Homework 3

Recitation Exercises

In [1]:
```
support = lambda occurances, records: occurances / records
confidence = lambda supAB, supA: supAB / supA
```

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.

A:

In [2]:
```
print("Support for {e}: ", support (8, 10))
print("Support for {b, d}: ", support (2, 10))
print("Support for {b, d, e}: ", support (2, 10))
```

```
Support for {e}:  0.8
Support for {b, d}:  0.2
Support for {b, d, e}:  0.2
```

B:

In [3]:
```
print("Confidence for {b, d} -> {e} =  s({b, d, e}) / s({b, d}) = ", confidence(support(2, 10), support(2
print("Confidence for {e} -> {b, d} = s({b, d, e}) / s({e})= ", confidence(support(2, 10), support(8, 10)
```

```
Confidence for {b, d} -> {e} =  s({b, d, e}) / s({b, d}) =  1.0
Confidence for {e} -> {b, d} = s({b, d, e}) / s({e})=  0.25
```

- Confidence is evidently not a symetric measure, which is good, becuase it shows that if item set {b, d} is purchased we can say with confidence item set {e} will probably be purchased but if item set {e} is purchased it will not say much about wheather itemset {b, d} will be purchased

C:

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

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

D:

In [5]:
```
print("Confidence for {b, d} -> {e} =  s({b, d, e}) / s({b, d}) = ", confidence(support(4, 5), support(4,
print("Confidence for {e} -> {b, d} = s({b, d, e}) / s({e})= ", confidence(support(4, 5), support(5, 5)))
```

```
Confidence for {b, d} -> {e} =  s({b, d, e}) / s({b, d}) =  1.0
Confidence for {e} -> {b, d} = s({b, d, e}) / s({e})=  0.8
```

E:

- When we use the transactionID as a market basket, we're looking to predict the assocation between two itemsets for a given transaction, but with CustomerID, we're trying to predict the association between two itemsets from a customer's purchase history. It would be dangerous to say that a high confidence or support in one may lead to high confidence or support in another. If a customer has ever bought a common item at anypoint in time say "batteries" and later buys another common item say "eggs" it is very easy to show high support/confidence. However if each customer had only purchased those items in a few transactions relative to their total transactions, the support/confidence would lower when using transactionID as a market basket.

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 [6]:
```
maxRules = lambda items : 3**items - 2**(items+1) + 1
totalUniqueItems = 6

print("Max association rules:", maxRules(totalUniqueItems))
```

```
Max association rules: 602
```

B:

With minsup > 0, then we just need to find any rule with the most item sets. That would be: **4**

C:

Max number of itemsets with size three from a data set of 6 unique items: $\binom{6}{3} = \mathbf{20}$

D:

In [7]:
```
N = 10
## for {bread, butter}
suppotCount = 5
print("Max support is {bread, butter} with support:", support(suppotCount, N))
```

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

E:

We need to find confidences where s{A, B} / S{A} == S{A, B} / S(B)

S{A} == S{B}

{Beer} has 4 occurances

{Cookies} has 4 occurances

{Beer, Coookies} has 2 occurances

{Beer, Cookies} holds this property

In [8]:
```
print("Confidence of {A} -> {B} and {B} -> {A} =", confidence(support(2, 10), support(4, 10)))
```

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

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 the case of this problem, each 4-Itmeset candidate has 3 subsets, so it needs to be made up of 3 frequent subsets, of this candidates, the only one that makes it past this pruning process is:

{1, 2, 3, 4}

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.

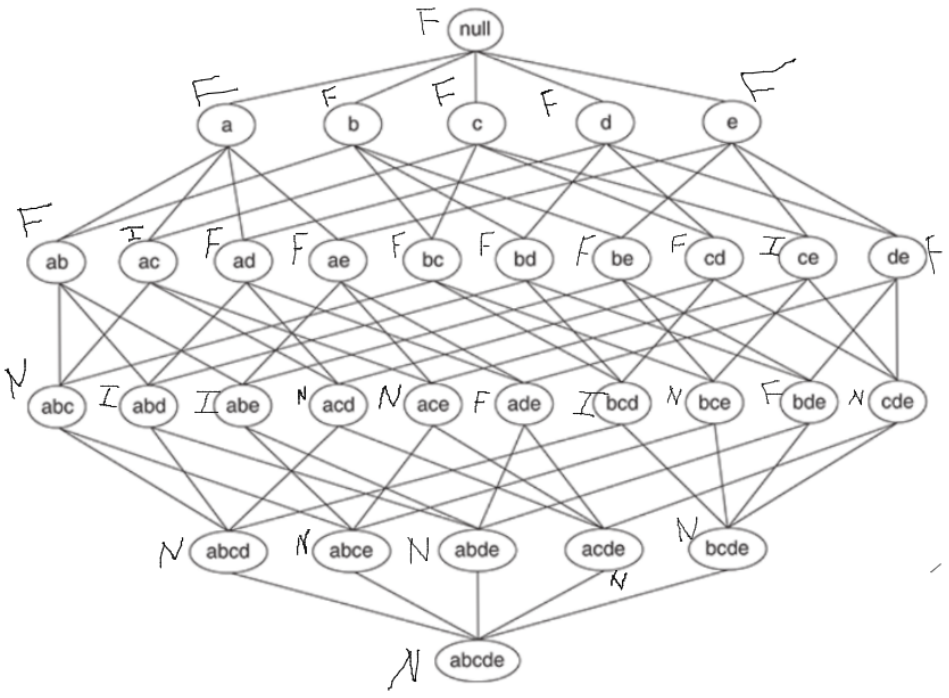
b. What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

