde
cde
abcd
abce
abde
acde
bcde
abcde

Exercise 5.13

13. The original association rule mining formulation uses the support and confidence measures to prune uninteresting rules.

a. Draw a contingency table for each of the following rules using the transactions shown in **Table 5.23**.

Table 5.23. Example of market basket transactions.

Transaction ID	Items Bought
1	{*a*, *b*, *d*, *e*}
2	{*b*, *c*, *d*}
3	{*a*, *b*, *d*, *e*}
4	{*a*, *c*, *d*, *e*}
5	{*b*, *c*, *d*, *e*}
6	{*b*, *d*, *e*}
7	{*c*, *d*}
8	{*a*, *b*, *c*}
9	{*a*, *d*, *e*}
10	{*b*, *d*}

Rules: $\{b\} \rightarrow \{c\},\ \{a\} \rightarrow \{d\},\ \{b\} \rightarrow \{d\},\ \{e\} \rightarrow \{c\},\ \{c\} \rightarrow \{a\}$.

b. Use the contingency tables in part (a) to compute and rank the rules in decreasing order according to the following measures.

i. Support.

ii. Confidence.

iii. Interest $(X \rightarrow Y) = \frac{P(X,Y)}{P(X)} P(Y)$.

iv. $\text{IS}(X \rightarrow Y) = \frac{P(X,Y)}{\sqrt{P(X)P(Y)}}$.

v. $\text{Klosgen}(X \rightarrow Y) = \sqrt{P(X,\ Y\ )} \times \max(P(Y|X)) - P(Y),\ P(X|Y) - P(X))$, where $P(Y\Big|X) = \frac{P(X,Y)}{P(X)}$.

vi. $\text{Odds ratio}(X \rightarrow Y) = \frac{P(X,Y)P(\bar{X},\ \bar{Y}\ )}{P(X,\bar{Y})P(\bar{X},Y)}$.

A.

$\{b\} \rightarrow \{c\}$

	B	$\bar{B}$	
C	3	2	5
$\bar{C}$	4	1	5
	7	3	10

$\{a\} \rightarrow \{d\}$

	A	$\bar{A}$	
D	4	5	9
$\bar{D}$	1	0	1
	5	5	10

$\{b\} \rightarrow \{d\}$

	B	$\bar{B}$	
D	6	3	9
$\bar{D}$	1	0	1
	7	3	10

$\{e\} \rightarrow \{c\}$

	E	$\bar{E}$	
C	2	4	5
$\bar{C}$	4	1	5
	6	4	10

$\{c\} \rightarrow \{a\}$

	C	$\bar{C}$	
A	2	3	5
$\bar{A}$	3	2	5
	5	5	10

In [22]:
```
def interest(suppX, suppY, suppXY):
    return "Interest", (suppXY / suppX) * suppY

def IS(suppX, suppY, suppXY):
    return "IS", suppXY / ((suppX * suppY)**0.5)

def klosgen(suppX, suppY, suppXY):
    return "Klosgen",((suppXY)**0.5) * (max(conf(suppXY, suppX) - sup
pY, conf(suppXY, suppY)-suppX))

def odds_ratio(suppXY, notXnotY, xNotY, yNotX):
    return "oddsRatio", (suppXY * notXnotY) / (xNotY*yNotX)
```

B.

$\{b\} \rightarrow \{c\}$

In [24]:
```
rules = [interest(7/10, 5/10, 3/10), IS(7/10, 5/10, 3/10),
         klosgen(7/10, 5/10, 3/10),odds_ratio(3/10,1/10,4/10,2/10)]

rules = sorted(rules, key=lambda tup:tup[1])[::-1]
print("Sorted rules for {b} -> {c}:\n")
for i in rules:
    print(i)
```

```
Sorted rules for {b} -> {c}:

('IS', 0.50709255283711)
('oddsRatio', 0.3749999999999999)
('Interest', 0.2142857142857143)
('Klosgen', -0.03912303982179756)
```

$\{a\} \rightarrow \{d\}$

In [25]:
```
rules = [interest(5/10, 9/10, 4/10), IS(5/10, 9/10, 4/10),
         klosgen(5/10, 9/10, 4/10),odds_ratio(4/10,1/10,1/10,5/10)]

rules = sorted(rules, key=lambda tup:tup[1])[::-1]
print("Sorted rules for {a} -> {d}:\n")
for i in rules:
    print(i)
```

```
Sorted rules for {a} -> {d}:

('oddsRatio', 0.8000000000000002)
('Interest', 0.7200000000000001)
('IS', 0.5962847939999439)
('Klosgen', -0.03513641844631531)
```