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:

file:///home/aakef/Downloads/5-9-lattice.png



B:

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

C:

The number of "N"s that are produced is 11, so $\frac{11}{32} = 34\%$ of itemsets that were pruned

D:

The number of "I"s that are produced is 5, so $\frac{5}{32} = 16\%$ of itemsets that were labled as infreqent by the apriori algorithim after support counting

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):

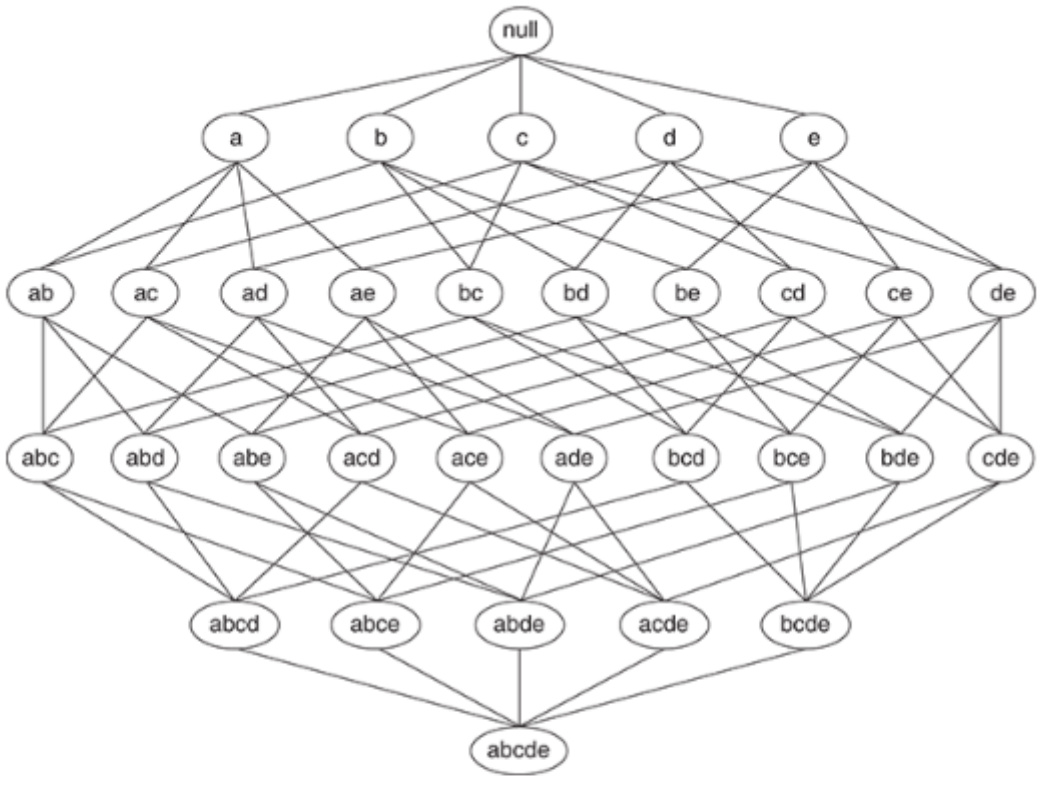
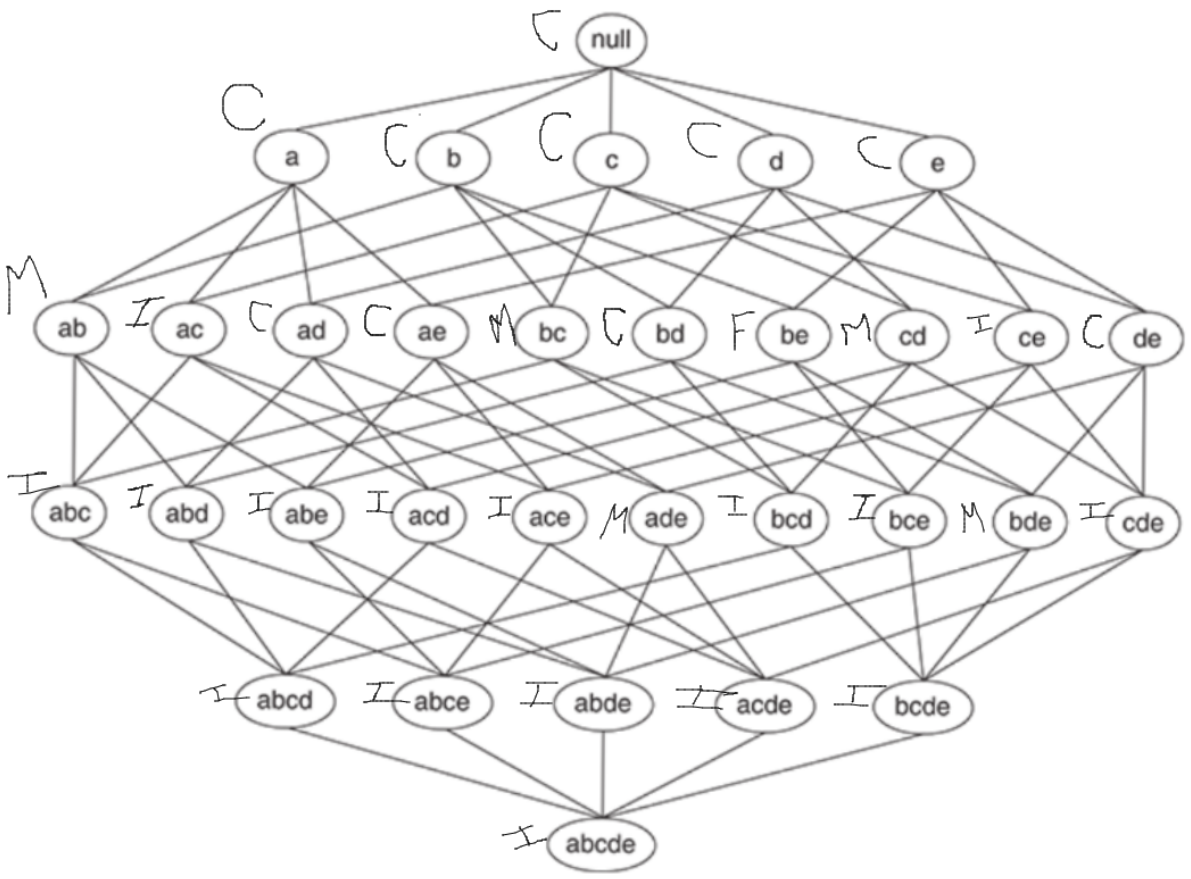


Figure 5.33.
An itemset lattice

- *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%.

file:///home/aakef/Downloads/5-12-lattice.png



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 $P(Y|X) = \frac{P(X,Y)}{P(X)}$.

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

A:

{b} -> {c}

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

{a} -> {d}

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

{b} -> {d}

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

{e} -> {c}

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

{c} -> {a}

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

B:

In [9]:
```
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(confidence(suppXY, suppX) - suppY, confidence(suppXY, suppY)-

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

{b} -> {c}

In [10]:
```
suppX = 7/10
suppY = 5/10
suppXY = 3/10
notXnotY = 1/10
xNotY = 4/10
yNotX = 2/10
rules = [interest(suppX, suppY, suppXY), IS(suppX, suppY, suppXY),
         klosgen(suppX, suppY, suppXY),oddsRatio(suppXY, notXnotY, xNotY, yNotX)]

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} -> {d}

In [11]:
```
suppX = 5/10
suppY = 9/10
suppXY = 4/10
notXnotY = 1/10
xNotY = 1/10
yNotX = 5/10
rules = [interest(suppX, suppY, suppXY), IS(suppX, suppY, suppXY),
         klosgen(suppX, suppY, suppXY),oddsRatio(suppXY, notXnotY, xNotY, yNotX)]

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} -> {d}

In [12]:
```
suppX = 7/10
suppY = 9/10
suppXY = 6/10
notXnotY = 0/10
xNotY = 1/10
yNotX = 3/10
rules = [interest(suppX, suppY, suppXY), IS(suppX, suppY, suppXY),
         klosgen(suppX, suppY, suppXY),oddsRatio(suppXY, notXnotY, xNotY, yNotX)]