$\{b\} \rightarrow \{d\}$

In [26]:
```
rules = [interest(7/10, 9/10, 6/10), IS(7/10, 9/10, 6/10),
         klosgen(7/10, 9/10, 6/10),odds_ratio(6/10,0/10,1/10,3/10)]

rules = sorted(rules, key=lambda tup:tup[1])[::-1]
print("Sorted rules for {b} -> {d}:\n")
for i in rules:
    print(i)
```

```
Sorted rules for {b} -> {d}:

('Interest', 0.7714285714285715)
('IS', 0.7559289460184544)
('oddsRatio', 0.0)
('Klosgen', -0.02581988897471611)
```

$\{e\} \rightarrow \{c\}$

In [27]:
```
rules = [interest(6/10, 5/10, 2/10), IS(6/10, 5/10, 2/10),
         klosgen(6/10, 5/10, 2/10),odds_ratio(2/10,2/10,4/10,3/10)]

rules = sorted(rules, key=lambda tup:tup[1])[::-1]
print("Sorted rules for {e} -> {c}:\n")
for i in rules:
    print(i)
```

```
Sorted rules for {e} -> {c}:

('IS', 0.36514837167011077)
('oddsRatio', 0.3333333333333334)
('Interest', 0.16666666666666669)
('Klosgen', -0.07453559924999296)
```

$\{c\} \rightarrow \{a\}$

In [28]:
```
rules = [interest(5/10, 5/10, 2/10), IS(5/10, 5/10, 2/10),
         klosgen(5/10, 5/10, 2/10),odds_ratio(2/10,1/10,3/10,3/10)]

rules = sorted(rules, key=lambda tup:tup[1])[::-1]
print("Sorted rules for {c} -> {a}:\n")
for i in rules:
    print(i)
```

```
Sorted rules for {c} -> {a}:

('IS', 0.4)
('oddsRatio', 0.22222222222222227)
('Interest', 0.2)
('Klosgen', -0.04472135954999578)
```

Exercise 5.20

20. Consider the contingency tables shown in Table 5.25.

a. For table I, compute support, the interest measure, and the ϕ correlation coefficient for the association pattern {A, B}. Also, compute the confidence of rules $A \rightarrow B$ and $B \rightarrow A$.

b. For table II, compute support, the interest measure, and the ϕ correlation coefficient for the association pattern {A, B}. Also, compute the confidence of rules $A \rightarrow B$ and $B \rightarrow A$.

Table 5.25. Contingency tables for Exercise 20.

(a) Table I.

	B	$\overline{B}$
A	9	1
$\overline{A}$	1	89

(b) Table II.

	B	$\overline{B}$
A	89	1
$\overline{A}$	1	9

c. What conclusions can you draw from the results of (a) and (b)?

A.

In [29]:
```
print("interest:", interest(0.01, 0.01, 0.09)[1])
print("correlation coefficient:", ((9*89)-(1*1)) / ((10+10+90+90)**0.5))
print("Confidence {A} -> {B}:", conf(9/100, 10/100))
print("Confidence {B} -> {A}:", conf(9/100, 10/100))
```

```
interest: 0.09
correlation coefficient: 56.5685424949238
Confidence {A} -> {B}: 0.8999999999999999
Confidence {B} -> {A}: 0.8999999999999999
```

B.

In [30]:
```
print("interest:", interest(90/100, 90/100, 89/100)[1])
print("correlation coefficient:", ((9*89)-(1*1)) / ((10+10+90+90)**0.5))
print("Confidence {A} -> {B}:", conf(89/100, 90/100))
print("Confidence {B} -> {A}}:", conf(89/100, 90/100))
```

```
interest: 0.89
correlation coefficient: 56.5685424949238
Confidence {A} -> {B}: 0.9888888888888889
Confidence {B} -> {A}}: 0.9888888888888889
```

Practicum Problems

In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk
import numpy as np
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
```

Problem 2.1

2.1 Problem 1

Load the *Online Retail* dataset (**Online Retail.xlsx**) from the UCI Machine Learning Repository into **Python** using a Pandas dataframe. Using the *apriori* module from the **MLxtend** library, generate Frequent Itemsets for all transactions for the country of France. What itemset has the largest support? Set the minimum support threshold to 5% and extract frequent itemsets, and use them as input to the *association_rules* module. Use each of the confidence and lift metrics to extract the association rules with the highest values, respectively. What are the antecedents and consequents of each rule? Is the rule with the highest confidence the same as the rule with the highest lift? Why or why not?

In [33]:
```
path = r"https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx"
ds = pd.read_excel(path)
```

In [34]:
```
ds
```

Out[34]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Co
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	U Kin
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	U Kin
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	U Kin
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	U Kin
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	U Kin
...	...	...	...	...	...	...	...	
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	F
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	F
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	F
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	F
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	F

541909 rows × 8 columns

In [35]:
```
France = ds["Country"] == "France"
france_ds = ds[France][["InvoiceNo", "Description"]]

france_ds[["InvoiceNo", "Description"]]
cols = france_ds["Description"].unique()
rows = france_ds["InvoiceNo"].unique()
txns = pd.DataFrame(columns=cols)
txns.insert(column="InvoiceNo", loc=0, value=rows)
txns = txns.replace(np.nan, False)
for idx, row in france_ds.iterrows():
    invoice, item= row["InvoiceNo"], row["Description"]
    txns.loc[txns['InvoiceNo'] == invoice, item] = True

txns = txns.drop("InvoiceNo", axis=1)
```

In [36]:
```
freq_itemsets = apriori(txns, min_support=0.05, use_colnames=True)
freq_itemsets
```

Out[36]:

	support	itemsets
0	0.086768	(ALARM CLOCK BAKELIKE PINK)
1	0.080260	(ALARM CLOCK BAKELIKE RED)
2	0.084599	(ALARM CLOCK BAKELIKE GREEN)
3	0.138829	(ROUND SNACK BOXES SET OF4 WOODLAND)
4	0.106291	(SPACEBOY LUNCH BOX)
...	...	...
125	0.071584	(POSTAGE, SET/6 RED SPOTTY PAPER CUPS, SET/20 ...
126	0.071584	(POSTAGE, SET/6 RED SPOTTY PAPER PLATES, SET/2...
127	0.058568	(PLASTERS IN TIN SPACEBOY, PLASTERS IN TIN WOO...
128	0.084599	(SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT...
129	0.069414	(POSTAGE, SET/6 RED SPOTTY PAPER CUPS, SET/6 R...

130 rows × 2 columns

In [37]:
```
rules_confidence = association_rules(freq_itemsets, metric='confidenc
e',min_threshold=1)
rules_confidence
```

Out[37]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.065076	0.67462	0.065076	1.0	1.482315	0.02117

In [38]:
```
rules_lift = association_rules(freq_itemsets, metric='lift',min_thres
hold=11)
rules_lift
```

Out[38]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(CHILDRENS CUTLERY SPACEBOY)	(CHILDRENS CUTLERY DOLLY GIRL)	0.058568	0.062907	0.05423	0.925926	14.719029	0.0505
1	(CHILDRENS CUTLERY DOLLY GIRL)	(CHILDRENS CUTLERY SPACEBOY)	0.062907	0.058568	0.05423	0.862069	14.719029	0.0505

The rule with highest confidence is not the same as the rule with the highest lift. Confidence is a measure of how often {B} occurs with {A}. In the Fomula is supportcount(A U B) / supportcount(A). Lift adds a factor of 1/supportcount(B). Thus if B occured with A every time the lift would be equal 1. But as B increases lift increases also which means that {A} -> {B} is an actual rule and {B} isnt just a frequent item.

Problem 2.2

2.2 Problem 2

Load the *Extended Bakery* dataset (**75000-out2-binary.csv**) into **Python** using a Pandas dataframe. Calculate the binary correlation coefficient Φ for the *Chocolate Coffee* and *Chocolate Cake* items. Are these two items symmetric binary variables? Provide supporting calculations. Would the association rules $\{Chocolate\ Coffee\} \implies \{Chocolate\ Cake\}$ have the same value for Φ as $\{Chocolate\ Cake\} \implies \{Chocolate\ Coffee\}$?

In [39]:
```
pathcsv = r"75000-out2-binary.csv"
bakery_ds = pd.read_csv(pathcsv)
```

In [42]:
```
name1 = 'Chocolate Coffee'
name2 = 'Chocolate Cake'

cnc = bakery_ds[[name1,name2]]

condition1 = cnc[name1]==1
condition2 = cnc[name2]==1

data = cnc.groupby([condition1,condition2]).count()
data
```

Out[42]:

		Chocolate Coffee	Chocolate Cake
Chocolate Coffee	**Chocolate Cake**		
False	False	65802	65802
	True	2962	2962
True	False	2933	2933
	True	3303	3303

In [43]:
```
print(cnc[name1].corr(cnc[name2]))
print(cnc[name2].corr(cnc[name1]))
```

```
0.48556649252787826
0.48556649252787837
```

These items are symmetric binary variables. The association rules have the same corrolation values because to calculate it we do txns where both items occur multiplied by txns where neither items occur then subtract the product of the occurances for each item. then you divite that by the root of the sum of total occurances and not-occurances for each item. As seen order doesnt matter for corrolation as corrolation compares the items to each other.