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} -> {c}

In [13]:
```
suppX = 6/10
suppY = 5/10
suppXY = 2/10
notXnotY =21/10
xNotY = 4/10
yNotX = 4/10
rules = [interest(suppX, suppY, suppXY), IS(suppX, suppY, suppXY),
         klosgen(suppX, suppY, suppXY),oddsRatio(suppXY, notXnotY, xNotY, yNotX)]

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}:

('oddsRatio', 2.6249999999999996)
('IS', 0.36514837167011077)
('Interest', 0.16666666666666669)
('Klosgen', -0.07453559924999296)
```

{c} -> {a}

In [14]:
```
suppX = 5/10
suppY = 5/10
suppXY = 2/10
notXnotY = 1/10
xNotY = 3/10
yNotX = 3/10
rules = [interest(suppX, suppY, suppXY), IS(suppX, suppY, suppXY),
         klosgen(suppX, suppY, suppXY),oddsRatio(suppXY, notXnotY, xNotY, yNotX)]

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', 0.2222222222222227)
('Interest', 0.2)
('Klosgen', -0.04472135954999578)
```

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 [17]:
```
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}:", confidence(9/100, 10/100))
print("Confidence {B} -> {A}:", confidence(9/100, 10/100))
```

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

B:

In [18]:
```
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}:", confidence(89/100, 90/100))
print("Confidence {B} -> {A}}:", confidence(89/100, 90/100))
```

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

C:

We would ideally want more records in the diagonal, (i.e) AB or Not (A and B) and less in the other boxes. Also corelation is the same

Practicum Exercises

In [19]:
```
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 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 [20]:
```
path = r"https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx"
df = pd.read_excel(path)
```

In [22]:
```
df
```

Out[22]:

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

541909 rows × 8 columns

In [23]:
```
isFrance = df["Country"] == "France"
france_df = df[isFrance][["InvoiceNo", "Description"]]

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

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

In [24]:
```
freq_sets = apriori(txn_df, min_support=0.05, use_colnames=True)
```

In [25]:
```
rules_conf = association_rules(freq_sets, metric="confidence", min_threshold=1)
rules_conf
```

Out[25]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.065076	0.67462	0.065076	1.0	1.482315	0.021174	inf

In [26]:
```
rules_lift = association_rules(freq_sets, metric="lift", min_threshold=11)
rules_lift
```

Out[26]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(CHILDRENS CUTLERY SPACEBOY)	(CHILDRENS CUTLERY DOLLY GIRL)	0.058568	0.062907	0.05423	0.925926	14.719029	0.050546	12.650759
1	(CHILDRENS CUTLERY DOLLY GIRL)	(CHILDRENS CUTLERY SPACEBOY)	0.062907	0.058568	0.05423	0.862069	14.719029	0.050546	6.825380

- The rule with the highest confidence, does not == the rule with the highest lift
- Confidence simply measures, how often B occurs when A occurs, to predict the liklihood that if A occurs B will occur
- Lift takes into account how often B occurs on it's own. For example if B were to always occur it would have a support of 1, and the lift would be == to the confidence, but the less B occurs the more the lift will increase compared to the confidence, meaning that there is a higher likelihood that A actually causes B to occur, and that B is not just a frequent item on it's own.

Problem 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 [27]:
```
local_path = r"75000-out2-binary.csv"
bakery_df = pd.read_csv(local_path)
```

In [29]:
```
item1_name = 'Chocolate Coffee'
item2_name = 'Chocolate Cake'

cake_n_coffee = bakery_df[[item1_name,
                   item2_name]]

cond_1 = cake_n_coffee[item1_name] == 1
cond_2 = cake_n_coffee[item2_name] == 1
stats = cake_n_coffee.groupby([cond_1,
                       cond_2]).count()
stats
```

Out[29]:

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

In [30]:
```
print(cake_n_coffee["Chocolate Coffee"].corr(cake_n_coffee["Chocolate Cake"]))
print(cake_n_coffee["Chocolate Cake"].corr(cake_n_coffee["Chocolate Coffee"]))
```

```
0.48556649252788
0.48556649252788003
```

- Yes the items are symetric binary variables

Yes the association rules would have the same correlation values going each way becuase we take the product of txns where both items occur and neither occur and subtract the product of the amount of occurances of each item. we then divide that entire difference by the root of the sum of the total amount of occcurances and non-occurances of each item. The order of the ruling does not affect this calucation which makes sense becuase correlation simply measures the behavior of both feeatures and compares them.